http://radio-mir.com

Индексы: 48925, 45995

радиомир

загрузочный

В НОВЫЙ ВЕК —
С НОВЫМ НАЗВАНИЕМ!

5•2002

Ваш компьютер

ЗОЛОТАЯ КОЛЛЕКЦИЯ

LINGVO 2002 4В1

4В1 СЛОВАРЬ

Lingvo

REANIMATOR XP

РУССКАЯ ВЕРСИЯ

КТО ХОЧЕТ СТАТЬ МИЛЛИОНЕРОМ

О, СЧАСТЛИВЧИК!

КТО ХОЧЕТ СТАТЬ МИЛЛИОНЕРОМ

РУССКИЙ Pokе

ZEMФИРА

+ НОЧНЫЕ СНАЙПЕРЫ

7 АЛЬБОМОВ

192 kbs • 44 kHz • STEREO

MP

LEARN TO SPEAK ENGLISH

МУЛЬТИМЕДИЙНЫЙ УЧЕБНИК АНГЛИЙСКОГО ЯЗЫКА

ПОЛНЫЙ ИНТЕРАКТИВНЫЙ КУРС

НАУЧИТЕСЬ ГОВОРИТЬ

VIDEO FOR PC

ВСПОМНИТЬ ВСЁ

АРНОЛЬД ШВАРЦЕНЕГГЕР

В фильме Пауля Верхувена

DVD КАЧЕСТВО

www.KM.ru
www.4User.ru

ЭНЦИКЛОПЕДИЯ

Персонального компьютера и Интернета

Кирилла и Мефодия

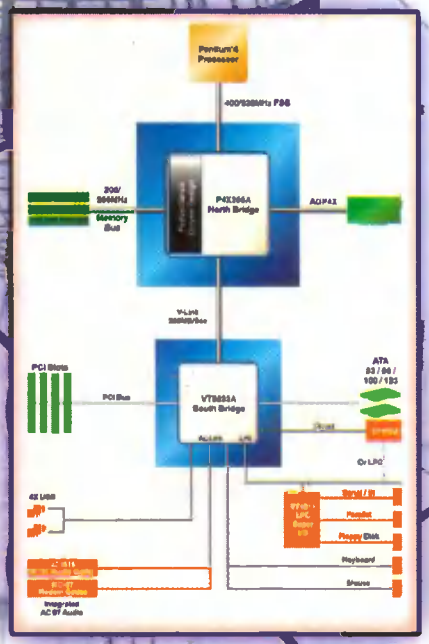
ПРОСТО СОФТ

ВСЁ ДЛЯ ХАКЕРА

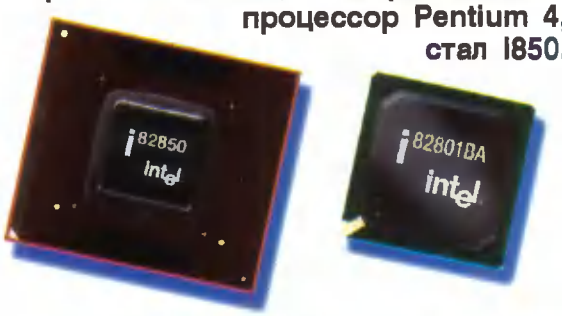
ЗЕМФИРА

РУССКИЙ ШАНСОН

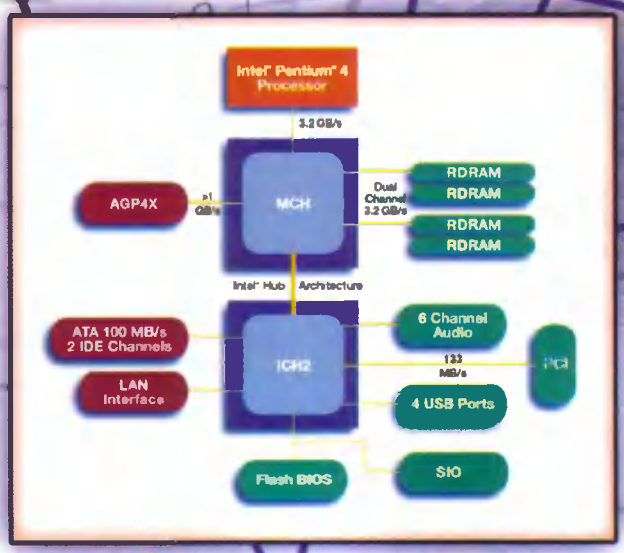
В.Куц. Процессоры Pentium 4 и чипсеты для него

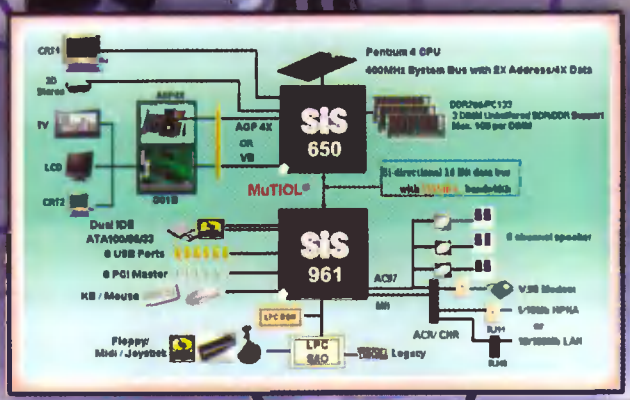


Первым чипсетом, поддерживающим процессор Pentium 4, стал I850.



Процессоры Pentium 4 выпускаются в различном конструктивном исполнении: в корпусе PGA 423 (справа) и в корпусе mPGA 478 (слева).





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:
117454, РФ, г.Москва, а/я 37 или
141406, РФ, г.Москва, Химки-6,
ул.Библиотечная, 18-84;

220095, РБ, г.Минск-95, а/я 199

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

Коммерческий отдел:
в Москве (095) 139-75-77,
а Минске (017) 221-01-10
E-mail: rm-sales@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: 117454, РФ, г.Москва,
ул.Коштоянца, 6 — 233, тел. (095) 105-99-89

Подписано к печати 27.03.2002 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободная

Отпечатано в типографии
ЗАО "Красногорская типография"

Заказ 844. Тираж 2000 экз.

радиомир
Ваш компьютер

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

№5/май/2002

ЧИТАЙТЕ В НОМЕРЕ

Дорогие друзья!

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

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

Редакция.

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

Н.ЛУТКОВСКАЯ, В.ЛУТКОВСКИЙ.
E-mail: SRLSA@bsu.by

Машина Тьюринга — прототип квантового компьютера

В 1936 г. английский математик Алан Тьюринг (1912 — 1954) предложил абстрактную схему автоматического устройства, принципиально пригодного для реализации любого алгоритма [1]. Этот автомат, получивший название "машина Тьюринга", стал прототипом современного компьютера. Разрабатывая компьютеры на новых физических принципах, специалисты снова и снова возвращаются к пионерской идее Тьюринга. Дело в том, что возможность реализации того или иного алгоритма с помощью такой машины гарантирует успех его реализации и на классическом компьютере, и на квантовом.

Схема машины Тьюринга

Запоминающим устройством в машине Тьюринга (рис.1) служит лента, разделенная на ячейки №1, №2, ... так, что эта лента имеет начало (ячейка №1), но не имеет конца (простирается в бесконечность как последовательность натуральных чисел). В каждой из ячеек может быть записан ноль или единица. Над лентой помещается головка T, управляемая автоматом L с конечной памятью.

Действие машины Тьюринга однозначно определяется исходным заполнением ячеек ленты и оператором преобразования управляющего автомата, который может быть задан таблицей переходов [1]. Обозначим через S_i ($S_0=0$, $S_1=1$) символ, считываемый головкой; через R_j (R_0 — стоп, R_1 — влево, R_2 — вправо) — команду на перемещение головки, и через q_k ($k=1, 2, \ldots, n$) — состояние управляющего автомата. Тогда таблица переходов управляющего автомата может быть представлена, как показано в табл.1.

Как видно из табл.1, действие автомата L зависит от входа и его состояния q. Определенным значениям S_i и q_i соответствует определенная совокупность значений трех величин — S, R и q, обозначающих соответственно: какой символ S запишет головка на ленту, какова будет команда R на перемещение головки и в какое новое состояние q перейдет автомат L. Следует иметь в виду, что среди состояний q автомата L должно быть по крайней мере одно такое его состояние q^*, при котором: головка не изменяет символ S, команда $R=R_0$ (стоп), и автомат L остается в состоянии покоя q^*. Придя в состояние q^*, автомат заканчивает выполнение алгоритма, и дальнейшая работа машины Тьюринга прекращается.

Пусть, например, таблица перехода автомата L имеет вид, аналогичный табл.2.

Тогда, если в начальный момент времени автомат находится в состоянии q_1, а головка расположена над ячейкой, в которой записан символ *1*, то головка будет перемещаться вправо до тех пор, пока не обнаружит ячейку с символом *0*, заменит его символом *1* и остановится. Если начальное состояние системы (состояние в нулевом такте) и заполнение ленты соответствуют показанным на рис.1, то, согласно табл.2, система в последующие два такта перейдет в состояния, показанные на рис. 2 и 3, и на втором такте остановится.

Табл. 1

Состояние \ Вход	$S_0=0$	$S_1=1$
q_1	S_0,R_2,q_k	S_1,R_1,q_m
q_2	S_1,R_0,q_s	S_0,R_1,q_i
q_3	S_1,R_1,q_p	S_0,R_2,q_s
...	...	...

Табл. 2

q \ S	0	1
q^*	$0,R_0,q^*$	$1,R_0,q^*$
q_1	$1,R_0,q^*$	$1,R_2,q_1$

Рис. 1

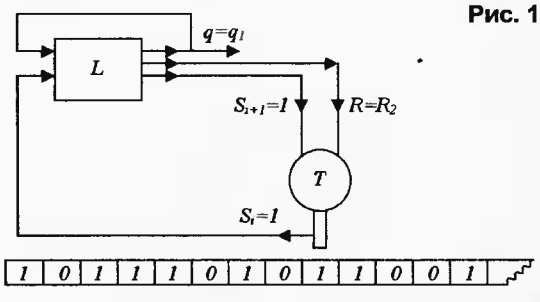


Рис. 2

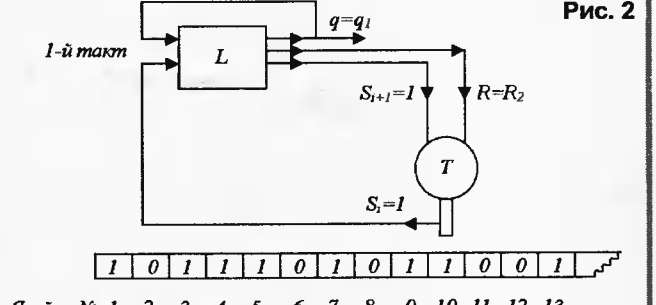
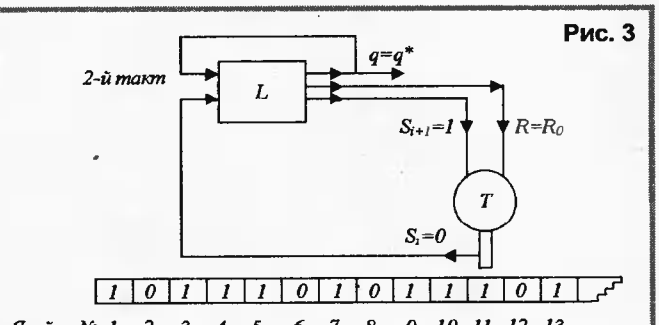


Рис. 3



Автомат L работает по тактам. На его вход поступает информация о символе (*0* или *1*), считываемом головкой с ленты. Под влиянием команд, получаемых каждый раз от автомата L, головка может оставаться на месте или передвигаться на одну ячейку вправо или влево. Одновременно головка получает от автомата L команды, под влиянием которых она может заменить символ, записанный в ячейке, над которой она находится.

Приведенный пример показывает лишь одну из самых простых задач, решаемых машиной Тьюринга. При соответствующем расширении таблицы переходов можно заставить машину Тьюринга решать сколь угодно сложные задачи. Во всяком случае, она может решить любые задачи, которые способна решать какая-либо другая машина. Однако при этом следует иметь в виду, что практическое применение автоматов, построенных по схеме машины Тьюринга, нецелесообразно, ибо число шагов, необходимых для решения сколько-нибудь сложных задач, чрезвычайно велико, а значит, велико и время решения.

Тем не менее, теория машины Тьюринга имеет большое значение для решения таких важных вопросов как существование алгоритма решения того или иного класса задач, а значит, и для выяснения того, какие функции могут, а какие принципиально не могут выполняться, в частности, таким универсальным автоматом как персональный компьютер.

Прототип квантового компьютера

При необходимости построить действующую модель машины Тьюринга по приведенной схеме, нам пришлось бы использовать большое количество механических узлов для перемещения головки или перемотки ленты с символами. Очевидно, что такая машина будет работать очень медленно.

Выгоднее в качестве ленты применить полупроводниковое запоминающее устройство, а вместо механического перемещения головки изменять состояние счетчика адреса. Еще выгоднее вместо электрических сигналов использовать световое излучение. Ослабляя эти сигналы до уровня отдельных фотонов и используя цепочку атомов в качестве ленты запоминающего устройства, мы получим квантовый вариант машины Тьюринга. Но работать такая машина будет уже не по законам Ньютона, а по законам квантовой механики. И логические элементы, необходимые для реализации автомата L на рис.1, будут существенно отличаться от привычных нам цифровых микросхем.

Квантовая логика

Обычно вычислительные устройства строятся из логических элементов (вентилей) и триггеров. Логически активные элементы — устройства, выполняющие простейшие действия по преобразованию информации. Триггер — устройство, имеющее два устойчивых состояния и способное переключаться из состояния "0" в состояние "1" и обратно. Вне зависимости от физического представления "0" и "1" (в виде электрического сигнала, характеристик намагничивания, света, состояния молекул, и т.п.), логически активный элемент реализует какую-либо функцию булевой алгебры (или двоичной логики). На языке квантовой механики состояние квантового бита (кубита) описывается волновой функцией ψ. Состояние кубита можно наглядно представить себе как положение вектора ψ единичной длины на плоскости, где оси X соответствует значение $|0\rangle$, а оси Y — значение $|1\rangle$. На рис.4 показано действие квантового вентиля NOT (НЕ) на кубит. Такое представление возможно для вещественных α и β.

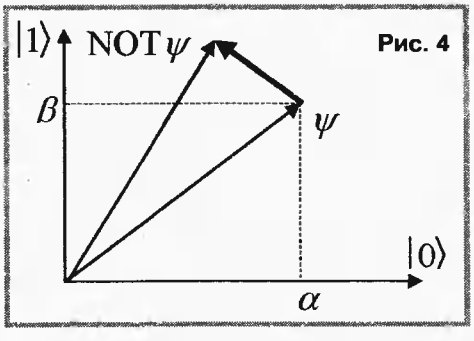


Рис. 4

Тогда управление кубитом с помощью квантового вентиля можно уподобить вращению единичного вектора ψ вокруг начала координат на различные углы и/или его отражению относительно биссектрисы прямого угла, образованного осями координат.

Рассмотрим для примера квантовый вентиль NOT (НЕ), осуществляющий над кубитом логическую операцию инверсии (отрицания) — действуя на $|0\rangle$, он дает $|1\rangle$, а действуя на $|1\rangle$ — $|0\rangle$. Коэффициенты α и β при этом не меняются (в силу линейности вентиля). Таким образом

$$NOT\,\psi = \alpha|1\rangle + \beta|0\rangle.$$

Благодаря тому что кубит может находиться в состоянии суперпозиции $|1\rangle$ и $|0\rangle$, помимо обычных логических операций, таких как NOT (НЕ), AND (И), OR (ИЛИ) и т.п., можно определить и новые, не имеющие классических аналогов, например, задать операцию $\sqrt{NOT}$ так, чтобы $\sqrt{NOT}\cdot\sqrt{NOT} = NOT$.

Для того чтобы рассмотренный прототип квантового компьютера мог выполнять вычисления, необходимо обеспечить:

1) хранение квантовой информации в кубитах;

2) обработку этой информации с помощью квантовых вентилей;

3) считывание информации [2].

Работы по созданию подобных устройств ведутся уже около двух десятилетий и уже приводят к ощутимым результатам [3]. Современный уровень технологии уже сегодня позволяет реально создавать квантовые транзисторы предельно малых размеров. На повестке дня — квантовые компьютеры. Не исключено, что в будущем будет создан компьютер на одном атоме. Звучит неправдоподобно, но идея реализации квантовой вычислительной системы на одном атоме имеет под собой прочный теоретический фундамент [4]. Об этом, как и о возможности реализации машины Тьюринга на квантовых элементах, мы узнали из лекции, прочитанной в Белорусском государственном университете российским математиком И.В.Воловичем.

Мы рассмотрели только принципиальную возможность реализации машины Тьюринга на квантовых логических элементах. Но на практике возникает много проблем. Как заставить атомы выстроиться в цепочку? Как обеспечить надежное хранение и считывание информации, если фотоны будут испускаться и поглощаться спонтанно? Чтобы ответить на эти вопросы, необходимо более глубоко рассмотреть физические принципы построения элементов квантовых компьютеров. Этому будет посвящена отдельная статья.

Литература

1. Лернер А.Я. Начала кибернетики. — М.: 1967, 400 с.

2. Scalable quantum computing with quantum optical systems/ T. Colarco, D.Jaksch, J.I.Cirac, P.Zoller/ — Abstracts of XVII International Conference on Coherent and Nonlinear Optics. P. FN2.

3. История и география квантовых компьютеров. — Радиомир. Ваш компьютер, 2001, №12. С.2–3.

4. Volovich I.V. Atomic Quantum Computer. — arXiv:quant-ph/9911062, 14 Nov. 1999. — 5 p.

В.КУЦ,
E-mail: victor_kootz@mail.ru

Процессор Pentium 4 и чипсеты для него

Продолжая тему о современных процессорах и чипсетах для них, начатую в прошлом номере [1], сегодня поговорим о процессоре Pentium 4, по задумкам его разработчика, фирмы Intel, призванного стать безоговорочным лидером в процессорной гонке последних лет.

Немного о самом процессоре Pentium 4

Первоначальный вариант процессора Pentium 4 (фирменное обозначение ядра — Willamette) был выпущен в конце 2000 г. и производился по технологическому процессу 0,18 мкм. Он содержал 42 млн. транзисторов, занимающих площадь 217 мм^2. Такие большие размеры кристалла обусловлены тем, что Pentium 4 создавался, в первую очередь, с большим заделом на будущее, в том числе и на применение в процессе производства более тонкого технологического процесса. Напряжение питания ядра процессора —1,7 В, изначально для него был разработан разъем Socket 423, уже через год бесславно закончивший свое существование в связи с переводом процессора на новый корпус типа mPGA 478, отличающийся значительно меньшими габаритами.

Процессор Pentium 4 имеет новую микроархитектуру, получившую специальное название — NetBurst. В числе наиболее значимых новшеств, появившихся в NetBurst — 400-мегагерцовая процессорная шина, технологии Hyper-Pipelined Technology и Rapid Execution Engine, а также новый набор команд SSE2.

400-мегагерцовая системная шина, иногда еще называемая Quad Pumped Bus, у Intel Pentium 4 имеет тактовую частоту в 100 МГц. Однако, благодаря использованию технологии Quad Pumping, по этой шине передается четыре блока данных за один такт, что дает эффективную рабочую частоту 400 МГц и обеспечивает пропускную способность 3,2 Гб/с.

Hyper-Pipelined Technology. Pentium 4 имеет очень длинный конвейер, состоящий из 20 стадий, тогда как конвейер процессоров семейства P5 — всего 5 стадий, семейства P6 — 10, а AMD Athlon/Duron — 12 стадий. Более длинный конвейер, с одной стороны, позволяет достичь высоких тактовых частот даже при существующем уровне технологии, но, с другой стороны, при обнаружении неправильно предсказанного перехода весь конвейер останавливается, и его перезагрузка требует значительного времени. Поэтому преимущества технологии Hyper-Pipelined в полной мере начинают проявляться только сейчас, после достижения процессором рабочих частот 2 ГГц и более. Кроме того, в Intel Pentium 4 интегрирован более совершенный механизм предсказания переходов, и количество ошибочно предсказанных переходов у него в среднем на 33% меньше, чем у процессоров семейства P6.

Rapid Execution Engine — блок выполнения арифметико-логических операций, причем он состоит из двух ALU-модулей, работающих параллельно, да еще и на удвоенной тактовой частоте по отношению к тактовой частоте процессора. Таким образом, весь Rapid Execution Engine способен выполнять до четырех целочисленных операций за один рабочий такт процессора.

Streaming SIMD Extensions 2 (SSE2) содержит 144 новые SIMD-инструкции, предназначенные для увеличения производительности системы при обработке аудио- и видеоданных, которые добавлены к базовому набору SSE-инструкций, реализованному ранее в процессоре Pentium III. Из них 68 расширяют возможности старых SIMD-инструкций по работе с целыми числами, а 76 являются совершенно новыми, позволяющими оперировать со 128-разрядными числами (как целыми, так и вещественными с двойной точностью).

В начале 2002 г., когда были выпущены процессоры Pentium 4 на ядре Northwood под разъем Socket 478, для них началась новая эпоха. Эти процессоры, выполненные с соблюдением норм техпроцесса 0,13 мкм, имели увеличенный до 512 Кб кэш L2 и работали при напряжении питания ядра 1,5 В. Как следствие применения нового техпроцесса, не только рабочая частота этих процессоров стала более высокой, но и их тепловыделение значительно уменьшилось (49,8 Вт рассеиваемой мощности у 2,2-гигагерцовой версии Northwood против 75 Вт у Willamette 2 ГГц). Любопытно отметить, что процессор на новом ядре содержит 55 млн. транзисторов (у Willamette, как уже упоминалось, "всего" 42 млн.), что связано с изменением объема L2. В ядре Northwood наконец-то стали использоваться медные соединения вместо алюминиевых, а его площадь уменьшилась до 145 мм^2. В ближайшей перспективе должен состояться перевод процессоров Pentium 4 на высокоскоростную 533-мегагерцовую системную шину Quad Pumped (что увеличит ее пиковую пропускную способность до 4,27 Гб/с!), дающую им возможность работать с памятью PC2700 DDR SDRAM в синхронном режиме, обеспечивающем, как известно, за счет отказов от лишних тактов ожидания, наибольшую производительность. Все это дает основание высказать предположение, что процессоры Pentium 4 наконец-то сумели, в той или иной степени, раскрыть тот потенциал, который был изначально заложен в них разработчиками. Похоже, что в сегменте высокопроизводительных настольных систем как минимум еще год, вплоть до появления процессоров AMD семейства Hammer, процессору Athlon XP очень трудно будет с ними конкурировать.

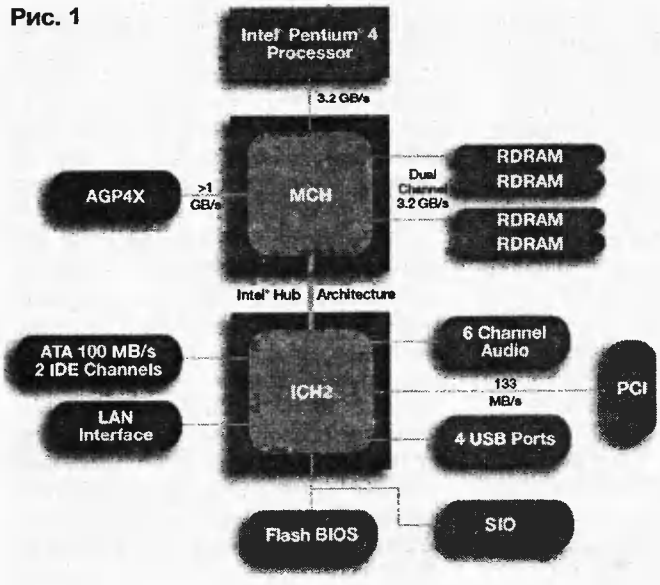
В середине следующего, 2003 г. планируется выпустить следующее процессорное ядро, пока известное как Prescott. Оно будет производиться по нормам техпроцесса 0,09 мкм, и станет первым несерверным ядром, обладающим поддержкой технологии HyperThreading (Thread-Level Parallelism), позволяющей эмулировать работу нескольких процессорных ядер на одном кристалле. Это, по оценкам специалистов, даст прирост производительности процессоров до 30%. Кроме того, предвидя возможный успех технологии x86-64, представляющей собой 64-битное расширение классической IA-32 и реализованной в семействе процессоров Hammer от AMD, специалисты Intel не исключают возможность включения в Prescott собственной аналогичной 32/64-битной микропроцессорной технологии, названной "Yamhill Technology".

Чипсеты Intel

Первым чипсетом, поддерживающим процессор Pentium 4, стал **i850**, работающий только с относительно дорогой памятью фирмы Rambus и до сих пор являющийся самым высокопроизводительным среди всех существующих решений (рис.1).

Он создан на основе серверного чипсета i840 и состоит из трех основных микросхем (хабов): Intel 82850 — Memory Controller Hub (MCH), Intel 82801BA — I/O Controller Hub (ICH2) и Intel 82802 — Firmware Hub (FWH). Аналогом северного моста, использующегося во всех классических чипсетах, является хаб MCH. В его составе имеются контроллер системной шины (Quad Pumping Bus, тактовая частота — 100 МГц), двухканальный контроллер памяти Direct Rambus (под-

Рис. 1



держка модулей памяти PC600/800 RDRAM, максимальная пропускная способность — 3,2 Гб/с, максимальный поддерживаемый объем памяти — 2 Гб, модули устанавливаются в слоты попарно), контроллер шины AGP (поддерживает режимы работы AGP 4x и Fast Write). Для связи между MCH и ICH2 используется шина Hub Link с пропускной способностью 266 Мб/с. Уже достаточно "пожилой" южный мост ICH2 (Intel 82801BA) является универсальным и используется в составе всех чипсетов Intel, начиная с i810. В ICH2 встроены контроллер PCI-шины (поддерживает до шести PCI Bus Master-устройств), двухканальный IDE-контроллер (поддерживает до четырех устройств ATA/100), двухканальный USB-контроллер (4 USB-порта), 6-канальный звуковой контроллер AC97 (с цифровыми выходами типа SP/DIF) и интерфейс локальной сети (LCI), обеспечивающий поддержку 10/100 Мбит/с Ethernet и Home PNA. Поддерживается коммуникационный интерфейс CNR (Communications Network Riser). FWH (Firmware Hub) предназначен в первую очередь для размещения системного BIOS'а, а также содержит генератор случайных чисел (Random Number Generator — RNG), который можно использовать для построения систем защиты или для других системных нужд. Хотя чипсет i850 и не поддерживает двухпроцессорные конфигурации, что значительно ограничивает применение систем на его основе в серверах, имеется возможность использования в его составе дополнительного хаба P64H для подключения 64-разрядной 66-мегагерцовой шины PCI. Одним из самых значительных недостатков i850 является сложность разводки соединений элементов чипсета на плате, что заставляет разработчиков использовать дорогой 6-слойный дизайн системных плат. Это делает, учитывая все еще высокую стоимость самого чипсета, системные платы на его основе достаточно дорогими. Летом этого года ожидается выпуск обновленной версии чипсета — **i850E**, который станет последним набором логики от Intel, использующим память PC1066/PC800 RDRAM. От своего прародителя он будет отличаться лишь поддержкой 533-мегагерцовой шины Quad Pumped и новой микросхемой южного моста ICH4, поддерживающей три канала высокоскоростной шины USB 2.0 (всего 6 портов, максимальная скорость обмена — 480 Мб/с).

Летом 2001 г. был анонсирован новый чипсет Intel, положивший конец гегемонии памяти Rambus на платформе Pentium 4. Главной отличительной особенностью чипсета **i845**, разрабатывавшегося под кодовым наименованием Brookdale, является поддержка самого дешевого на сегодня типа памяти — PC133 SDRAM. Это, по мнению руководства Intel, должно помочь процессору Pentium 4 завоевать рынок недорогих компьютерных систем. MCH нового чипсета практически аналогичен северному мосту i850, за исключением, естественно, контроллера памяти — он позволяет работать с 3 Гб PC133 SDRAM (6 банков). Наконец-то поклонники массовых чипсетов Intel вздохнули с облегчением — преодолен "заколдованный" 512-мегабайтный барьер, преследовавший их со времен i810. В качестве южного моста i845 используется микросхема ICH2 (поддерживающая протокол IDE ATA/100).

Честно говоря, чипсет i845A своим появлением вызвал больше вопросов, чем дал ответов. Во-первых, он отличался крайне низкой для современного решения производительностью, сводившей на нет и без того мизерные преимущества первых вариантов Pentium 4. Во-вторых, i845A, изначально планировавшийся как недорогое решение, призванное продвинуть процессор Pentium 4 на рынок систем начального уровня, имеет отпускную цену (впрочем, Intel всегда завышала цены на свою продукцию), значительно превышающую аналогичный показатель своих основных конкурентов — VIA P4X266 и, особенно, SiS645, которые, поддерживая гораздо более быструю память DDR, имеют гораздо более высокую производительность.

Чипсет i845A, по всей вероятности, можно занести в книгу рекордов Гиннеса как самый короткоживущий из всех массово выпускавшихся чипсетов Intel. Так, уже в конце 2001 г. начались массовые анонсы системных плат на основе его более совершенного варианта — **i845D (B-step)**. Он отличается от своего предшественника — i845 — только контроллером памяти, который теперь поддерживает PC2100/PC1600 DDR SDRAM, но, к сожалению, общий объем памяти снижен до 2 Гб (не более 4 банков). Это означает, что большинство материнских плат на базе i845D будет оснащаться только двумя разъемами DDR DIMM.

Весной этого года Intel выпускает еще два чипсета — **i845G** и **i845E**. Оба они рассчитаны на новую процессорную шину Pentium 4, имеющую тактовую частоту 133 МГц, а также на более скоростную память PC2700 DDR и усовершенствованный AGP 8x-интерфейс. Для них разработан новый южный мост ICH4, в который включена поддержка 6 портов USB 2.0, а интегрированный i845G, кроме того, наконец-то получит новое видеоядро i760, являющееся полностью самостоятельной разработкой инженеров Intel. Но самое смешное заключается в том, что практически одновременно с этими современными чипсетами, полностью свернув к началу года производство первоначального SDRAM-варианта i845, Intel снова пошла на попятную, начав производство версии **i845S** с отключенной поддержкой DDR, т.е. фактически просто удалила из i845 поддержку двух банков памяти SDRAM. Такое шараханье общепризнанного законодателя компьютерной моды свидетельствует о его явной неуверенности в правильности выбранной им стратегии развития.

Во второй половине года должен появиться новый чипсет **Granite Bay**, который будет ориентирован на использование в высокопроизводительных системах и однопроцессорных рабочих станциях. Ключевой особенностью этого чипсета будет поддержка двухканальной памяти DDR266. Использование двух 64-битных каналов обеспечит пропускную способность памяти 5,4 Гб/с, что наконец-то позволит полностью насытить данными шину Pentium 4 с 533-мегагерцовой Quad Pumped, для которой он, в принципе, и предназначен. По своей производительности Granite Bay обещает значительно превзойти i850, т.к., благодаря чередованию каналов (интерливингу), латентность подсистемы памяти у этого набора логики обещает быть крайне низкой. Кроме того, Granite Bay будет поддерживать интерфейс AGP 8x и комплектоваться южным мостом ICH4.

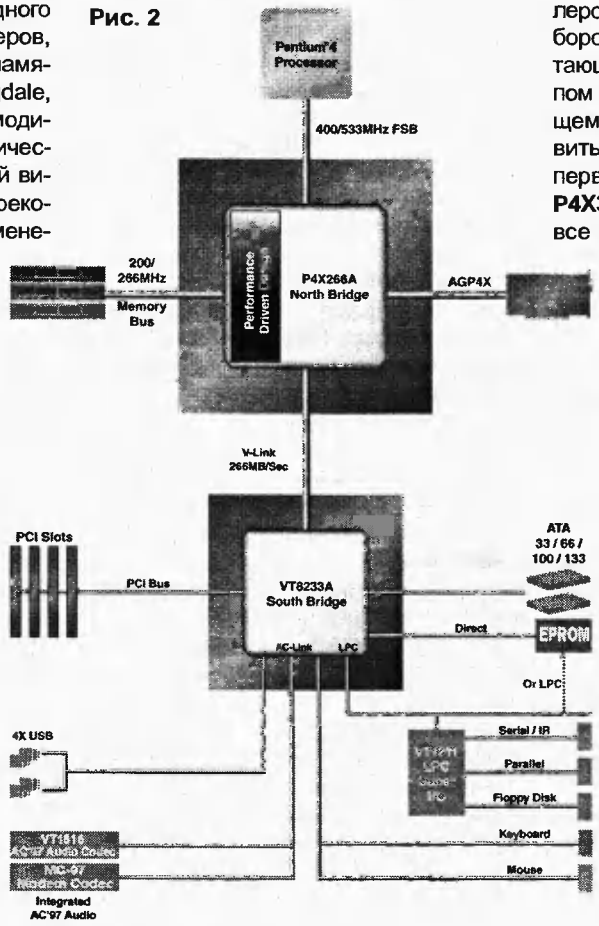
В более отдаленной перспективе Intel планирует выпустить набор логики **Springdale**, который сменит "на боевом посту" линейку чипсетов i845. Хотя ранее сообщалось, что Springdale будет готов уже в начале 2003 г., Intel приняла решение немного повременить с выходом этого чипсета, но зато добавить в него поддержку памяти DDRII+ вместо изначально планировавшейся DDRII. Благодаря этому Springdale сможет поддерживать DDR-память, работающую на частоте 533 МГц, что позволит достичь пропускной способности подсистемы памяти 4,3 Гб/с, т.е. полного соответствия по пропускной способности 533-мегагерцовой шине Quad Pumped процессоров Pentium 4. По всей видимости, Springdale не сможет работать с DDRII-модулями памяти вообще, поскольку спецификация DDRII+ определяет другой сигнальный уровень —1,8 В вместо 2,5 В. Таким образом, в чипсетном роадмапе Intel теперь вообще нет ни одного чипсета для настольных компьютеров, обладающего поддержкой DDRII памяти. Помимо обычной версии Springdale, в планах Intel стоит и выпуск его модификации с интегрированным графическим ядром — Springdale-G. По всей видимости, это графическое ядро перекочует в Springdale-G без особых изменений из i845G. Одновременно со Springdale свет увидит и новый южный мост ICH5, который придет на смену ICH4 и значительно расширит спектр его возможностей. В первую очередь хочется сказать о том, что в ICH5 появится поддержка протокола Serial-ATA, а также встроенный контроллер беспроводной сети, соответствующий стандарту 802.11. Остальные возможности ICH5, включая поддержку USB 2.0, будут заимствованы из ICH4. В качестве шины, связывающей мосты чипсета, в ICH5 будет применяться Hub Link 2 с пропускной способностью 1,06 Гб/с.

Чипсеты VIA

Из всех конкурентов интеловских чипсетов для процессора Pentium 4, раньше всего появился на свет VIA **P4X266**, несмотря на явно негативное отношение к этому самой Intel. Тем не менее, несмотря даже на то, что он был самым первым чипсетом для Pentium 4 с поддержкой DDR SDRAM, особо широкого распространения он не получил — слишком уж сильно зависят от благорасположения Intel "независимые" производители системных плат.

Чипсет P4X266 (рис.2), выполненный по хабовой архитектуре, содержит северный мост на микросхеме VT8753, который, в отличие от i845, поддерживает двухпроцессорную конфигурацию. Это, наряду с использованием отдельного чипа-компаньона VPX-64 для организации 64-разрядной шины PCI, может сыграть существенную роль при продвижении его на серверный рынок. Продолжая традиции VIA, контроллер памяти обеспечивает поддержку широкой номенклатуры типов памяти — помимо основной PC1600/2100 DDR SDRAM, возможно использование PC100/133 SDRAM и VCM SDRAM — всего до 4 Гб. Северный мост поддерживает асинхронный режим работы шины памяти и FSB. Для связи с южным мостом VT8233 используется шина V-Link с общей пропускной способностью 266 Мб/с. VT8233 содержит контроллер шины PCI, двухканальный IDE-контроллер (ATA/100), три канала USB (6 портов), 6-канальный звуковой контроллер AC97 (с цифровыми выходами типа SP/DIF) и LAN-контроллер 10/100 Мб/с Ethernet и Home PNA. Наряду с коммуникационным интерфейсом CNR, поддерживается и ACR (Advanced Communications Riser). Южный мост имеет ряд модификаций — в VT8233C встроен сетевой контроллер, разработанный широко известной на рынке коммуникаций компанией 3Com, а появившийся в начале года VT8233A включает в себя IDE-контроллер ATA/133. Следуя своей давней традиции, вслед за P4X266 VIA выпустила и P4M266 с интегрированным графическим ядром Savage 4, причем этот чипсет имеет внутреннюю шину AGP 8x с возросшей до 2,1 Гб/с пиковой скоростью передачи данных.

Рис. 2



Последнее время у компании VIA появилась не самая приятная для пользователей привычка торопиться с выпуском в свет не доведенных до полной кондиции продуктов, и лишь потом, сняв первые "сливки", выпускать обновленные чипсеты, полностью соответствующие рекламным обещаниям. Так получилось и с многострадальным P4X266, производительность которого была не более чем средней. И только лишь с выходом в конце 2001 г. его модификации — **P4X266A**, поддерживающей высокопроизводительную процессорную шину 533 МГц и оснащенной усовершенствованным контроллером памяти, этот чипсет на равных смог бороться с хитом сезона — SiS645, работающим с самым быстрым на сегодня типом DDR-памяти — PC2700. По-настоящему же конкуренцию ему сможет составить только планируемый к выпуску в первой половине года новый чипсет — **P4X333**, в котором будут присутствовать все последние инновации: и PC2700 DDR SDRAM, и видеоинтерфейс AGP 8x. Помимо этого, будет поддерживаться и системная шина 533 МГц Quad Pumped. Очень интригующе выглядит и заявленная в предварительной спецификации на чипсет поддержка двухпроцессорности, что, в случае ее реализации на практике позволит VIA сделать еще один шаг к завоеванию очень перспективного рынка недорогих рабочих станций и серверов. Никуда от него не денется и его извечный напарник — интегрированный вариант **P4M333** с новым графическим ядром Zoetrope GFX, содержащим аппаратный блок T&L и увеличенное по сравнению со старичком Savage 4 число конвейеров. Вместе с P4X333 должен появиться и южный мост — VT8235, который будет соединяться с северными мостами по ускоренной шине V-Link с пропускной способностью 533 Мб/с. Новый южный мост будет поддерживать протокол ATA/133 и интерфейс USB 2.0. Еще большую производительность должен показать следующий чипсет — **P4X600**, выпущенный по горячим следам P4X333, в котором должна появиться поддержка 128-битной шины памяти, поддерживающей пары DDR266- или DDR333-модулей. Этот

набор логики, в соответствии с типом установленного модуля, будет обеспечивать пиковую пропускную способность памяти 4,2 или 5,3 Гб/с. Таким образом, шина памяти в P4X600 будет иметь даже большую пропускную способность, чем процессорная шина Pentium 4 (напомню, ее скорость — 3,2 Гб/с при частоте FSB 100 МГц или 4,2 Гб/с при частоте FSB 133 МГц). Поддержка 533 МГц Quad Pumped шины, AGP 8x и V-Link 533 Мб/с подразумевается.

Чипсеты SiS

Компания SiS. выпустив в конце 2001 г. чипсет **SiS645**, снова вернулась в ряды производителей популярных на рынке продуктов. И действительно, демонстрируя высокую производительность, этот чипсет получился очень дешевым, чего так не хватает продуктам Intel и, в некоторой мере, VIA. Согласно официальной спецификации, SiS645 поддерживает до шести банков памяти при использовании PC2100 DDR SDRAM или обычной PC133 SDRAM, и до четырех банков памяти при работе с PC2700 DDR SDRAM. Это означает, что максимальный объем памяти, поддерживаемой чипсетом от SiS, составляет 3 или 2 Гб — в зависимости от используемой частоты (133/166 МГц) шины памяти. Поддержка ECC в SiS645 отсутствует. В этом чипсете фирма вернулась к двухчиповой конфигурации, причем связь между северным (SiS645) и южным (SiS961) мостами осуществляется по 16-битной 266 МГц-шине MuTIOL собственной разработки с пропускной способностью до 533 Мб/с. Что касается самого южного моста, то фактически он не представляет собой ничего особенного, и его характеристики близки к характеристикам аналогичных продуктов конкурентов. Он поддерживает до шести PCI Master-устройств, CNR- и ACR-слотов, 6 портов/2 канала USB, 5.1-канальный AC97-звук, сети 10/100MB Fast Ethernet и 1/10MB HomePNA 2.0.

Интегрированный вариант SiS645 — **SiS650** (рис.3) с графическим ядром SiS315 и поддержкой внешнего разъема AGP 4x не работает с памятью PC2700. Обновленные версии чипсетов SiS645 и SiS650 — соответственно, **SiS645DS** (ранее известный как **SiS646)** и интегрированный **SiS651**, включают поддержку процессорной шины 533 МГц Quad Pumped. Эти чипсеты могут комплектоваться как старым южным мостом SiS691, так и новым SiS962. Своеобразной переходной моделью между SiS645DS и SiS655 должен стать **SiS648**, в котором добавлена поддержка памяти DDR400. В очередной раз компания SiS старается "бежать впереди паровоза", т.к. стандарт PC3200 (модули на чипах DDR400) еще даже не утвержден комитетом JEDEC.

Среди более отдаленных планов SiS — выпуск во втором-третьем кварталах этого года чипсетов **SiS655** и **SiS660**. SiS655 будет поддерживать двухканальную PC3200/2700 DDR SDRAM, шину 533 МГц Quad Pumped, видеоинтерфейс AGP 8x. Он будет комплектоваться новым южным мостом SiS962 (с поддержкой USB 2.0, ATA/133 и IEEE 1394a). Его массовые поставки должны начаться в течение третьего квартала. Чипсет SiS660 — это версия чипсета SiS655 с интегрированным 256-разрядным графическим ядром SiS330.

Поскольку Intel уже, похоже, окончательно утратила интерес к Rambus, на его место собираются прийти другие компании-производители чипсетов, в частности, SiS уже занялась разработкой набора логики с поддержкой этого типа памяти. Rambus-продукт, создаваемый SiS, будет называться SiS658. Пока о нем известно лишь то, что новый набор логики будет предназначен для процессоров семейства Pentium 4 (с 400 МГц и 533 МГц Quad Pumped Bus) и будет поддерживать PC1066 RDRAM.

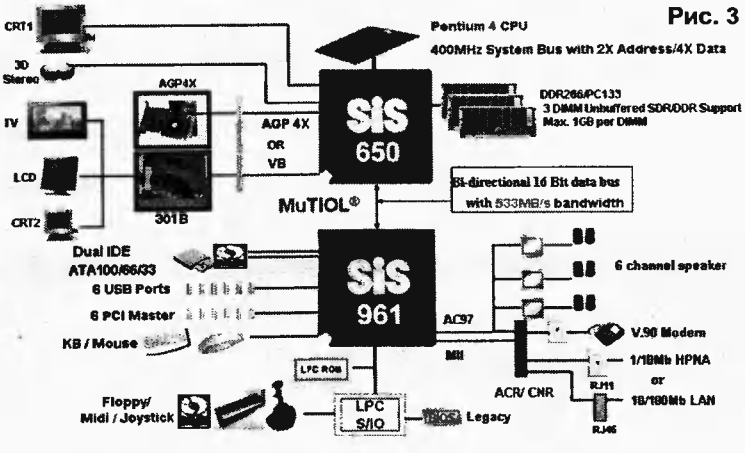


Рис. 3

Чипсеты ATI

Еще один серьезный игрок претендует на лавры лучшего интегрированного чипсета — фирма ATI с ее давно обещаемым (и ожидаемым) **A4** (или RS200) с интегрированным графическим ядром, массовое производство которого планируется начать в третьем квартале этого года. Вот последние данные по характеристикам A4: техпроцесс — 0,13 мкм, графическое ядро — RV2xx, поддержка памяти DDR333. Судя по всему, этот чипсет будет использовать южный мост SB200 от самой ATI, который будет включать USB 2.0 и сетевые контроллеры 10/100 Ethernet и 802.11b (беспроводной).

"Разбор полетов"

Ну что, не закружилась еще голова от всех этих "географических" мостов, шин, типов памяти, протоколов и интерфейсов? У меня лично — все сливается в сплошную однообразную серую массу. Но ведь так оно, в принципе, и есть — на сегодняшний день, если брать реально имеющиеся в магазинах платы, а не красивые бумажные анонсы, все чипсеты для Pentium 4 можно разделить на три группы:

- в первой группе самых производительных систем в гордом одиночестве скучает "старичок" i850, который, несмотря на дороговизну, возраст и ряд недостатков, по-прежнему остается вне конкуренции в тех случаях, когда нужна максимальная производительность любой ценой. И такое положение дел будет сохраняться вплоть до начала широкого распространения DDR-чипсетов нового поколения, оснащенных спаренным контроллером памяти;
- во второй группе "мастеров на все руки" располагаются все сегодняшние DDR-чипсеты. Вот они-то и создают ощущение однородной массы, т.к. различия между ними как в плане производительности, так и "навороченности", конечно, есть, но они минимальны. Да, давно уже научились на Тайване разрабатывать и производить изделия, не уступающие общепризнанному гранду — Intel — по своим функциональным возможностям, а теперь дышат ей в затылок в плане скорости и стабильности работы. И все это за значительно меньшие деньги! Вот и приходится мировому технологическому лидеру отбиваться от наседающих "азиатов" всеми честными и не очень способами, включая юридическое крючкотворство;
- в третьей группе "аутсайдеров", также в одиночестве, но отнюдь не гордом, сиротливо пригорюнился "гадкий утенок" i845, которому никогда не стать "гордым лебедем". Совсем непонятно, для чего (и для кого) Intel его делала? Нет ни одного параметра, по которому он не уступал бы своим тайваньским конкурентам.

Литература

1. В.Куц. Процессоры AMD Athlon и чипсеты для них. — Радиомир. Ваш компьютер, 2002, №4, С.7.

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

Работа в системе Mathematica 3.0/4.0.
Обработка данных

В предыдущей статье [1] мы рассмотрели возможности импорта и экспорта данных в системе Mathematica. После этого наступает очередь обработки полученных данных. Сочетание графических, численных и аналитических методов в рамках одной программы качественно изменяет само понятие обработки данных.

Часто необходимо прежде всего получить наглядное визуальное представление о результатах численных расчетов или эксперимента, и здесь пользователю помогут команды **ListPlot** и **ListPlot3D**. Первая из этих команд позволяет построить график по набору точек, заданных в виде $\{\{x_1, y_1\}, \ldots, \{x_n, y_n\}\}$. Если же x_i=1, 2, ..., то набор данных можно давать и в сокращенном формате $\{y_1, \ldots, y_n\}$ (рис.1). Команда **ListPlot3D** позволяет построить трехмерный график по заданному набору точек.

Дальнейшие ваши действия будут зависеть от решаемой задачи. В тех случаях, когда необходимо провести кривую через заданные точки, Mathematica предлагает команду полиномиальной интерполяции **InterpolatingPolynomial**. В качестве примера рассмотрим интерполяцию функции квадратного корня. Для этого зададим набор данных: tb = Table[{i^2, i},{i, 1, 10}], а затем к этим точкам применим команду **InterpolatingPolynomial** (рис.2). Интересно, что с ростом степени полинома качество интерполяции может упасть за счет резких колебаний многочлена между узлами, поэтому на практике широко используется кусочно-непрерывная полиномиальная интерполяция. В этом случае Mathematica не дает никакой подробной информации об используемых полиномах, ограничиваясь только сообщением **InterpolatingFunction** (рис.3). Нужно заметить, что объекты такого типа возникают при решении численных дифференциальных уравнений (в результате выполнения команды **NDSolve**).

Однако на практике часто встречаются задачи обработки экспериментальных данных, которые, естественно, обладают некоторой погрешностью. В этой ситуации нет необходимости проводить кривую через экспериментальные точки. Здесь необходимо найти некоторую "среднюю" кривую. Для этих целей предлагаются команды **Fit** и **NonlinearFit**. Первая из них (Fit) использует метод наименьших квадратов в его простейшем варианте — осуществляется поиск линейной комбинации выбранных функций для минимизации ошибки.

Проиллюстрируем это на примере. Для набора данных dt = Table[{i, i/2 + 2.0*Exp[-(i/2)]}, {i, 1, 10}], и набора функций {x, Exp[-(x/2)]} после применения команды получим, естественно, ответ 0.5 x + 2.0*Exp

Рис. 1

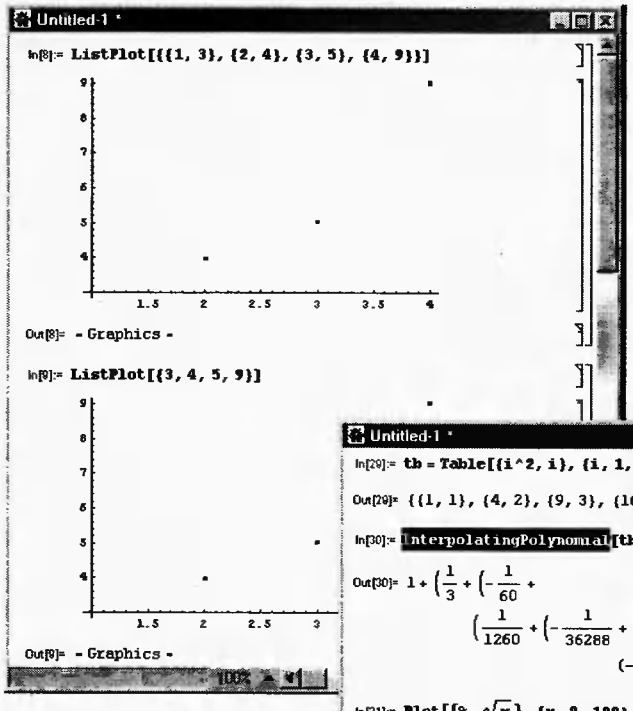


Рис. 2

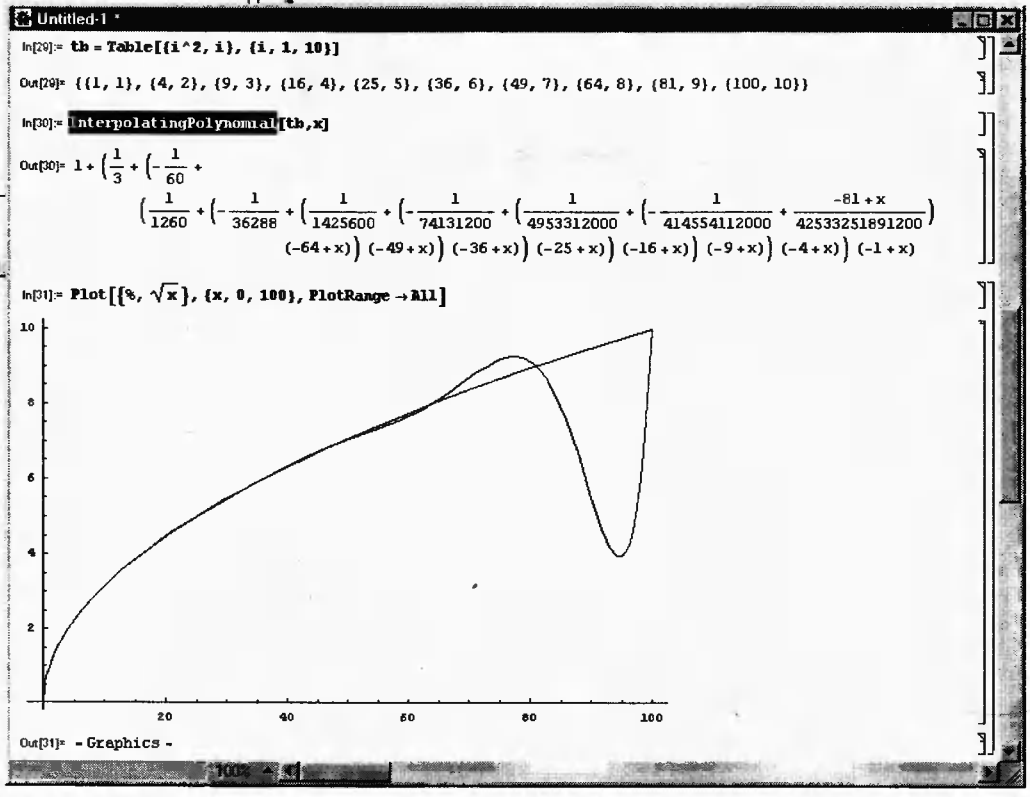


Рис. 3

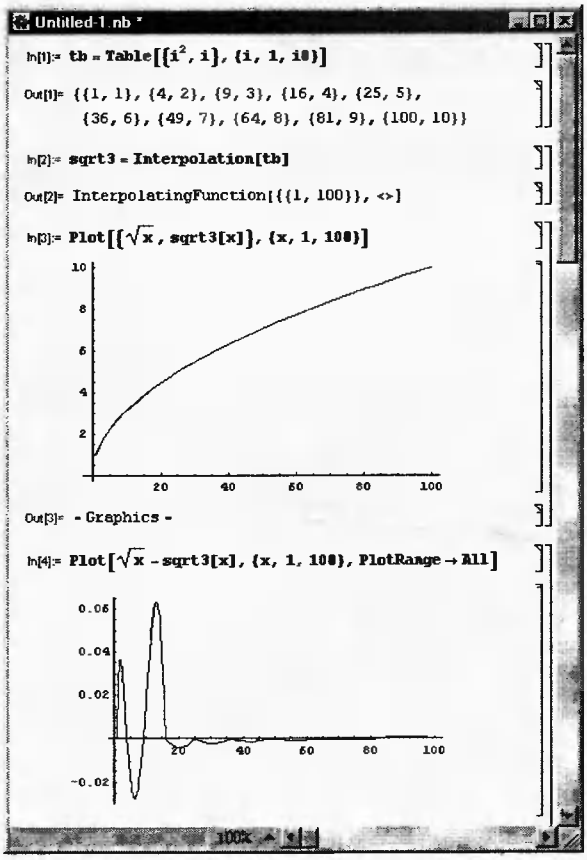
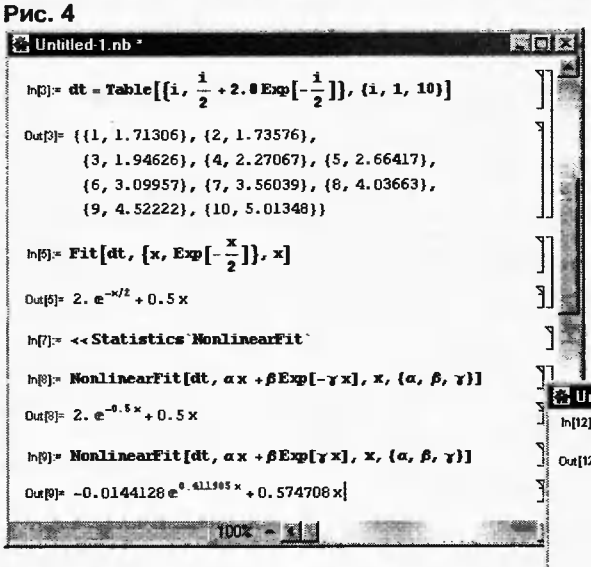


Рис. 4



[-0.5*x] (рис.4). Но далеко не всегда набор функций точно известен. Например, если по набору точек на графике еще можно догадаться, что речь идет о сумме линейной функции и экспоненты, то "на глазок" определить показатель экспоненты все-таки затруднительно. И тогда лучше всего использовать команду **NonlinearFit**, в которой указывается набор данных (dt), модель ($\alpha x + \beta \exp[-\gamma x]$), переменная (x) и набор подбираемых параметров ($\{\alpha, \beta, \gamma\}$) (рис.4). Сама команда содержится в пакете **Statistics`NonlinearFit`**. В результате опять восстанавливается исходная функция. Однако расчеты в этой ситуации значительно более сложные, и требуют большей аккуратности. Так, всего лишь изменение знака в экспоненте приводит к менее удачной с технической точки зрения модели, и, как следствие, к меньшей точности аппроксимации (рис.4). Для контроля можно воспользоваться командой **NonlinearRegress** и просмотреть отчет о результатах (рис.5).

Для обработки сигналов чаще применяется преобразование Фурье. Для дискретного преобразования Фурье набора данных служит команда **Fourier**, для обратного — **InverseFourier**. В качестве небольшого примера построим простейшую модель фильтрации данных. Во-первых, к синусоидальному сигналу при помощи команды **Random[ ]** (генерация псевдослучайных чисел с равномерным законом распределения на отрезке от 0 до 1) добавим "шум":

data = Table[Sin[Pi*i/32] + 0.3*(Random[] - 1/2), {i, 1, 64}].

После дискретного преобразования Фурье — fdata = Fourier[data], можно, с применением фильтра с заданной частотной характеристикой — filtr[x_] = 0.01/((x-2)^2+0.01)+0.01/((x-64)^2+0.01), произвести саму фильтрацию — fdata1 = Table[fdata[[i]]*filtr[i], {i, 1,64}], т.е. производится поэлементное умножение амплитуды гармоник.

Затем можно проделать обратное преобразование обработанного сигнала — InverseFourier[fdata1].

Гораздо эффектнее выглядит графическое представление результатов обработки (рис.6):

grs1 = ListPlot[data, PlotStyle → Hue[0.5], PlotJoined → True, PlotRange → All]

grs2 = ListPlot[Re[InverseFourier[fdata1]], PlotStyle → Hue[0.2], PlotJoined → True]

Show[grs1, grs2].

Кроме дискретного преобразования Фурье, Mathematica справляется как с интегральным преобразованием, так и с разложением в ряд Фурье. В первом случае необходимо предварительно загрузить (в Mathematica 3.0) дополнительный пакет << Calculus`FourierTransform`, а сами команды вряд ли нуждаются в особых комментариях (рис.7).

А вот что касается разложения в ряд, то здесь имеется некоторое различие между Mathematica 3.0 и Mathematica 4.0. Причем разница вовсе не в пользу более поздней версии системы. Так, в Mathematica 3.0 есть возможность указать отрезок, на котором будет производиться разложение, а Mathematica 4.0 работает только на отрезке (-1/2, 1/2).

Помимо всего прочего, нельзя обойти вниманием и возможности системы в области статистической обработки данных. В наборе предлагаемых функций содержатся как операции описательной статистики, так и сглаживание данных и проверка гипотез. Даже использованная выше команда нелиней-

Рис. 5

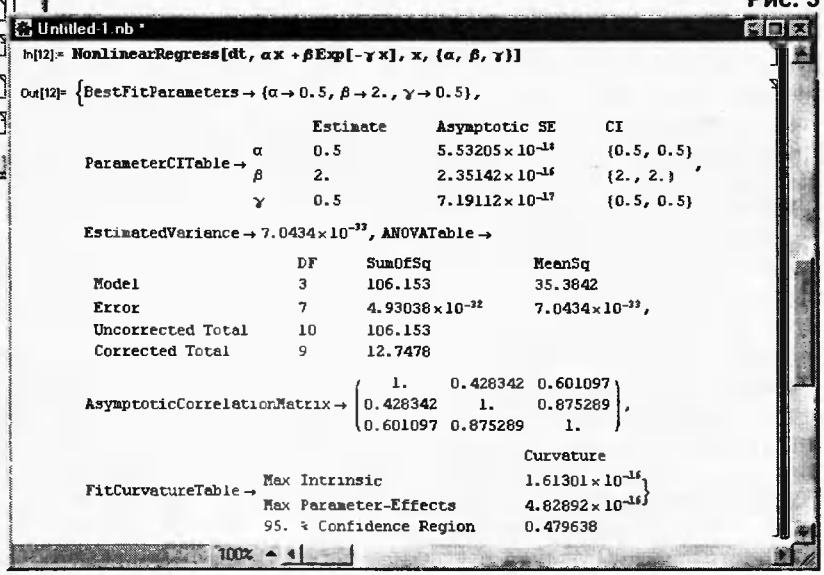


Рис. 6

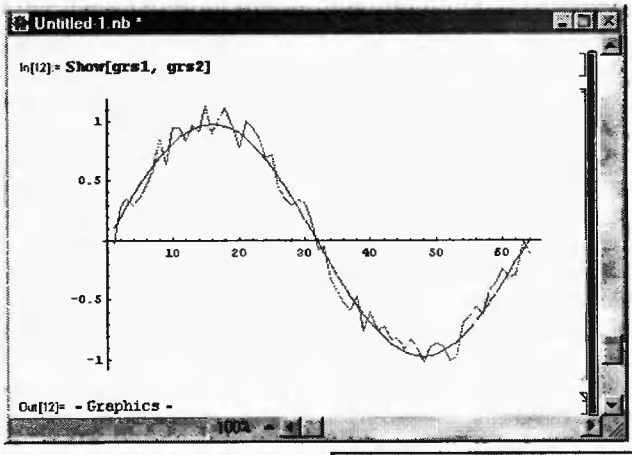
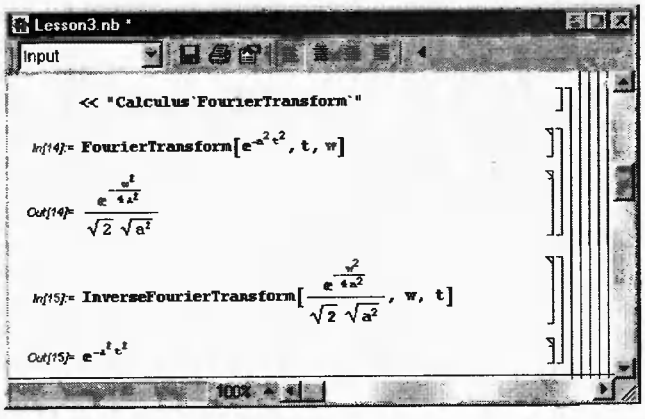


Рис. 7



ной регрессии также относится к этой группе. К тому же, этот раздел системы относится к числу наиболее интенсивно разрабатываемых в последнее время. На пути от версии 3.0 к 4.0 значительно вырос сам набор операций, а в последней на данный момент версии (4.1) численные статистические алгоритмы были значительно оптимизированы, и скорость работы увеличилась в несколько раз. Например, в Mathematica 3.0 для построения гистограммы по набору данных с именем lia необходимо вначале разбить интервал, на котором разбросаны данные, на отрезки и подсчитать количество точек на каждом из отрезков — freq = **BinCounts[**lia, **{**-4, 4, 0.4**}]**. А затем по подсчитанным частотам строится гистограмма **BarChart[**freq**]**, но на горизонтальной оси откладывается номер интервала. Mathematica 4.0 предлагает значительно более удобную команду **Histogram[**lia**]**, не требующую промежуточных вычислений.

На этом мы завершаем краткое знакомство со средствами обработки данных в Mathematica. Вполне вероятно, что с выходом новой версии системы эти средства будут значительно изменены и расширены. Но в следующих статьях мы будем иметь дело с более устоявшимся и разработанным разделом — программированием. Такой "застой", как мы увидим, обусловлен не забывчивостью разработчиков, а изначально удачным выбором языка программирования и системы работы с символьными выражениями.

Действие	Реализация
Преобразование Фурье и фильтрация данных	
Вызов пакета для работы с преобразованиями Фурье	<< Calculus`FourierTransform`
Прямое преобразование Фурье	**FourierTransform[**Exp[-a^2*t^2], t, w**]**
Обратное преобразование Фурье	**InverseFourierTransform[**%, w, t**]**
Разложение в ряд Фурье в Mathematica 3.0	**FourierTrigSeries[**x/Pi, **{**x, 0, Pi**}**, 3**]** **FourierTrigSeries[**функция, **{**переменная, x_нач, x_кон**}**, порядок_разложения**]** — возвращает разложение в ряд Фурье на отрезке {x_нач, x_кон} с точностью до указанного порядка
Разложение в ряд Фурье в Mathematica 4.0	**FourierTrigSeries[**x/Pi, x, 3**]** **FourierTrigSeries[**функция, переменная, порядок_разложения**]** — возвращает разложение в ряд Фурье на отрезке {-1/2, 1/2} с точностью до указанного порядка
Генерация случайных чисел	
Вызов дополнительных пакетов для работы с основными статистическими операциями и визуализации данных	<<Statistics`ContinuousDistributions` — работа с непрерывными распределениями (включая генерацию случайных чисел); <<Statistics`DataManipulation` — сортировка данных и описательная статистика; <<Graphics`Graphics`
Генерирование случайных чисел с заданным законом распределения	lia = **RandomArray[NormelDistribution[**0, 1**]**, 1000**]** — генерируется массив из 1000 псевдослучайных чисел с нормальным законом распределения (со средним значением 0 и стандартным отклонением 1)
Основные статистические операции. Гистограмма	
Минимальное значение данных в списке	**Min[**lia**]**
Максимальное значение данных в списке	**Max[**lia**]**
Вычисление среднего значения	**Mean[**lia**]**
Вычисление медианы	**Medlan[**lia**]**
Вычисление вариации	**Variance[**lia**]**
Вычисление стандартного отклонения	**StandardDevlation[**lia**]**
Получение списка основных величин, характеризующих распределение данных	**DispersionReport[**lia**]**
Сортировка данных	**Sort[**lia**]**
Построение гистограмм	1. freq = **BinCounts[**lia, **{**-4, 4, 0.4**}]** — разбиение интервала [-4, 4] на подынтервалы шириной 0.4 и подсчет числа элементов, находящихся в каждом из подынтервалов. 2. **BarChart[**freq, **PlotRange** → **{{**0, 30**}, {**0, 200**}}]** — построение гистограмм по данным подсчета. В Mathematica 4.0 имеются дополнительные команды для построения гистограмм: 1. **HIstogram[**lia**]** — построение гистограмм непосредственно по исходным данным; 2. **Histogram[**freq, **FrequencyData** → True**]** — построение гистограмм по данным подсчета

Литература

1. А.Гринчук. Работа в системе Mathematica 3.0/4.0. Работа с внешними файлами. — Радиомир. Ваш компьютер, 2002, №4, С.11.

Кроссворды и компьютеры

С.РЮМИК,
г.Чернигов.

"Догадка не хуже разума".
Народная мудрость

Слово "кроссворд" произошло от слияния двух английских терминов: "cross" — пересечение и "word" — слово. Решение кроссвордов тренирует память, расширяет кругозор, развивает сообразительность. Кроссворд — прекрасное средство, чтобы скоротать время в поездах, самолетах, залах ожидания, на скучных лекциях и длинных собраниях. Некоторые типы кроссвордов используются в профотборе и даже при приеме на работу.

Кроссворды появились сравнительно недавно, в начале XX века [1, 2]. Официальная дата рождения — 21 декабря 1913 г., когда Артур Уинн опубликовал в воскресном приложении "Fun" американской газеты "New York World" кроссворд, состоящий из 31 слова. У нас первый кроссворд появился 18 августа 1925 г. в ленинградской "Новой вечерней газете". Правда, в то время он назывался иначе — "Переплетные слова/задачи крест-накрест".

Немного статистики. Самый большой в мире кроссворд опубликовал Роберт Тюрко из Квебека — 82951 клетка и 25614 слов. Бельгиец Роже Боукарт составил кроссворд длиной 31 м и шириной 53 см (50400 слов). Андриан Белл за 50 лет прислал в газету "Таймс" 4520 кроссвордов. Уникальной "производительности труда" добился Роджер Ф.Сквайерс — 35 публикуемых кроссвордов каждую неделю.

Глядя на эти фантастические цифры, логично предположить, что человеку в составлении кроссвордов должен помогать компьютер. Действительно, процедуры поиска и расстановки слов, рисования шаблонов сеток, проверки грамматики можно поручить вычислительной системе, максимально облегчив себе труд. За человеком остается главное — выбор темы и составление вопросов. Одна незадача — профессиональные кроссвордисты, как правило, гуманитарии, далекие от программирования. Программисты, наоборот, имеют довольно туманное представление о кроссвордах как о виде искусства.

Объединить "физиков" и "лириков" должна их совместная работа по составлению компьютерных программ для кроссвордистов. Различают три направления разработок:

- программы для решения кроссвордов;
- программы для отбора и поиска слов;
- программы для составления кроссвордов.

В количественном отношении программ третьего направления известно примерно в 4 раза больше, чем программ первого и второго направлений. К сожалению, русскоязычных творений среди них не так уж и много, а если учесть качество программ, то их и вовсе можно пересчитать по пальцам одной руки.

В Интернете проблемам кроссвордов посвящено около 300 сайтов, адреса которых приведены в [3]. Хорошо налажена система рассылок свежей информации, издаются электронные журналы, накапливаются банки данных словарей и текстов кроссвордов различной тематики.

Прежде чем начать обзор программного обеспечения, необходимо уяснить базовые понятия.

Теория кроссворда

Кроссворды классифицируют по содержанию и оформлению (см.таблицу) [4]. Для справки, так называемые "японские кроссворды" кроссвордами как таковыми не являются. Они относятся к разряду головоломок-рисунков и имеют исходное название — "нонограммы".

Традиционные кроссворды, которые печатаются в различных периодических изданиях, как правило, относятся к разряду классических тематических или классических полиглотов. Классический кроссворд содержит рисунок сетки с двух- или четырехсторонней симметрией. Каждое слово имеет не менее двух пересечений с остальными. Количество черных (неинформативных) клеток минимально. Тематические кроссворды, в отличие от полиглотов, посвящаются определенной теме. Составлять их трудно, а решать их интересно и приятно.

Лингвистическое содержание кроссвордов многонационально — сколько языков народов мира, столько и разновидностей кроссвордов. Особенности языка, на котором выполнен кроссворд, вносят существенные коррективы во внешнее оформление рисунка. Например, англоязычные кроссворды имеют чрезвычайно густую по заполнению сетку, чем их составители очень гордятся. Словарная база достигает 80 тысяч терминов с пиком на четырех- пяти- и шестибуквенных словах. В американских кроссвордах по традиции допускается вводить двойные слова без пробела между ними.

Русский язык достаточно "неудобен" для составления кроссвордов — активный лексикон имен существительных составляет 30 тысяч терминов с пиком на шести-, семи- и восьмибуквенных словах. В связи с этим русскоязычные кроссворды обычно имеют "рыхлую" структуру с неплотной сеткой.

Классификация	
По содержанию	**По оформлению**
Тематический	Классический
Полиглот	Белый
Алфавитный	С перегородками
С нестандартными определениями (юмор, с фрагментами, ребусный, крисс-кросс, рассыпной)	Сканворд
	Координатный
	Кейворд
	Русский
	Скользящий
С нестандартным заполнением (слого-кроссворд, двухбуквенный, символьный, реверсивный, дуаль)	Словарный
	Линейный
	Филворд
	Фигурный
	Объемный

Для количественных измерений используется коэффициент густоты заполнения сетки. Это так называемая плотность кроссворда, которая измеряется в процентах и определяется отношением числа белых (заполненных словами) клеток к их общему количеству на рисунке. В частности, "белый" кроссворд, упоминаемый в таблице, имеет плотность 100%. Средней для русскоязычных кроссвордов считается плотность 60...80%.

Низкая плотность свидетельствует о недостаточной проработанности идеи. Такие кроссворды трудно разгадываются, поскольку имеют мало пересечений слов, а следовательно, и подсказок. Другая крайность — очень высокая плотность кроссворда. При этом трудно избежать употребления редких и малоизвестных слов, многие из которых заведомо "не берутся". Единственная надежда на то, что подобные слова в дальнейшем автоматически разгадаются за счет заполнения их клеток буквами других слов.

В разных издательствах предъявляются свои требования к составлению кроссвордов. Жесткого канона не существует, но рекомендуется для классических кроссвордов соблюдать следующие правила:

- использовать только нарицательные и собственные существительные в именительном падеже единственного числа (кроме слов, имеющих только множественное число). Нельзя употреблять слова, пишущиеся через тире или имеющие уменьшительно-ласкательную форму;
- в каждую клетку кроссворда допускается вписывать только одну букву. Нумерацию слов на рисунке следует производить по возрастанию номеров слева направо и сверху вниз;
- каждое слово должно иметь не менее двух пересечений с другими словами;
- количество черных клеток внутри кроссворда, расположенных подряд по вертикали или горизонтали, не должно превышать четырех;
- желательна четырехсторонняя симметричность рисунка и квадратные клетки в сетке;
- вопросы, рассчитанные на массового читателя, должны быть политически, национально, медицински и религиозно нейтральными.

Программы для решения кроссвордов

Прежде чем приступить к созданию собственного электронного кроссворда, хорошо бы посмотреть, как это делают другие. Известно несколько "программ-решалок", которые по своей сути являются развлекательными и относятся к жанру игровых. Пользователю предлагается ввести в компьютер файл со специальным расширением (обычно .crw или .crs), в котором содержится вся информация о кроссворде. На экране монитора появляется незаполненная сетка и перечень вопросов для слов по горизонтали и вертикали. Ввод ответов на вопросы производится с клавиатуры. Программа автоматически проверяет заполнение сетки и выносит свой вердикт о правильности решения. В сложных случаях активизируется система подсказок.

Некоторые программы, например, "Универсальный генератор кроссвордов" (автор — А.Пискунов http://ns1.uni-vologda.ac.ru/students/paa/CROSSGEN, файл — "cross.zip", 599 Кб), могут самостоятельно генерировать кроссворды разной сложности, исходя из имеющегося словарного багажа. Файлы с уже готовыми кроссвордами можно скачать через Интернет. Именно такая функция предусмотрена в программе "Классический Кроссворд v.1.4" (авторы — В.Синицын, О.Синицына, http://sowsoft.chat.ru, файл — "crosswrd.zip", 253 Кб) (рис.1). Авторы программы разместили на своем сайте большую подборку кроссвордов на все вкусы и предлагают бесплатно "увековечить" в Интернете творение любого желающего.

К удовольствию посетителей сайтов, обе перечисленные программы распространяются свободно, по принципу "как есть". Программы для решения кроссвордов позволяют заменить бумажные рисунки их электронными копиями на экране монитора. Вместо карандаша и ластика здесь используется клавиатура и мышь. К дополнительным (но

Рис. 1

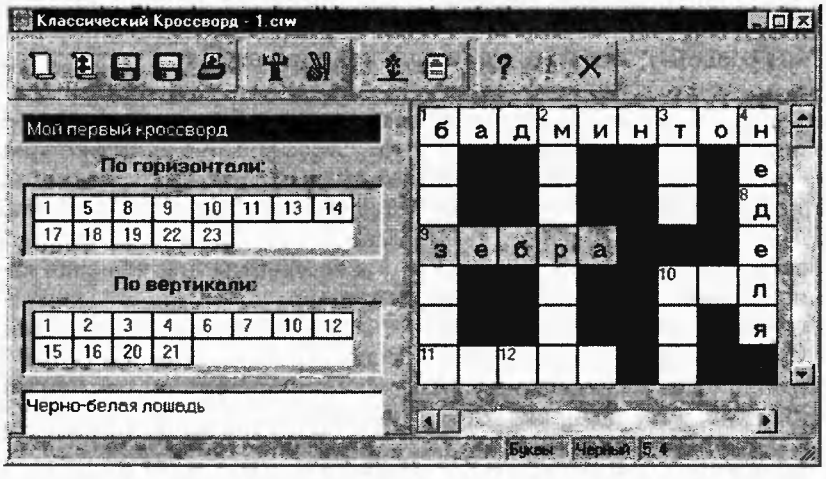
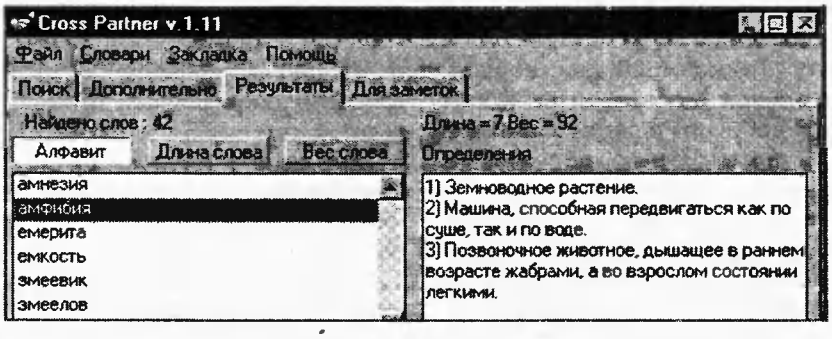


Рис. 2



не основным) функциям относится возможность составления собственных кроссвордов, переноса рисунков в формат .bmp, а текстов — в .txt.

Программы для подбора слов

Часто возникает ситуация, когда кроссворд почти полностью заполнен словами, но оставшееся "почти" никак не удается подобрать. На этот случай разработаны специальные программы поиска и сортировки слов, обладающие профессиональным словарным запасом. Снимок картинки экрана одной из таких программ — "CrossPartner v.1.11" (авторы — С.Плотников, И.Шмелев, http://crosspartner.chat.ru, файл — "CrossPart.zip", 767 Кб) приведен на рис.2.

Поиск производится по маске, длине и даже "весу" слова. Для справки, "вес" слова измеряется в безразмерных единицах и представляет собой арифметическую сумму алфавитных порядковых номеров всех букв, входящих в слово. Таким образом, буква "Я" ровно в 33 раза "тяжелее", чем буква "А", "вес" которой принят за единицу.

Словари к программе "CrossPartner v.1.11" скачиваются отдельно. Они составлялись большой интернациональной командой энтузиастов. Для большинства слов приводится один или несколько вариантов вопросов (определений). Читать словари можно вместо энциклопедии — познавательно, лаконично и интересно.

Еще одна программа, помогающая в поиске слов — это "Paseek v.2.6" (автор — С.Пронякин, http://paseek.chat.ru/download.html, файл — "paseek26.exe", 975 Кб) (рис.3).

Программа имеет встроенный словарь на 54475 слов, кроме того, она содержит четыре словарные игры: "Цепочка слов" (переход от одного слова к другому заменой буквы), "Палиндромы" (поиск слов, читающихся одинаково слева направо и справа налево), "Разложение на слова" (из одного слова несколько слов), "Анаграммы" (из букв слова новые слова). Программа "Paseek v.2.6" многофункциональна, она может использоваться в качестве обучающей на уроках русского языка. Эта программа, как и предыдущая, поставляется Freeware, то есть бесплатно.

Программы для составления кроссвордов

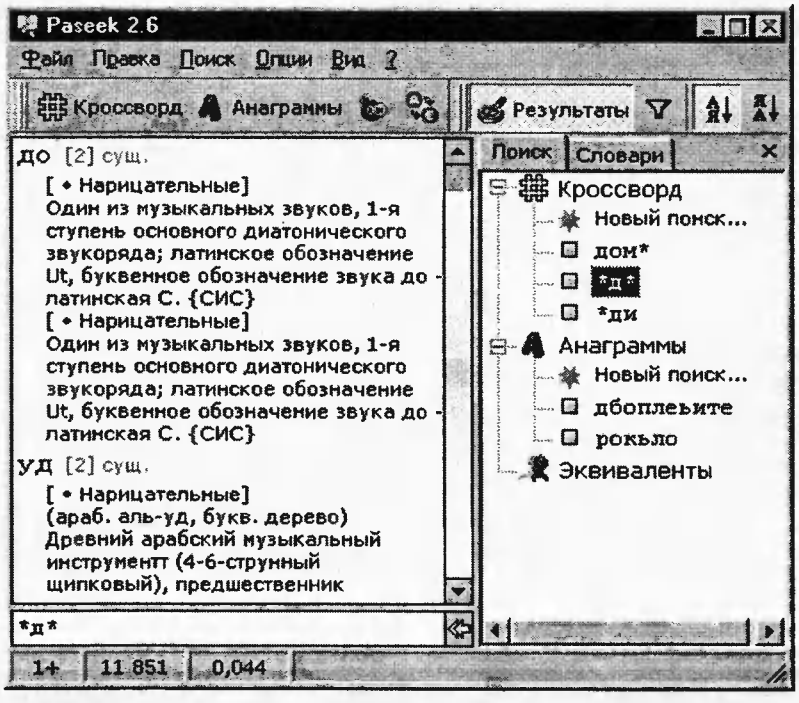
Пофантазируем, каким хотелось бы видеть диалог работы с программой составления кроссвордов. В идеале — ввести тему кроссворда, количество слов и нажать клавишу <ENTER>. Но так не бывает. Чтобы кроссворд был интересным, оригинальным, самобытным, необходимо предварительно составить хорошо продуманный тематический словарь вопросов-ответов. Каждому термину (ответу) в таком словаре должны соответствовать одна или несколько дефиниций (вариантов вопросов). Например, в [3] приводится 26 различных определений слова "яблоко", и это далеко не предел.

После составления словаря терминов начинается рутинная работа по выбору рисунка сетки, размера кроссворда, заполнения ячеек словами. Именно с такими задачами хорошо справляются компьютерные программы. Рассмотрим некоторые из них.

Из русскоязычных программ наиболее мощная — это "Кроссмейкер" Л.Федорчука (http://fedcross.xelon.ru). Занимаемый объем — 270 Мб, в составе которого 24 разноплановые программы, поддерживающие практически все разновидности кроссвордов. Физически "Кроссмейкер" встроен в Microsoft Word97 и Microsoft Excel97, содержит несколько специализированных и универсальных словарей с объемом от 15 до 60 тысяч слов. Единственный недостаток программного комплекса — он не тиражируется и не продается...

Следующей по насыщенности функций является программа "CrossWord v.2.0" (авторы — В.Танчук, А.Сасько). На сайте http://users.i.com.ua/~cross размещается ее бесплатная демо-версия (файл "cross2.zip", 1957 Кб). Программа имеет встроенный редактор "Vocedit v.1.5" и словарную базу объемом 47 тысяч терминов. Ограничения в демо-версии накладываются на процесс печати, сохранения рисунка кроссворда и импорт шаблонов сеток. К счастью, эти ограничения не являются фатальными. Естественно, пользователю придется больше потрудиться, но: "Любишь кататься, люби и саночки возить".

Рис. 3



(Окончание следует)

ЛИСТАЯ СТРАНИЦЫ

Тем, кто собирается использовать процессор Pentium 4 в своем компьютере, пытается помочь *редакция журнала* ***"Hard'n'Soft"*** **(№12/2001 С.46...61)** статьей "Двадцать четыре для Pentium 4", рассказывая **о результатах тестирования материнских плат на основе чипсетов Intel i845, i850 и VIA Apollo P4X266.**

Выпустив процессор Pentium 4, Intel создала ряд проблем как для пользователей, так и для производителей системных плат, отмечается в статье. Мало того, что первый чипсет для Pentium 4 — i850 — работает только с памятью RDRAM, так еще и процессорное гнездо Socket 423 требует замены матплаты при переходе с Pentium III или Celeron. А самые новые (и быстрые) процессоры выпускаются в новом формате Socket 478. Вместе с Pentium 4, кроме упомянутого i850, можно использовать недорогой, но весьма эффективный i845, работающий с памятью SDRAM PC133 (к началу следующего года должны появиться платы на основе DDR-версии этого чипсета), а также чипсет VIA Apollo P4X266, вокруг которого, однако, идет несколько судебных процессов между Intel и VIA. Все это следует учитывать при выборе системной платы для Pentium 4.

Тестировались платы BL7 и TH7-RAID ф.ABIT, Acorp 4S845A, AOpen AX4BS Pro, ASUS P4B, Canyon CN-8G2AS, Chaintech CT-9BJA, NB72-SR, NT70-SA и WT70-EC ф.DFI, EPoX EP-4B2AE, PIC VC11, GA-8IDXH и GA-8TX ф.Gigabyte, D850GB и D850MD ф.Intel, Lucky Star P4A845S, 845 Pro2, 850 Pro2 и 850 Pro5 ф.MSI, Shuttle AV40, SL-85DRV и SL-85SD+ ф.Soltek, SuperGrace 845A3, построенные на основе одного из упомянутых чипсетов. Их основные технические характекристики приведены в табл.1 и 2.

Использовалась память RDRAM (чипсет i850), SDRAM PC133 (чипсет i845) и DDR266 SDRAM (чипсет Apollo P4X266) объемом по 256 Мб. Процессоров было тоже три — два с тактовой частотой 1,7 ГГц для Socket 423 и Socket 478 и один 2 ГГц под Socket 478. Остальное оборудование было следующим:

- видеоадаптер ABIT Siluro GF3 (на основе чипа NVIDIA GeForce3 с 64 Мб видеопамяти);
- жесткий диск Samsung SV1021H (10 Гб, Ultra ATA/100);
- дисковод CD-ROM Creative CD-4820E (48X);
- операционная система Windows 98SE.

Тестирование проводилось с помощью пакетов ZD Winstone 2001, ZD Content Creation 2001 и ZD WinBench 99 версии 1.1. Первые два пакета измеряют интегральную производительность системы при работе с совре-

Табл. 1

Материнская плата		BL7	TH7-RAID	4S845A	AX4BS Pro	P4B	CN-8G2AS	CT-9BJA	NB72-SR	NT70-SA	WT70-EC	EP-4B2-AE	VC11
Производитель		ABIT Computer		Acorp Electronics	AOpen	ASUSTeK Computer	Canyon	Chaintech Computer	DFI			EPoX Computer	First International Computer
Форм-фактор		ATX	ATX	ATX	ATX	ATX	ATX	ATX	ATX	ATX	ATX	ATX	ATX
Чипсет		i845	i850	i845	i845	i845	i850	i845	i845	i850	i850	i845	i845
Разъем процессора		Socket 478	Socket 423	Socket 478	Socket 478	Socket 478	Socket 423	Socket 478	Socket 478	Socket 478	Socket 423	Socket 423	Socket 478
Интегрированная аудиоподсистема		AC'97	AC'97	AC'97	AC'97	AC'97	AC'97	C-Media 8738/AC'97	AC'97	AC'97	AC'97	AC'97	AC'97
Количество слотов PCI		6	5	5	5	6	5	5	5	5	5	6	5
Слот CNR		1	1	1	1	1	1	1	1	1	1	1	1
AGP		4x	4x	4x	4x	4x	4x	4x	4x	4x	4x	4x	4x
IDE RAID		-	HPT370	-	-	-	-	-	PDC20265R	PDC20265R	-	-	-
Количество разъемов DIMM	SDRAM	3	-	3	3	3	-	3	3	-	-	3	3
	DDR	-	-	-	-	-	-	-	-	-	-	-	-
	RDRAM	-	4	-	-	-	4	-	-	4	4	-	-
Максимальный объем поддерживаемой памяти, Гб		3	2	1,5	3	3	2	3	3	2	2	3	3
Тип поддерживаемой памяти		PC133	PC600, PC800	PC100, PC133	PC133	PC100, PC133	PC600, PC800	PC133	PC133	PC600, PC800	PC600, PC800	PC133	PC133
Средняя розничная цена, USD		140	165	100	150	170	120	130	130	165	130	130	130

Табл. 2

Материнская плата		GA-8IDXH	GA-8TX	D850GB	D850MD	P4A845S	845 Pro2	850 Pro2	850 Pro5	AV40	SL-85DRV	SL-85SD+	845A3
Производитель		Gigabyte Technology		Intel		Lucky Star Technologies	Micro-Star			Shuttle	Soltek		SuperGrace
Форм-фактор		ATX	ATX	ATX	Micro ATX	ATX	ATX	ATX	ATX	ATX	ATX	ATX	ATX
Чипсет		i845	i850	i850	i850	i845	i845	i850	i850	VIA Apollo P4X266		i845	i845
Разъем процессора		Socket 478	Socket 423	Socket 423	Socket 478	Socket 478	Socket 478	Socket 423	Socket 478	Socket 478	Socket 478	Socket 478	Socket 478
Интегрированная аудиоподсистема		Creative CT5880	Creative CT5880	AC'97	AC'97	AC'97	C-Media 8738	AC'97	C-Media 8738	AC'97	AC'97	Creative CT5880	AC'97
Количество слотов PCI		6	5	5	3	5	6	4	4	5	6	6	5
Слот CNR		1	1	-	-	1	1	1	1	-	1	1	1
AGP		4x	Pro 4x	4x	4x	4x	4x	4x	4x	4x	4x	4x	4x
IDE RAID		-	-	-	-	-	-	-	-	-	-	PDC20265R	-
Количество разъемов DIMM	SDRAM	3	-	-	-	3	3	-	-	-	-	3	3
	DDR	-	-	-	-	-	-	-	-	3	-	-	-
	RDRAM	-	4	4	4	-	-	4	4	-	3	-	-
Максимальный объем поддерживаемой памяти, Гб		3	2	2	2	3	3	2	2	3	3	3	3
Тип поддерживаемой памяти		PC133	PC600, PC800	PC600, PC800	PC600, PC800	PC133	PC133	PC600, PC800	PC600, PC800	DDR200, DDR266	DDR200, DDR266	PC133	PC133
Средняя розничная цена, USD		150	160	150	160	125	150	160	170	110	130	140	110

Табл. 3

Материнская плата	CPUmark 99	FPU WinMark 99	3Dmark2001	Disk CPU Utilization	DiskAccess Time	Disk Playback (Business)	Business Winstone 2001	Content Creation Winstone 2001
BL7	95,9	5800	5241	0,88	14,3	3290	34,3	43,1
TH7-RAID	98,9	5820	5741	0,59	14,2	3250	33,7	47,8
4S845A	94,1	5810	5246	2,12	15,8	3630	34,2	43,2
AX4BS Pro	95,2	5740	5224	0,86	14,4	3300	34,6	43,1
P4B	97,1	5920	5300	0,6	14,4	3260	33,3	45,1
CN-8G2AS	90,3	5750	5248	1,32	14,3	3180	30,5	44,7
CT-9BJA	94,9	5790	5238	0,73	14,4	3250	34,7	43,7
NB72-SR	95,6	5950	5276	1,86	15,7	3590	35,9	45,7
NT70-SA	100,0	5780	5761	0,45	14,3	3290	32,9	49,1
WT70-EC	97,7	5460	5629	0,72	14,4	3080	33,9	48,7
EP-4B2AE	96,6	5900	5290	0,52	14,3	3250	33,2	46,0
VC11	97,2	5780	5284	1,79	14,4	3310	34,5	45,8
GA-8IDXH	96,4	5730	5254	0,91	14,3	3250	34,2	45,4
GA-8TX	99,5	5800	5744	0,73	14,4	3170	34,1	49,0
D850GB	98,2	5850	5752	0,83	14,5	3200	33,9	47,6
D850MD	99,7	5940	5774	0,75	14,4	3150	34,1	50,3
P4A845S	95,9	5840	5276	0,39	14,4	3200	32,2	45,0
845 Pro2	96,2	5890	5230	0,72	14,4	3350	34,3	44,9
850 Pro2	100,0	5920	5762	0,84	14,3	4220	34,4	49,8
850 Pro5	102,0	5940	5629	0,34	14,3	3210	34,6	50,7
AV40	99,0	5810	5474	0,42	14,4	3160	34,3	45,7
SL-85DRV	99,5	5900	5480	0,81	14,5	3250	36,2	47,7
SL-85SD+	96,4	5830	5290	0,89	14,4	3100	34,5	45,9
845A3	93,3	5850	5288	0,17	14,3	3180	32,7	45,6

Табл. 4

Материнская плата	CPUmark 99	FPU WinMark 99	3Dmark2001	Disk CPU Utilization	DiskAccess Time	Disk Playback (Business)	Business Winstone 2001	Content Creation Winstone 2001
BL7	109	6920	5476	0,82	14,3	3300	35,6	47,5
4S845A	107	6860	5380	1,56	15,7	3510	35,4	46,6
AX4BS Pro	110	6900	5434	0,79	14,4	3320	34,7	47,5
P4B	110	6960	5490	0,51	14,4	3340	34,9	47,8
CT-9BJA	111	6980	5444	0,73	14,3	3250	34,8	47,7
NB72-SR	110	7200	5420	1,35	15,7	3630	38,3	49,0
NT70-SA	110	6840	5971	0,22	14,4	3110	36,2	49,1
VC11	110	6880	5493	1,23	14,4	3500	38,5	48,9
GA-8IDXH	108	6990	5422	0,85	14,3	3250	38,0	49,1
D850MD	110	7010	5764	0,62	14,4	3200	35,6	52,3
P4A845S	108	6840	5447	0,35	14,4	3200	34,2	48,8
845 Pro2	106	6870	5442	0,73	14,4	3400	34,6	47,9
850 Pro5	115	7060	5862	0,19	14,3	3210	36,1	55,5
AV40	110	6950	5664	0,38	14,4	3160	38,1	50,6
SL-85DRV	109	6990	5760	0,88	14,4	3250	38,2	50,6
SL-85SD+	109	7000	5512	0,78	14,4	3500	37,6	48,8
845A3	108	6900	5469	0,17	14,3	3220	35,2	49,2

менными офисными приложениями, в частности, Microsoft Office 2000. Тесты WinBench позволяют оценить производительность блока целочисленной арифметики, блока вычислений с плавающей запятой, а также возможности дисковой подсистемы. Утилита 3DMark2001 позволила оценить производительность при работе со сложной трехмерной графикой, характерной для современных игр. Все тесты проводились в видеорежиме 1024x768x32 с частотой 75 Гц. Результаты тестирования плат с процессором Pentium 4 1,7 ГГц приведены в табл.3, с процессором Pentium 4 2 ГГц (только для плат с гнездом Socket 478) — в табл.4. Оценки редакции "Hard'n'Soft" сведены в табл.5 и 6.

При подведении итогов был сделан вывод, что полное превосходство имеет чипсет i850, а точнее, память RDRAM над SDRAM. Ситуация с DDR не столь очевидна — будет ли жить P4X266, пока неизвестно, и дело даже не в судебном процессе, а в огромном влиянии Intel на компании, производящие материнские платы. В конце концов, уже скоро (обещали к концу 2001 г.) появятся первые платы на чипсете i845 с поддержкой DDR, тогда-то RDRAM и предстоит пройти серьезную проверку на прочность. Для тех же, кому экстремальная производительность необходима уже сейчас, лучший выбор — MSI 850 Pro5, считает редакция "Hard'n'Soft". У этой платы и производительность, и запас на будущее очень велики. Тем, кто нуждается в большой производительности, но не готов платить много за саму плату и еще больше за RDRAM, имеет смысл поближе присмотреться к Soltek SL-85DRV и Shuttle AV40 — пусть чипсет VIA Apollo P4X266 и в немилости, но работать-то хуже он от этого не стал. Для обычной работы и игр, когда не приходится постоянно обрабатывать в оперативной памяти немыслимые гигабайты, наиболее разумно будет приобрести плату EPoX EP-4B2AE и не самый дорогой процессор. Таким образом, выигрыш от оптимизации программ под Pentium 4 будет оправдан, а переплачивать за лишнюю пару сотен мегагерц (или около 10% общей производительности) не придется. Не придется также менять оперативную память. Если же

Табл. 5

Материнская плата	BL7	TH7-RAID	4S845A	AX4BS Pro	P4B	CN-8G2AS	CT-9BJA	NB72-SR	NT70-SA	WT70-EC	EP-4B2AE	VC11
Производительность	7	9	7	8	9	6	8	9	10	9	9	8
Удобство в использовании	8	8	8	9	10	7	8	9	9	8	9	9
Аксессуары	7	8	7	9	9	7	8	9	8	7	9	8
Документация	8	8	7	9	9	8	8	9	8	7	8	8
Оправданность цены	8	8	9	8	8	8	9	9	8	9	10	9
Оценка "Hard'n'Soft"	8/10	8/10	7/10	8/10	9/10	7/10	8/10	9/10	9/10	8/10	9/10	8/10

Табл. 6

Материнская плата	GA-8IDXH	GA-8TX	D850GB	D850MD	P4A845S	845 Pro2	850 Pro2	850 Pro5	AV40	SL-85DRV	SL-85SD+	845A3
Производительность	8	9	9	9	7	9	9	10	9	9	9	8
Удобство в использовании	9	9	8	8	7	9	7	9	8	8	9	9
Аксессуары	9	9	7	8	6	8	8	9	7	7	9	6
Документация	8	8	7	7	5	9	9	9	7	8	9	7
Оправданность цены	8	8	7	7	8	6	7	9	10	9	9	10
Оценка "Hard'n'Soft"	8/10	9/10	8/10	8/10	7/10	8/10	8/10	9/10	8/10	8/10	9/10	8/10

остановить свой выбор на малоизвестной, но добротной SuperGrace 845A3, то отпадет необходимость в специальном блоке питания. В любом случае на сегодняшний день можно выбрать для себя процессор в "устаревшем" 423-контактном корпусе, ведь покупка процессора с частотой выше 1,7 ГГц может быть оправдана только жесткой необходимостью, иначе это бессмысленная трата немалых денег, считают авторы. Однако при этом следует отдавать себе отчет в том, что при желании значительно "ускорить" свой компьютер придется менять не только процессор, но и материнскую плату — дни Socket 423, по-видимому, все же сочтены.

Е.Крылов в статье "Подсветка LCD-дисплеев" **(Компоненты и технологии, №6/2001, С.18...20)** рассказывает **о технических решениях, используемых для подсветки жидкокристаллических панелей мониторов.**

Аббревиатура LCD (Liquid Crystal Display) расшифровывается как дисплей на жидких кристаллах. Для формирования изображений в таких дисплеях используются элементарные ячейки, представляющие собой плоский конденсатор, диэлектриком в котором является жидкий кристалл а обкладками — поляризаторы. Свет, пройдя через первый поляризатор, станет линейно поляризованным. Затем в жидком кристалле плоскость поляризации света повернется на угол, определяемый напряжением, приложенным к кристаллу. Плоскость поляризации второго поляризатора перпендикулярна плоскости первого. Таким образом, меняя напряжение, приложенное к жидкому кристаллу, можно управлять степенью поглощения света, проходящего через такой конденсатор. Однако это означает, что в мониторе, кроме ЖК-ячеек, необходим хороший источник света.

Электролюминесцентная (EL) подсветка яеляется наиболее экономичной с точки зрения потребления электроэнергии (примерно 30 мВт). Она чаще всего используется в устройствах с батарейным питанием. Срок службы (снижение яркости наполовину от исходной) относительно невелик и составляет порядка 3...5 тыс. часов — в зависимости от установленной яркости свечения. Отличительные особенности электролюминесцентной подсветки:

- плоский источник света с максимальной толщиной 1,3 мм (1,5 мм с учетом выводов) обеспечивает равномерную подсветку большой площади;
- широкий диапазон напряжений питания переменного тока (максимальное значение 150 В) частотой 60...1000 Гц. При наличии повышающих преобразователей возможно питание от одной батареи с напряжением 1,5 В;
- цвет свечения — зелено-голубой, желто-зеленый и белый;
- рабочие характеристики типовых модулей питания: выходное напряжение — 110 В частотой 400 Гц; ток нагрузки — 8 мА (при T = 20°C и относительной влажности 60%);
- диапазон рабочих температур — 0...50°C;
- диапазон температур хранения — 20...60°C.

Светодиодная (LED) подсветка характеризуется самым длительным сроком службы — минимум 50 тыс. часов, и большей, чем у EL-подсветки, яркостью. Энергопотребление — примерно 60 мВт. Отличительные особенности светодиодной подсветки:

- низкое напряжение питания, нет необходимости использовать специальные преобразователи;
- длительный жизненный цикл — в среднем свыше 100 тыс. часов;
- возможность подсветки красного, зеленого, оранжевого и белого цветов или многоцветной подсветки (с переключением);
- боковая или матричная подсветка;
- типовое напряжение питания — 4,2 В;
- потребление — до 200 мА и выше;
- яркость — 250 кд/м^2;
- отсутствие генерации шумов.

Подсветка флуоресцентными лампами с холодным катодом (CCFL) используется преимущественно в больших плоскопанельных дисплеях. Большим достоинством CCFL является возможность получения бумажно-белого цвета, что делает CCFL практически единственным источником подсветки цветных дисплеев. Энергопотребление относительно велико (700 мВт), срок службы — 10...15 тыс. часов. Отличительные особенности:

- яркость — 450 кд/м^2;
- долговечность;
- излучение белого цвета;
- прямая и боковая подсветка;
- используется с многоцветными и/или точечно-матричными модулями ЖК-дисплеев.

До недавнего времени жидкостное охлаждение использовалось только в больших стационарных компьютерах. Современный ПК — тоже достаточно "горячее" устройство, охлаждать которое в ряде случаев непросто. **Так, может, использовать систему жидкостного охлаждения?** Несколько решений подобного рода рассматривает *С.Асмаков* в статье "Системы охлаждения: переходим к водным процедурам" **(КомпьютерПресс, №1/2002, С.123...125).**

Тенденция развития современных ПК такова, что выделение тепла в системном блоке постоянно растет. Если изначально основными тепловыделяющими элементами были центральный процессор и блок питания, то теперь к ним присоединились процессор видеокарты и чипсет материнской платы, отмечает автор. На очереди — модули памяти (как системной, так и видео). Вполне очевидно, что чем больше появляется источников тепла внутри корпуса, тем менее эффективным становится воздушное охлаждение — нагретый воздух, отводимый от отдельных компонентов системы, застаивается внутри системного блока, вызывая повышение температуры. Чтобы справиться с этим, требуется установка пары дополнительных вентиляторов, один из которых принудительно нагнетает наружный холодный воздух, а другой выдувает нагревшийся за пределы корпуса. В настоящее время выпускаются даже специальные управляющие системы, считывающие информацию с установленных в разных точках корпуса температурных датчиков и по заданным алгоритмам регулирующие режимы работы всех установленных внутри корпуса вентиляторов. Конечно, пока нельзя утверждать, что потенциал систем воздушного охлаждения исчерпан, считает автор. Ее возможностей хватит еще на некоторое время. Но чем ближе этот предел, тем меньше остается запас для "разгона" и прочих экстремальных ситуаций (например, для эксплуатации компьютера в условиях жаркого лета).

Принцип работы системы жидкостного охлаждения прост. В ее состав входят помпа, один или несколько теплообменников, устанавливаемых на охлаждаемые устройства, и радиатор. Перечисленные элементы системы соединяются при помощи шлангов (трубок) в замкнутый контур, внутри которого циркулирует жидкость. Как правило, в качестве рабочей жидкости используется дистиллированная вода с небольшим добавлением какого-либо спирта, например, метанола (это препятствует образованию микрофлоры). Помпа создает определенное давление, обеспечивая циркуляцию жидкости в системе. Далее жидкость поступает в теплообменник, проходя через который она нагревается, забирая тепло от активно нагревающегося компонента компьютера. Нагретая жидкость поступает в радиатор, в котором охлаждается. Далее все повторяется.

В качестве примера автор рассматривает несколько систем. Так, 3D Power Poseidon — одна из наиболее простых, предназначена для охлаждения видеокарты 3D Power Morpheus Titanium GeForse3 Ti500. Electric Iceberg и Swiftech MCW-462 используются для охлаждения центрвльного процессора. InfiniPro AquaCool, представляющая собой набор для самостоятельной сборки, позволяет охлаждать два устройства.

В отличие от остальных, специальный корпус системного блока Koolance представляет собой комплексное решение для организации жидкостного охлаждения. В нижней части корпуса имеется отдельный отсек, в котором размещены помпа и радиатор. Охлаждаются центральный и графический процессоры и блок питания.

Фирма Toshiba в своем ноутбуке Portege 3440CT впервые применила систему охлаждения центрального процессора, в которой используется водяной пар низкого давления.

Рассматривая достоинства и недостатки систем жидкостного охлаждения, автор подчеркивает, что применение таких систем позволяет не только снизить рабочую температуру охлаждаемых компонентов, но и сделать работающий системный блок значительно более тихим. Кроме того, если в случае воздушного охлаждения теплый воздух, отводимый от охлаждаемых компонентов, остается внутри корпуса, постепенно нагревая все остальное, то при использовании жидкостного охлаждения обеспечивается поддержание более низкой температуры внутреннего объема, что обуславливает более стабильную работу компьютера в целом. С учетом нынешней тенденции увеличения рассеиваемой современными компонентами тепловой мощности, водяное охлаждение обеспечит больший запас прочности при апгрейдах за счет более высокой эффективности.

Недостатками таких систем автор считает более сложную процедуру монтажа, некоторый риск разгерметизации водяного контура и значительно более высокую стоимость таких систем и их компонентов.

Значительный рост емкости флэш-памяти позволил создать аудиоплейеры, не содержащие вращающихся деталей. **Обзор технических характеристик и результатов тестирования MP3-плееров на флэш-памяти** привел *С. Маленкович* в статье ""Музыка" в дорогу" **(ПЛ: компьютеры, №1/2002, С.58...65).**

Плеер должен быть компактным, обеспечивать длительное воспроизведение любимой музыки и хорошее качество звучания, быть удобным в использовании и надежным, считает автор. Желательна невысокая стоимость. Плееры на кассетах, компакт-дисках и мини-дисках удовлетворяют перечисленным критериям в разной степени, но все-таки не идеальны. Компакт-диски вовсе не компактны, кассеты не обеспечивают должного качества, мини-дисковые плееры довольно дороги, а длительного звучания и высокой надежности тоже не обеспечивают. Гораздо симпатичней выглядят MP3-плееры на флэш-памяти. Скажем сразу, признается автор, они дороги сами по себе, и карты памяти к ним тоже недешевы. Зато, потратив один раз двести сотни "зеленых рублей" на плеер и дополнительную карту большой емкости, мы получим комплект, очень близкий к идеалу. Габариты пачки сигарет, масса менее сотни граммов, воспроизведение в любом удобном порядке, звук хорошего качества без "зависаний" и хрипов, полное отсутствие движущихся частей и высочайшая надежность — вот его особенности. А еще — работа в течение 10...15 часов от 1-2 батареек и привлекательная внешность. Аргументы весомы, но остался важный момент — выбрать лучший плеер из того изобилия, которое создали производители всех рангов и масштабов.

Тестировались плееры: MM-VX200, MM-FX500, Dpro, Nomad IIc, mp6421, M-Any DAH 200V, M-Any DAH OR 100, mPIO 64SV, Lyra, VaroMan. Их основные характеристики приведены в табл.1

Плееры оформлены по-разному. Так, если MM-VX200, MM-FX500, Dpro, Nomad IIc, mPIO 64SV и VaroMan похожи на обычные кассетные плееры, то mp6421, M-Any DAH 200V и M-Any DAH OR 100 по форме и размерам повторяют аудиокассету, что позволяет дома или в автомобиле использовать любой кассетный магнитофон для прослушивания музыки. Lyra очень компактен, его можно сравнить со спичечным коробком. Плееры MM-VX200, MM-FX500, mp6421, M-Any DAH OR 100 имеют пульт дистанционного управления, в M-Any DAH 200V и VaroMan встроен цифровой диктофон, а в mPIO 64SV — еще и FM-приемник.

Результаты тестирования приведены в табл.2.

Подводя итоги, автор отмечает, что по сочетанию удобства, компактности, качества звука и доступной цены побеждает плеер Thomson Lyra. Это "Лучшая покупка". Хороши также аппарат mPIO 64SV от Digitalway, в частности, наличием FM-приемника, что актуально для дачников и отпускников, отлученных от компьютера, и Aiwa HM-FX500, который позволяет записывать MP3-файлы без компьютера, что удобно в самых разных ситуациях. Удивляет, правда, то, что два последних снабжены LPT-интерфейсом. Музыка в них грузится очень медленно. Тем не менее, каждый из них достоин звания "Выбор редакции". Особого упоминания, по мнению автора, заслуживает mp6421 от российской компании NRG Electronics. Он удобен сам по себе (очень хорош пульт ДУ с дисплеем), но вторую жизнь ему дают музыкальный центр или автомагнитола — копирующий внешние размеры аудиокассеты плеер позволяет послушать музыку через большие динамики почти в любых условиях.

Табл. 1

Модель	MM-VX200	MM-FX500	Dpro	Nomad IIc	mp6421	M-Any DAH 200V	M-Any DAH OR 100	mPIO 64SV	Lyra	VaroMan
Производитель	Aiwa	Aiwa	Eisen	Creative	NRG Electronics	Hyun-Won Inc	Hyun-Won Inc	DigitaWay	Thomson	VaroVision
Носитель информации	Встроенная память, карта Smartmedia	Карты MultiMediaCart	Встроенная память, карта Smartmedia	Встроенная память, карта Smartmedia	Встроенная память, карты MultiMediaCart	Встроенная память, карты MultiMediaCart	Встроенная память, карта Smartmedia	Встроенная память, карта Smartmedia	Встроенная память, карты MultiMediaCart	Встроенная память
Объем памяти	32Мб+карта памяти	32Мб	96Мб+карта памяти	32Мб+карта памяти	64Мб+карта памяти	64Мб+карта памяти	32Мб+карта памяти	64Мб+карта памяти	32Мб+карта памяти	32Мб+карта памяти
Воспроизводимые форматы файлов	MP3	MP3, wma	MP3	MP3, wma	MP3, wma	MP3, wma, Audible	MP3, wma	MP3, wma	MP3	MP3, wma
Возможность переноса файлов произвольного типа	нет	нет	да	да	да	да	да	да	да	нет
Связь с компьютером	USB	LPT	USB	USB	USB	USB	Centronix (LPT)	LPT	USB	LPT
Питание	1 батарея AA	1 батарея AA	1 батарея AA	1 батарея AA	аккумулятор Ni-MH	аккумулятор Ni-MH	аккумулятор Ni-MH	2 батареи AAA	2 батареи AAA	1 батарея AA
Габариты, мм	86x68x22	90x65x18,5	95x65x26	65x93x21	102x63x11,5	102x63x11,5	102x63x11,5	90x66x18	57x46x28	95x65x26
Масса, г	72	80	81	88	60	51	45	75	80	81
Цена, USD	200	250	190	180	225	340	155	190	150	127

Табл. 2

Модель	Дизайн	Эргономика	Управление воспроизведением	Взаимодействие с компьютером	Сервис-функции	Качество звука
MM-VX200	4	3	4	4	4	3
MM-FX500	5	5	4	3	5	4
Dpro	4	3	4	4	3	3
Nomad IIc	4	5	4	5	5	3
mp8421	4	5	4	4	3	4
M-Any DAH 200V	4	3	3	4	4	3
M-Any DAH OR 100	4	4	3	3	3	3
mPIO 84SV	4	4	4	3	5	4
Lyra	4	5	4	5	3	5
VaroMan	3	4	5	3	4	4

М.ДОЛИНСКИЙ,
г.Гомель.

Решение задач с помощью рекуррентных соотношений

(Продолжение. Начало в №№3-4/2002)

4. Задачи на рекуррентные соотношения с тремя и более параметрами

4.1. Задача "Аудиосалон"

(Республиканская олимпиада школьников по информатике, 1997 г., тур 2, Задача 2, автор — Котов В.М.)

Во время трансляции концерта "Старые песни о главном-3" предприниматель К решил сделать бизнес на производстве кассет. Он имеет M кассет с длительностью звучания D каждая и хочет записать на них максимальное число песен. Эти песни (их общее количество N) передаются в порядке 1, 2, ..., N и имеют заранее известные ему длительности звучания L(1), L(2), ..., L(N). Предприниматель может выполнять одно из следующих действий:

1. Записать очередную песню на кассету (если она туда помещается) или пропустить ее.

2. Если песня на кассету не помещается, то можно пропустить песню или начать ее записывать на новую кассету. При этом старая кассета откладывается, и туда уже ничего не может быть записано.

Определить максимальное количество песен, которые предприниматель может записать на кассеты.

Входные данные записаны в файле KASS.DAT следующим образом. В первой строке файла находятся числа N, M и D. Начиная со второй строки, находятся числа L(1), L(2), ..., L(N), по одному в строке. Все числа — натуральные.

Ответ выдать в файл с именем KASS.OUT.

Пример:

```
KASS.DAT          KASS.OUT
6 2 7             5
5
2                 Пояснение :
2                 5+2     - 1-я кассета -  2 песни
4                 2+2+3   - 2-я кассета -  3 песни
2                           всего    -  5 песен
3
```

Т.е. в примере имеется 6 песен с длительностями звучания 5, 2, 2, 4, 2, 3. Требуется записать максимальное количество из них на 2 кассеты длительностью звучания по 7.

Введем и будем рассчитывать рекуррентную величину от трех параметров — номера песни, номера кассеты и единицы длительности звучания (минуты, например, для определенности). R[p,k,t] — максимальное количество из первых *p* песен, которое можно записать на *k* кассет, до минуты *t* на *k*-й кассете включительно.

Очевидно, что ответ будет содержаться в массиве R[N,M,D]. Понятно также, что начальные значения R[0,i,j] равны 0 для всех i и j. Посмотрим, как заполняется этот массив R при переходе от песни с номером P-1 к песне с номером P.

```
P=0       0 0 0 0 0 0 0
          0 0 0 0 0 0 0
P=1       0 0 0 0 1 1 1
L[1]=5    1 1 1 1 1 1 1
```

Первую песню записываем на кассету с первой минуты. Эта песня имеет длительность 5. Поэтому ее запись закончится на 5-й минуте. И это означает, что начиная с 5-й минуты 1-й кассеты, мы можем на данное множество кассет записать 1 песню из предъявленной одной песни. Теперь записываем следующую песню:

```
P=2       0 1 1 1 1 1 2
L[2]=2    2 2 2 2 2 2 2
```

Т.е. начиная со второй минуты первой кассеты, можно записать одну песню из предъявленных (если первую песню пропускать, а сразу от начала писать вторую песню). А начиная с 7-й минуты первой кассеты, можно записать 2 песни из предъявленных двух песен (с 1-й по 5-ю минуту будет записана первая песня, а с 6-й по 7-ю — вторая песня).

```
P=3       0 1 1 2 2 2 2
L[3]=2    2 3 3 3 3 3 3
```

Т.е. начиная с 4-й минуты второй песни, мы можем записать 2 песни (вторую и третью, длительностью по две минуты каждая), а начиная со второй минуты второй кассеты можем записать три песни (5 и 2 на первую кассету, 2 на вторую) и т.д.

Очевидно, что при попытке обработать очередную песню мы либо не изменяем значение элемента массива, либо добавляем к нему единицу в том случае, если песня помещается на кассету от начала до текущей минуты включительно.

Полностью учитывающее все нюансы рекуррентное соотношение выглядит следующим образом:

```
if t-L[p]>=0
  then R[p,k,t]:=max(R[p-1,k,t-L[p]]+1,R[p-1,k,t])
  else R[p,k,t]:=max(R[p,k-1,D],R[p-1,k,t]);
```

Т.е.:

ЕСЛИ текущая песня помещается на текущую кассету до минуты T (включая эту минуту T), **то** максимальное количество песен, из предъявленных P песен, которое можно записать на K кассет до минуты T (включительно), равно максимуму из количества песен, которые удалось записать до этого места при записи песни P-1, и увеличенного на 1 количества песен, которые удалось записать до минут T-L[P] включительно.

ИНАЧЕ (если текущая песня не помещается на текущую кассету до минуты T включительно), максимальное количество песен будет равно максимуму из количества песен, которые удалось записать до этого места при записи песни P-1, и количества песен, которое удалось записать до последней минуты звучания предыдущей кассеты.

Полный текст решения задачи приводится ниже:

```
program by97s2t2;
var
    R                 : array [0..40,0..10,0..60] of byte;
    L                 : array [1..40] of longint;
    i,p,k,t,N,M,D     : longint;

  function max(a,b:longint):longint;
    begin
      if a>b then max:=a else max:=b
    end;

begin
    read(N,M,D);
```

```
    for i:=1 to N do read(L[i]);
    for k:=1 to M do
      for t:=0 to D do R[0,k,t]:=0;
    for p:=1 to N do
      for k:=1 to M do
        for t:=0 to D do
          if t-L[p]>=0
            then R[p,k,t]:=max(R[p-1,k,t-L[p]]+1,R[p-1,k,t])
             else R[p,k,t]:=max(R[p,k-1,D],R[p-1,k,t]);
  writeln(R[N,M,D]);
end.
```

Внимательный читатель, очевидно, заметил, что на самом деле нам нет небходимости хранить весь трехмерный массив в памяти, достаточно только 2-х его компонентов-"слоев" (предыдущая и новая песня). Поэтому можно объявить массив R следующим образом:

```
R          : array [0..1,0..100,0..100] of byte;
```

А для переключения "слоев" использовать 2 переменные: p0 (индексирует предыдущий слой) и p1 (индексирует новый слой). Тогда расчетная часть программы изменится следующим образом:

```
p0:=0; p1:=1;
for p:=1 to N do
    begin
       for k:=1 to M do
          for t:=0 to D do
             if t-L[p]>=0
               then R[p1,k,t]:=max(R[p0,k,t-L[p]]+1,R[p0,k,t])
              else R[p1,k,t]:=max(R[p1,k-1,D],R[p0,k,t]);
       p0:=1-p0; p1:=1-p1;
    end;
writeln(max(R[0,M,D],R[1,M,D]));
```

А вся программа будет выглядеть так:

```
program by97s2t2;
var
   R              : array [0..1,0..100,0..100] of byte;
   L              : array [1..40] of longint;
   i,p,k,t,N,M,D,p0,p1 : longint;
  function max(a,b:longint):longint;
     begin
         if a>b then max:=a else max:=b
     end;
begin
   read(N,M,D);
   for i:=1 to N do read(L[i]);
   for k:=1 to M do
       for t:=0 to D do R[0,k,t]:=0;
  p0:=0; p1:=1;
  for p:=1 to N do
      begin
         for k:=1 to M do
           for t:=0 to D do
              if t-L[p]>=0
                then R[p1,k,t]:=max(R[p0,k,t-L[p]]+1,R[p0,k,t])
                else R[p1,k,t]:=max(R[p1,k-1,D],R[p0,k,t]);
        p0:=1-p0; p1:=1-p1;
      end;
  writeln(max(R[0,M,D],R[1,M,D]));
end.
```

4.2. Задача "Лабиринт"

(Четвертьфинал ACM, 1999 г.)

Карта лабиринта — это квадратное поле размером N*N. Некоторые квадраты этого поля запрещены для прохождения. Шаг в лабиринте представляет собой перемещение из одной разрешенной клетки к другой разрешенной клетке, смежной с первой по стороне. Путь — это некоторая последовательность таких шагов. Требуется подсчитать количество различных путей из клетки (1,1) в клетку (N,N) ровно за K шагов (т.е. оказаться в клетке (N,N) после K-го шага). Каждую клетку, включая начальную и конечную, можно посещать несколько раз. Начальная и конечная клетки всегда разрешены для прохождения.

Ограничения:

$1 < N \le 20$

$0 < K \le 50$

Ввод и вывод:

Первая строка входного файла состоит всего из двух чисел — N и K, разделенных одним или более пробелами. Следующие N строк, по N символов в каждой, содержат карту лабиринта, начиная с клетки (1,1). Символ "0" означает незапрещенную для прохождения клетку, а символ "1" означает запрещенную для прохождения клетку. Выходной файл содержит результат — количество различных возможных путей длиной K.

Пример ввода #1:

```
3 6
000
101
100
```

Вывод для примера ввода #1:

```
5
```

Пример ввода #2:

```
3 6
000
111
000
```

Вывод для примера ввода #2:

```
0
```

Будем составлять рекуррентное соотношение для величины A[k,i,j] — количества способов попасть в клетку (i,j) ровно за k ходов. Понятно, что за 0 ходов можно попасть только в клетку (1,1). Следовательно A[0,1,1]=1 и A[0,i,j]=0 для всех i и j.

На ходу k в клетку (i,j) по правилам, установленным в задаче, можно попасть (если она не запрещена для прохождения) только из соседних клеток (i-1,j), (i+1,j), (i,j-1) и (i,j+1).

А общее количество способов попасть в данную клетку после хода k будет равно сумме количеств способов попасть в соседние клетки на предыдущем (k-1)-м ходу, или 0, если клетка запрещена для прохождения.

Легко подсчитать, что для хранения всех значений рекуррентной величины A[k,i,j] требуется 50*20*20=20000 элементов. Даже 20000 элементов типа integer не могут поместиться в статической памяти Турбо-Паскаля. Поэтому будем хранить только реально необходимые два последних слоя.

Полное решение задачи:

```
{$R+,N+,E+}
program nee4c99g;
var
  A                : array [0..1,0..20,0..20] of comp;
  c                : array [1..20,1..20] of char;
  N,K,i,j,k0,k1,m  : longint;

begin
   readln(N,K);
  for i:=1 to N do
     begin
        for j:=1 to N do read(c[i,j]);    {Лабиринт}
        readln;
     end;

  for i:=1 to N do
     for j:=1 to N do A[0,i,j]:=0;
  for i:=1 to N do A[0,0,i]:=0;
  for i:=1 to N do A[0,i,0]:=0;
```


```
  A[0,1,1]:=1;

   K0:=0; K1:=1;
   for m:=1 to K do
      begin
         for i:=1 to N do
            for j:=1 to N do
               If C[i,j]='0'
                  then A[K1,i,j]:=A[K0,i,j-1] + A[K0,i-1,j] +
                                  A[K0,i,j+1] + A[K0,i+1,j]
                  else A[K1,i,j]:=0;
         K0:=1-k0; K1:=1-k1;
      end;
   if K1=1
      then writeln(A[0,N,N]:0:0)
      else writeln(A[1,N,N]:0:0);
end.
```

К сожалению, при ограничениях, приведенных в задаче, количество искомых вариантов может превысить число, помещающееся в величине типа comp. Т.е. полное решение задачи требует еще реализации "длинного" сложения. Оставляем это в качестве упражнения читателям.

4.3. Задача "Юбилей"

(Республиканская олимпиада школьников по информатике, 1998 г., День 1, Задача 2, автор — Котов В.М.)

В связи с открытием олимпиады-98 по информатике в г.Могилеве, N человек (N≤10) решили устроить вечеринку. Для проведения вечеринки достаточно купить MF бутылок фанты, MB бананов и MC тортов. Требуется определить минимальный взнос участника вечеринки. При покупке определенных наборов товара действуют правила оптовой торговли — стоимость набора товаров может отличаться от суммарной стоимости отдельных частей.

Написать программу, которая по входным данным определяет минимальный взнос участника вечеринки.

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

- в первой строке находятся числа N (количество человек, N≤10) и M (количество возможных наборов, M≤100000);
- в каждой из следующих M строк находятся 4 числа — F, B, C, S, где F, B, C — количество бутылок фанты, штук бананов и тортов в наборе (0≤F, B, C≤1000), а S — стоимость набора (S≤100000);
- в последней строке находятся числа MF, MB и MC (MF, MB, MF≤9).

Выходные данные должны находиться в текстовом файле с именем PART.OUT и содержать число V — минимальный взнос участника.

Итак, наша задача состоит в том, чтобы с учетом всех скидок сформировать массив O[0...9,0...9,0...9], где O[F,B,C] — минимальная стоимость приобретения F бутылок фанты, B бананов и C тортов.

Прежде всего разберемся с исходными данными.

В условиии сказано, что число скидок может быть ≤100000. Но, с другой стороны, известно, что на вечеринку нужно не более 9 товаров каждого вида — т.е. все скидки можно хранить в массиве SK[F,B,C] (минимальная скидка на приобретение F бутылок фанты, B бананов и C тортов), и ввод скидок может выглядеть следующим образом.

Вначале прописываем массивы O и SK значением maxlongint:

```
for i1:=0 to 9 do
   for i2:=0 to 9 do
      for i3:=0 to 9 do
         begin
            o[i1,i2,i3]:=maxlongint;
            sk[i1,i2,i3]:=maxlongint;
         end;
```

Если ничего не покупаешь, то ничего и не тратишь, поэтому обнуляем элемент O[0,0,0]:

```
o[0,0,0]:=0;
```

Теперь вводим очередную из 100000 скидок и с ее помощью формируем очередной элемент в массиве SK. При этом, если в скидке фигурирует число (бутылок фанты, бананов или тортов), большее 9, то мы его заменяем на 9. А в массив SK заносим соответствующую стоимость только если она меньше той, которая уже прописана в массиве на этой позиции, т.е. только если новая скидка лучше старой:

```
read(n,m);
for i:=1 to m do
    begin
       read (f,b,c,s);
       if f>9 then f:=9;
       if b>9 then b:=9;
       if c>9 then c:=9;
       if sk[f,b,c]>s then sk[f,b,c]:=s;
    end;
read(mf,mb,mc);
```

Теперь нужно, пользуясь массивом скидок SK, рекуррентно сформировать массив O, учитывая следующие соображения.

Во-первых. Пусть мы имеем оптимальное значение O[I1,I2,I3] — приобретения ровно I1 бутылок фанты, I2 бананов, I3 тортов и имеем скидку стоимостью S для F бутылок фанты, B бананов и C тортов.

Тогда мы можем построить оптимальную стоимость для приобретения I1+F бутылок фанты, I2+B бананов и I3+C тортов как минимальную из O[I1+F,I2+B,I3+C] и S+O[I1,I2,I3].

Во-вторых. Каждое из чисел I1+F, I2+B, I3+C может оказаться больше 9, поэтому мы их заменяем в этом случае на 9. В условиях задачи оговорено, что мы можем купить не менее (т.е. столько же или больше) товаров, чем нам нужно, главное, чтобы по наименьшей стоимости.

В-третьих. Мы должны применить ***каждую*** реальную (не равную maxlongint) скидку из массива исходных скидок SK к каждой реальной (не равной maxlongint) стоимости из массива O для хранения значений рекуррентно вычисленных оптимальных стоимостей закупок.

```
for f:=0 to 9 do
  for b:=0 to 9 do
    for c:=0 to 9 do
      begin
        s:=sk[f,b,c];
        if s=maxlongint then continue;
        for i1:=0 to 9 do
          for i2:=0 to 9 do
            for i3:=0 to 9 do
              begin
                if i1+f>9 then i1f:=9 else i1f:=i1+f;
                if i2+b>9 then i2b:=9 else i2b:=i2+b;
                if i3+c>9 then i3c:=9 else i3c:=i3+c;
                if (o[i1,i2,i3]<>maxlongint) and
                   (o[i1f,i2b,i3c]> s+ o[i1,i2,i3])
                   then o[i1f,i2b,i3c]:= s+ o[i1,i2,i3];
              end;
      end;
```

Напомним, что оператор CONTINUE обозначает — взять следующее значение переменной цикла.

Теперь нам осталось найти ***минимальный*** элемент части массива оптимальных скидок O[MF...9,MB...9,MC...9] и поделить его на количество участников (MF, MB, MC — исходные величины, обозначающие минимальное коли-

чество соответствующих продуктов, которое необходимо для вечеринки):

```
min:=maxlongint;
for i1:=mf to 9 do
   for i2:=mb to 9 do
      for i3:=mc to 9 do
         if o[i1,i2,i3]<min then min:=o[i1,i2,i3];
res:=min div n;
```

И последний штрих. Автор задачи В.М.Котов проверял на внимательность участников олимпиады, указывая в задаче место проведения вечеринки — Могилев, намекая, что деньги нужно использовать белорусские, а в то время в обороте ***не было*** меньше сотенных купюр, и поэтому результат нужно округлять до сотен:

```
res := trunc(min div (n*100))*100;          { округление }
if min mod (n*100) > 0 then res:=res+100;   { до сотен   }
```

Полный текст программы:

```
program by98d1t2;

var
    i1,i2,i3,mf,mb,mc,i1f,i2b,i3c : byte;
    o,sk                  : array [0..9,0..9,0..9] of longint;
    m,n,f,b,c,s,i,min,res : longint;

begin
  for i1:=0 to 9 do
     for i2:=0 to 9 do
        for i3:=0 to 9 do
            begin
                o[i1,i2,i3]:=maxlongint;
               sk[i1,i2,i3]:=maxlongint;
            end;
  o[0,0,0]:=0;
  read(n,m);
  for i:=1 to m do
     begin
         read (f,b,c,s);
         if f>9 then f:=9;
         if b>9 then b:=9;
         if c>9 then c:=9;
         if sk[f,b,c]>s then sk[f,b,c]:=s;
     end;

  for f:=0 to 9 do
    for b:=0 to 9 do
       for c:=0 to 9 do
          begin
            s:=sk[f,b,c];
            if s=maxlongint then continue;
            for i1:=0 to 9 do
              for i2:=0 to 9 do
                for i3:=0 to 9 do
                  begin
                     if i1+f>9 then i1f:=9 else i1f:=i1+f;
                     if i2+b>9 then i2b:=9 else i2b:=i2+b;
                     if i3+c>9 then i3c:=9 else i3c:=i3+c;
                     if (o[i1,i2,i3]<>maxlongint) and
                        (o[i1f,i2b,i3c]> s+ o[i1,i2,i3])
                        then o[i1f,i2b,i3c]:= s+ o[i1,i2,i3];
                  end;
          end;
  read(mf,mb,mc);
  min:=maxlongint;
  for i1:=mf to 9 do
    for i2:=mb to 9 do
       for i3:=mc to 9 do
          if o[i1,i2,i3]<min then min:=o[i1,i2,i3];
  res := trunc(min div (n*100))*100;          { округление }
   if min mod (n*100) > 0 then res:=res+100;  { до сотен   }
   writeln(res);
end.
```

(Окончание следует)

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

НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

(Продолжение. Начало в №№9-12/2001, №№1-4/2002)

Программирование на ассемблере под DOS

2.3. Печать "шестнадцатеричных циферек"

#1. Мы уже неоднократно юзали хорошую мнемоническую (ассемблерную) команду ADD. Напомню, что в результате выполнения команд

```
mov AX,2
mov BX,3
add AX,BX
```

в регистр AX у нас помещалась сумма (AX=AX+BX).

Мы смотрели на это дело под отладчиком и, к своей неописуемой радости, убеждались в том, что наша программулька действительно работает. Но что толку нам знать, что работает? Программа ведь не только работать должна, но еще и диалог какой-нибудь между юзером и компьютером обеспечивать! Например, спрашивать у него эти два числа и "выплевавать" на монитор результат их сложения. Вот именно, выводить на монитор, а не заносить в какой-то абстрактный регистр.

С клавиатурным вводом пока обождем, а вот с выводом (на монитор) разберемся прямо сейчас.

Как мы это сделаем? Вы уже неоднократно слышали, что в ассемблере все делается "ручками". Сейчас вы лишний раз убедитесь в том (некоторые замрут в ужасе), что это утверждение истинно. Для вывода значения регистра мы вовсе не "познакомимся с новым прерыванием". Даже такая простейшая операция как "вывод на дисплей значения регистра (переменной)" — это целая процедура. И не одна, как вы скоро в этом убедитесь. Страшно?

Поехали!!

#2. Задача — вывести на монитор значение регистра DL.

Народ! Давайте сразу расставим границы между ***кодом символа*** и его ***начертанием***.

Например, у нас есть значение F3h в DL. Как мы хотим это вывести? Как символы "F3" или же как ASCII-символ, соответствующий коду F3h?

Определяемся. Если в DL у нас F3h — то надо, чтобы именно "F3" у нас на монитор и выводилась. Не "э перевернутое", не "46 33", а именно "F3".

Помедитируйте и уловите разницу между "э", "46 33" и "F3".

Эту задачу мы немножко упростим. Для начала напишем процедуру, которая выводит только младшую тетраду регистра DL (цифру "3" в нашем примере). Для этого мы обратимся к процедуре WRITE_CHAR из прошлого номера. Именно она печатает нам на монитор символ, ASCII-код которого находится в DL.

Но тут загвоздка: в DL — код, а печатается-то символ :). А нам, собственно, именно две циферки шестнадцатеричного кода, как два символа, и нужно напечатать. Ну, или хотя бы младшую циферку этого кода...

Решается эта задача элементарно :). Главное — это правильно ее сформулировать!

Вот что я тут "нарисовал":

ЕСТЬ		НУЖНО		ЕСТЬ		НУЖНО	
Код	Символ	Символ	Код	Код	Символ	Символ	Код
00h	'?'	'0'	30h	08h	'?'	'8'	38h
01h	'?'	'1'	31h	09h	'?'	'9'	39h
02h	'?'	'2'	32h	0Ah	'?'	'A'	41h
03h	'?'	'3'	33h	0Bh	'?'	'B'	42h
04h	'?'	'4'	34h	0Ch	'?'	'C'	43h
05h	'?'	'5'	35h	0Dh	'?'	'D'	44h
06h	'?'	'6'	36h	0Eh	'?'	'E'	45h
07h	'?'	'7'	37h	0Fh	'?'	'F'	46h

Только вместо вопросительных знаков должны быть всякие соответствующие символы (посмотрите в таблице ASCII-кодов, какие они на вид страшные!).

Процедурка наша вот что должна делать — всего-навсего перевести (конвертировать) тетраду в код соответствующего ей символа... Завернуто? Если разобраться, то не очень-то и завернуто.

Смотрите, в DL у нас 03h. Мы хотим эту "3" на монитор вывести. Если вызовем процедуру WRITE_CHAR, то у нас выведется символ "сердечко". А надо, чтобы вывелся символ "3", код которого — 33h. Соответственно смотри по табличке и для остальных кодов.

А теперь обратите внимание, насколько шестнадцатеричная цифра (тетрада) отличается от ASCII-кода, этой цифре соответствующего. Скажу сам — на 30h для цифр от "1" до "9", и на 37h для цифр от "A" до "F". Т.е. "конвертацию" мы запросто можем сделать командами add DL,30h (если тетрада в диапазоне 0...9) и add DL,37h (если тетрада в диапазоне A...F). Короче, вот код:

```
;-[WRITE_HEX_DIGIT, V1]---------------------------------
;Печатает одну шестнадцатеричную цифру (младшую
;тетраду DL, старшая тетрада должна быть равна 0).
;На входе: DL - цифра
;На выходе: ничего
;Прерывания: нет
;Процедуры: WRITE_CHAR
;-----------------------------------------------------------
WRITE_HEX_DIGIT proc
      push DX
      cmp    DL,0Ah
      jae    HEX_LETTER
      add    DL,30h
      JMP    WRITE_DIGIT
 HEX_LETTER:
      add    DL,37h
 WRITE_DIGIT:
      call WRITE_CHAR
      pop    DX
      ret
WRITE_HEX_DIGIT endp
```

Сначала, естественно, сохраняем изменяемые регистры, они будут восстановлены в конце процедуры (команды PUSH и POP соответственно).

Потом у нас организовано логическое ветвление. Сравниваем значение DL с "общей границей" наших двух диапазонов (команда — CMP, "граница" — 0Ah). Если это значение больше или равно 0Ah, то прыжок на метку HEX_LETTER, прибавление к DL 37h и печать цифры (WRITE_CHAR). Иначе добавляем 30h и без всяких условий перепрыгиваем на вызов WRITE_CHAR (минуя add DL,37h).

Все. Тестируем.

#3. Про тестирование разговор отдельный. В данном случае мы можем запросто проверить нашу процедуру на абсолютно всех возможных значениях этой тетрады (всего-то ничего — 16 вариантов). Но намного более правильно, проанализировав алгоритм, установить своего рода "критические" значения, на которых целесообразно проводить проверку. Плюс, естественно, минимальное и максимальное значения.

```
TESTING proc
      mov DL,00h
      call WRITE_HEX_DIGIT
      mov DL,01h
      call WRITE_HEX_DIGIT
      mov DL,09h
      call WRITE_HEX_DIGIT
      mov DL,0Ah
      call WRITE_HEX_DIGIT
      mov DL,0fh
      call WRITE_HEX_DIGIT
      call EXIT_COM
TESTING endp
```

Если сия "тестовая" (она же — главная) процедура выведет на монитор

```
019AF,
```

значит, мы с высокой долей вероятности можем быть уверены в том, что процедура WRITE_HEX_DIGIT работает правильно на всех значениях младшей тетрады DL.

Те, кто не просто скопировал процедуру из буфера обмена, а действительно разобрался с тем, как она работает, сами знают, что значение старшей тетрады нашей процедуре НЕбезразлично. Оно должно быть равным 0.

#4. Что мы имеем? Процедуру для вывода на дисплей одной шестнадцатеричной цифры — младшей тетрады (в DL). Но нам-то нужно вывести две! Сначала — старшую цифру-тетраду, и только потом — младшую!

Таким образом, очередная задача разбивается на две части — печать старшей тетрады DL и печать младшей тетрады DL.

Первая "подзадача" решается легко — просто нужно старшую тетраду переместить на место младшей и вызывать процедуру (основательно протестированную и на 99,9% работающую) WRITE_HEX_DIGIT.

А вторая подзадача заключается в восстановлении предыдущего (мы же тетраду перенесли) значения DL и, опять же, в вызове WRITE_HEX_DIGIT.

(Вот теперь-то вы уж точно почувствуете всю прелесть "дробления кода на процедуры"!)

Перенос тетрады мы осуществим при помощи команды SHR, которую в умных книжках обзывают как "логический сдвиг разрядов операнда вправо". Объясню.

Представьте себе деревянную доску длиной в 8 бутылок пива и шириной в одну (в принципе, доску эту можно и в два раза длиннее представить, но тогда на ней надо "DX" написать, мы же пока только "DL" напишем). А еще дурня, у которого на лбу написано SHR. Так вот, если этому придурку стукнуть по хребту, то он слева от доски поставит ПУСТУЮ бутылку, а остальные сдвинет на одну позицию вправо, в результате чего самая правая бутылка, естественно, с доски упадет.

Бутылки, которые сразу стояли, могут быть пустыми или полными, а вот дурень SHR ставит только пустые, и только слева:

```
"исходное "   11110011
SHR           01111001
SHR           00111100
SHR           00011110
SHR           00001111
```

Ну, тут и ежу все понятно. Четыре раза дурню по хребту надо дать, чтоб старшая тетрада на место младшей переместилась. На самом деле самая правая бутылка, перед тем как об землю разбиться, на некоторое время в воздухе зависает, но вы пока этим голову не забивайте.

Реализовывается этот сдвиг вот как:

```
mov DL,11110011b
mov CL,4
shr DL,CL
```

В DL — наша цепочка битов (11110011b = F3h, естественно). В CL заносим "на сколько позиций" нам нашу цепочку

сдвинуть. Ну и SHR — это дурень, который сдвигает вправо, а слева нули дописывает.

#5. Думаете, это все? Разогнались! Не все так просто :).

Процедура WRITE_HEX_DIGIT у нас требует, чтобы первой тетрадой были только одни нули. Я заострял на этом ваше внимание. При печати первой тетрады это условие соблюдается — SHR слева нули дописывает.

А вот при печати второй цифры нужно вот что — ничего никуда не сдвигая, обнулить старшую тетраду, а младшую (которая, собственно, и есть "цифра") оставить в покое.

Решим мы эту задачу при помощи логической операции "И" (AND, англ.). Кто статьи С.Белоусова в прошлых номерах журнала читал [1], сами вспомнят, что это за "И" такое. А кто не читал, я напомню "таблицу истинности" для данной логической операции:

x1	x2	and
0	0	0
0	1	0
1	0	0
1	1	1

А теперь и для особо одаренных:

```
0 and  0  =  0
0 and  1  =  0
1 and  0  =  0
1 and  1  =  1
```

Смотрите, интересно как получается. Если мы AND чего-либо (ноль или единичку) с 0 делаем, то у нас в результате 0 и только 0 получается. А если AND с единичкой — то ЧТО БЫЛО, ТО И ОСТАЕТСЯ (это и есть потаенный дZенский смысл команды AND).

Решение нашей проблемы (обнулить старшую тетраду, а младшую оставить без изменений), таким образом, сводится к тому, что старшую тетраду нужно "AND 0", а младшую — "AND 1". Т.е. значению DL с 00001111b (оно же 0Fh) "AND" сделать. На ассемблере это вот как выглядеть будет:

```
and    DL,00001111b
```

Естественно, 00001111b = 0Fh.

#6. Уфф... Вот что должно получиться в итоге:

```
;-[WRITE_HEX, V1]------------------------------
;Печатает две шестнадцатеричные цифры
;На входе: DL - типа цифры две :))
;На выходе: ничего
;Прерывания: нет
;Процедуры: WRITE_HEX_DIGIT
;-----------------------------------------------
WRITE_HEX proc
       push CX
       push DX
       mov    DH,DL
       mov    CL,4
       shr    DL,CL
       call WRITE_HEX_DIGIT
       mov    DL,DH
       and    DL,0Fh
       call WRITE_HEX_DIGIT
       pop    DX
       pop    CX
       ret
WRITE_HEX endp
```

Ну, что тут объяснять? Я уже разжевал все! Единственное, что могу добавить — это про mov DH,DL. Этой командой мы значение регистра копируем, перед тем как биты "ему" сдвинуть. А потом командой mov DL,DH статус-кво восстанавливаем, чтобы и младшую цифру напечатать.

Все. Тестируем. Вроде, должно работать.

#7. Так сказать, "к вопросу о шаблонах мышления"...

Мы тут доооолго занимались с тетрадами. Вроде, успешно. Но, когда при тестировании понимаемости материала пяти "подопытным" было предложено самостоятельно написать процедуру для вывода на монитор "большого" регистра (DX), они все как один начали сдвигать байты... :(Народ! Это не есть правильно!

```
;-[WRITE_HEX_WORD, V1]-------------------------
;Печатает шестнадцатеричное слово
;На входе: DX - типа слово :))
;На выходе: ничего
;Прерывания: нет
;Процедуры: WRITE_HEX
;-----------------------------------------------
WRITE_HEX_WORD proc
  push DX
  xchg DL,DH
  call WRITE_HEX
  xchg DL,DH
  call WRITE_HEX
  pop  DX
  ret
WRITE_HEX_WORD endp
```

Команды xchg DL,DH и xchg DH,DL, кстати, работают абсолютно одинаково. Операнды просто меняются между собой значениями. В качестве одного из операндов может выступать память.

#8. Ну, и напоследок, информация к размышлению.

Команду shr можно использовать для деления целочисленных операндов без знака на степени 2 :)).

```
mov    cl,4
shr    ax,cl
```

Думаете, эти "фокусники", которые в уме офигенные уравнения решать умеют, шибко умные? Нет! Просто, они — люди ЗНАЮЩИЕ. А делить на десять и вы умеете...

Над этим я настоятельно рекомендую доооооолго помедитировать. А еще над тем, что девушки весьма и весьма любят, когда им фокусы показывают. Впрочем, это (вычисления в уме) тема отдельная. Ее мы тоже когда-нибудь коснемся :). MATRIX MUST DIE!

2.4. Печать десятичных "циферек"

#1. Для тех, кто медитировал над главой 1.1, алгоритм должен быть понятен как 2+2=(вписать желаемое). Кто не "въедет" в алгоритм WRITE_DECIMAL — научитесь сначала переводить числа между радиксами на листике в клеточку.

Итак, ставим новую задачу. В DX у нас шестнадцатеричное число. Нам нужно вывести его на экран монитора в "десятичном формате". Еще раз обращаю ваше внимание на то, что существует множество способов ее решения. Мы же выбрали способ, который:

а) вам легче всего будет понять;

б) требует минимального количества "новых" команд.

Кто скажет, что мы не правы — пусть первый кинет камень в эхо-конференцию RTFM_Helpers.

#2. Прежде всего посмотрите на процедуру WRITE_HEX_DIGIT и вспомните, какой алгоритм положен в основу ее работы. Вспомнили? Рад за вас!

А теперь мы познакомимся с командой деления. Что будем делить? Ну естественно, "регистры" :). Новая команда называется "деление беззнаковое" (DIVide unsigned). Что такое делимое/делитель/частное, я вас "грузить" не буду, это в учебнике арифметики для младших классов более чем понятно разжевано — во всяком случае, детишки это понимают (кто скажет, что 10/3=3, а остаток — 333 в периоде, тот неправ. Кто скажет, что остаток будет равен 1, скажет правильно.)

Следующий кусок кода демонстрирует деление значения регистра AX на значение регистра BL:

```
mov    ax,10d
mov    bl,3d
div    bl
```

Обратите внимание, ДЕЛИМОЕ у нас может располагаться только в регистре AX (другими словами, делимое задается неявно), а ДЕЛИТЕЛЬ — в любом регистре. Кто не верит — посмотрите под отладчиком...

1. Если размер делителя — байт, то после операции частное помещается в AL, а остаток в AH.

2. Если размер делителя — слово, то делимое должно быть расположено в паре регистров DX:AX (младшая часть делимого в AX). После операции частное помещается в AX, а остаток — в DX.

3. Если размер делителя — двойное слово, то делимое должно быть расположено в паре регистров EDX:EAX (младшая часть делимого находится в EAX.) После операции частное помещается в EAX, а остаток — в EDX.

Внимание, подводный камень! В следующем примере:

```
mov ax,10d
mov bx,3d
div bx
```

у вас вовсе не AX на BX делиться будет, а парочка DX:AX на BX.

Помедитируйте над этим :).

#3. А теперь, собственно, сама процедура вместе с ее традиционным разжевыванием.

```
;-[write_decimal, v1]----------------------
;печатает десятичное беззнаковое число
;на входе: dx - типа число
;на выходе: ничего
;прерывания: нет
;процедуры: write_hex_digit
;-------------------------------------------
write_decimal proc
  push ax
  push cx
  push dx
  push bx
  mov  ax,dx  ;(1)
  mov  bx,10d ;(2)
  xor  cx,cx  ;(3)
 non_zero:
  xor  dx,dx  ;(4)
  div  bx     ;(5)
  push dx     ;(6)
  inc  cx     ;(7)
  cmp  ax,0   ;(8)
  jne  non_zero
 write_digit_loop:
  pop  dx     ;(9)
  call write_hex_digit ;(10)
  loop write_digit_loop
  pop bx
  pop dx
  pop cx
  pop ax
  ret
write_decimal endp
```

Алгоритм простой — пока частное не равно 0, делим его, делим и еще раз делим на 10d, "запихивая" остатки в стек. Потом — извлекаем из стека. Вот и вся "конвертация" из HEX в BIN. Это если в двух словах. А если подробно, то вот что получается.

Бряк 1 — подготавливаем делимое. Как уже говорилось, оно у нас задается неявно — обязательно через AX. А параметр у нас процедуре передается через DX. Вот и перемещаем.

Бряк 2 — это, собственно, делитель.

Бряк 3 — очищаем CX. Он у нас будет в качестве счетчика. О нем мы еще поговорим.

Бряк 4 — очищаем DX. Если не очистим, то мы не 1234h какое-нибудь на 10 делить будем, а 12341234h. Первое 1234 нам надо? Вот и я говорю — очищаем!

Бряк 5 — делим! Частное — в AX, остаток — в DX.

Бряк 6 — заносим остаток (DX) в стек ;).

Бряк 7 — CX=CX+1. Это мы считаем, сколько раз "щемили остаток", который "щемится" по кругу (прыжок на метку non_zero), пока AX не равно 0 (бряк 8). То есть делим, делим AX, пока не окажется, что делить, собственно, нечего.

Так-с... Деление закончено, число раз, которое мы делили AX до его полного обнуления, хранится в CX.

Дальше все просто. Нам нужно такое же количество раз (CX) извлечь значение (DX) из стека. И это будет "HEX", переведенный в "DEC" (оно же — число в двоичном коде, разобранное на последовательность десятичных цифр).

Помните, как у нас организуется цикл? Через loop и CX? Если бы в качестве счетчика мы использовали какой-нибудь другой регистр, то пришлось бы извращаться со всякими метками и прыжками... А так все просто, все продуманно :). Цикл, в теле которого ИЗВЛЕЧЬ ЦИФРУ (бряк 9) и НАПЕЧАТАТЬ ЦИФРУ (бряк 10). Столько же раз, сколько мы и делили наше исходное шестнадцатеричное число.

Для тех, кто не понял: бряк — это брекпоинт. Для тех, кто еще не знает, что такое брекпоинт — ищите объяснение в "DZebug. Руководство юZверя" [2].

#4. Тестируем!

```
testing proc
       mov    dx,12345d
       call write_decimal
       call exit_com
testing endp
```

Двое из десяти "подопытных" чайников (есть такие) возмутились:

— В DX же только четыре циферки влазят!

— Ага! Аж два раза по четыре!

mov dx,'DZ'

Как видите, туда еще и две буковки "влазят" ;). А то и все четыре, если кавычки считать...

Медитируйте!

2.5. "Hello, World!" или Изврат-II

#1. Мы уже писали программу "Hello, World!" с использованием 13-й функции 10-го прерывания. Посмотрите на ее исходник...

Сегодня мы "слабаем" еще одно "Hello, World!", но уже несколько другим способом. Какой из этих способов более дZенский — решайте сами ;).

Для начала мы создадим блок данных после всех-всех-всех процедур, но между директивами начала и конца сегмента.

Блок данных будет выглядеть следующим образом:

```
abc db 'Hello, World-2$'
```

Вы должны спросить: "А почему мы хотим напечатать Hello, World-2, а в конце строки у нас 2$? $ — это что? World за баксы продавать, что ли? Да ну вас..."

Эту строчку мы будем выводить на монитор особым дZенским способом — посимвольно. Т.е. возьмем первый символ из блока данных, выведем, потом второй и т.д. аналогично, пока не встретим символ "$".

Опять-таки, это если в двух словах. Но ведь наверняка вам этого покажется мало ;).

"Щемить" символы мы будем двумя способами (тут я хотел было написать "неправильным" и "правильным", но потом передумал и решил обозвать их "первым" и "вторым").

В алгоритм первого способа вы и сами без труда "въедете" (я только "рабочую часть" приведу, пуши с попами сами

проставляйте):

```
...
next:
 mov dl,[BX]
 cmp dl,'$'
 je finish
 call write_char
 inc BX
 jmp next
finish:
...
```

#2. А вот второй способ немножко более "наворочен" :). Чтобы в нем разобраться, мы сначала познакомимся со следующей группой команд: LODSB, LODSW, LODSD.

Их назначение — загрузка элемента из последовательности (строки, цепочки) в регистр-аккумулятор AL/AX/EAX.

Для тех, кто не понял — наша строчка "Hello, World-2$" как раз и является "последовательностью/строкой/цепочкой" из элементов размером в байт.

Адрес цепочки передается через DS:ESI/SI, сами "элементы" (фиксированной ширины) возвращаются в AL (байт, команда LODSB), AX (слово, команда LODSW) или EAX (двойное слово, команда LODSD). В общем, последние буковки команд как раз и указывают на размерность элемента— [B]yte, [W]ord, [D]ouble word. Т.е. размер мы определяем неявно.

После выполнения одной из этих команд значение регистра SI изменяется на величину, равную длине элемента, но...

Пришло время еще одну большую тайну познать, братья. В справочнике Юрова [http://hi-tech.nsys.by:8101/tools/help_systems/urov.zip] написано, что знак этой величины зависит от состояния флага df:

- df=0 — значение положительное, т.е. просмотр от начала цепочки к ее концу;
- df=1 — значение отрицательное, т.е. просмотр от конца цепочки к ее началу.

Обидно, да? Флаги-то мы с вами еще не "расколупали"... Чего-ж дальше-то делать, а?

Как что? "Пропивать" стандартную процедуру и "колупать" ее, "колупать", "колупать"...

```
;-[write_string, v1]-------------------------
;печать строки символов на мониторе.
;(Строчка оканчивается символом '$'
;на входе: ds:dx - адрес строки
;на выходе: ничего
;прерывания: нет
;процедуры: write_char
;---------------------------------------------
write_string proc
  push ax
  push dx
  push si
  pushf                ;(1)
  cld                  ;(2)
  mov  si,dx           ;(3)
 string_loop:
  lodsb                ;(4)
  cmp  al,'$'          ;(5)
  jz   end_of_string   ;(6)
  mov  dl,al           ;(7)
  call write_char      ;(8)
  jmp  string_loop     ;(9)
 end_of_string:
  popf                 ;(10)
  pop  si
  pop  dx
  pop  ax
  ret
write_string endp
```

Итак, что делает команда LODSB, вы уже поняли. А вот с df=0/df=1 пока что непонятно. Будем разбираться.

Нам нужно, чтобы "просмотр цепочки" осуществлялся командой LODSB слева направа, для этого нужно установить значение df=0. Делаем мы это при помощи команды CLD (бряк 2).

Если нам нужно, чтобы "просмотр цепочки" осуществлялся справа налево, мы используем команду std (можете попробовать. Ерунда получится).

А теперь вспомним "золотое правило". Все регистры, которые мы изменяли ВНУТРИ процедуры, "на выходе" должны восстанавливать свои ПРЕДЫДУЩИЕ значения (кроме тех регистров, через которые мы возвращаем РЕЗУЛЬТАТ).

Так вот, то же самое касается и флагов, которые мы изменяем.

Для регистров мы использовали команды PUSH и POP. Для регистра флагов (изменять у которого мы можем только БИТЫ) используются команды PUSHF (записать в стек значения флагов, бряк 1) и POPF (извлечь из стека, бряк 10).

Примечание: Это несколько вольное положение (хотя, в общем-то, верное). Видимо, здесь следует хотя бы добавить, что сохранять/восстанавливать флаги нужно при изменении специальных флагов (if, df).

Теперь "колупаем" дальше...

Бряк 3. Мне удобнее передавать данные "в процедуру" через регистр DX, о чем и написано в заголовке. А LODSB (бряк 4) хочет, чтобы адрес ему в SI подавали. Удовлетворим его желание :).

Бряк 4. После выполнения этой команды ASCII-код первого символа "цепочки" "Hello, World-2$" помещается в AL, а значение регистра SI увеличивается на 1, и указывает теперь на второй элемент цепочки (символ "e").

Бряк 7. Удовлетворяем "пожелания" процедуры WRITE_CHAR. Из AL переносим в DL и печатаем (бряк 9), WRITE_CHAR'ом.

Тэк-с... Мы пропустили бряки 5-й и 6-й...

Смотрите. На бряке 9 мы "безусловно" зацикливаем "извлечение" и печать элементов цепочки. Безусловно — это значит до потери пульса. Чтобы этого не произошло, каждый из элементов цепочки мы сравниваем с символом-конца-цепочки (в нашем случае это "$", но можно использовать и любой другой). Если текущий символ равен символу-конца-цепочки, то выпрыгиваем (бряк 6) из этого безусловного цикла, восстанавливаем статус-кво (бряк 10), и все на этом...

#3. Тестируем!!

```
testing proc
       lea dx,abc
       call write_string
       call exit_com
testing endp
```

Напечаталось то, что надо?

Если true — читайте дальше.

Если false — "расколупывайте" и медитируйте до полного просветления...

Литература

1. А.Белоусов /HI-TECH. Алгебра логики и цифровые компьютеры. — Радиомир. Ваш компьютер, 2002, №2, С.22.

2. Nyron /HI-TECH. DZebug. Руководство юZверя. — Радиомир. Ваш компьютер, 2002, №№1-2, С.21

(Продолжение следует)

Sphinx C-- — язык не для всех

Часть 1

О.ЧЕРЕДНИЧЕНКО,
Днепропетровская обл.
E-mail: olegka.cher@newmail.ru
FidoNet: 2:464/86.62 aka /156.62

Зачем вам нужен "еще один" компилятор Си?

Время идет, все вокруг непрерывно преображается: развивается, деградирует или остается без видимых глазу изменений... Когда я изъявил желание написать о замечательном языке программирования Sphinx C--, редакция любезно предоставила все материалы о C--, которые были опубликованы на страницах журнала. Это статьи уважаемого К.Хилько из Минска [1]. С тех пор прошло достаточно много времени, а тем более, в компьютерном мире, где события развиваются порою молниеносно. У Sphinx C-- появился свой стойкий круг почитателей, пытающихся использовать его для всего подряд, особенно для написания маленьких и быстрых программ — игр, мини-демо (интро), резидентов (TSR), утилит, драйверов и, наконец, плагинов.

Автор C-- Peter Cellik (Canada) смог создать компилятор, который вполне законно занял свою нишу в области трансляторов Си-подобных языков. Но, оказывается, есть человек (Михаил Шекер, Тверь), который, приняв эстафету из рук Peter'a Cellik'a, развивает проект, придает ему свежие силы и выдвигает новые интересные идеи. У Sphinx C-- открылось второе дыхание! По сути, Михаилу удалось превратить компилятор Sphinx из достаточно узкопрофильного средства для системного программирования в полнофункциональную программную среду с большими возможностями, базирующуюся на мощном языке низкого уровня. Да-да, именно низкого, ведь для того чтобы успешно программировать на C--, необходимо знать ассемблер 80x86 и иметь достаточно хорошее представление о Си.

Компилятор и ранее умел создавать очень оптимальный машинный код (ближайший конкурент — ассемблер — не в счет!), для этого, собственно, и создавался. Теперь же он научился более эффективно использовать современные 32-битные команды, генерировать код в .exe файлы, в том числе и для Windows 9x\Me\NT. Появилась поддержка вещественных чисел и структур данных (вот разве что объектов пока немножко не хватает. Впрочем, в ближайшее время М. Шекер собирается ввести операции со структурами (сложение, вычитание, а также логические) с использованием, по возможности, мультимедийных инструкций — будет полезно при работе с графикой. С некоторой натяжкой это можно, в принципе, назвать объектной ориентированностью). Программы потихоньку перекочевали в защищенный режим работы процессора с большим количеством доступной памяти. В то же время, язык сохраняет за собой яркость и выразительность, неперегруженный синтаксис, сходный с синтаксисом Си, и потрясающую эффективность ассемблера. Разумеется, ко всему этому не смогли остаться равнодушными демомейкеры, поскольку кодить в C-- все-таки намного удобнее, чем в более низкоуровневых концептуальных приближениях к ассемблеру, например, в Chasm. Уже сейчас наработано большое количество библиотек, компилятор непрерывно развивается и совершенствуется М.Шекером. В сетевой фидоэхоконференции RU.C-- [2] он предлагает посетить свой сайт [3], на котором всегда можно найти самую свежую версию компилятора, полезную информацию по языку (русская документация на Sphinx C-- доступна в виде гипертекста в формате Norton Guide), множество библиотек, утилит и примеров программ. На момент написания статьи на сайте была выложена версия компилятора 0.238 beta 5, в которой можно работать с битовыми полями и имеется встроенный компилятор ресурсов. Предлагаю и вам принять участие в тестировании этого продукта и в совместном проекте Open Source по созданию новой библиотеки для компилятора (по представлению автора, это будет набор функций, сходных по названиям, входным и выходным параметрам со стандартными функциями Turbo C).

Что нового в последней версии компилятора по сравнению с предыдущими? Слово Михаилу Шекеру:

- расширены возможности указателей;
- в структурах можно объявлять процедуры;
- в стековые процедуры можно передавать параметры через регистры;
- введена поддержка связанного присваивания;
- введена операция разрешения видимости "::";
- добавлены зарезервированные слова "RETURN", "static";
- компилятор, при включенной оптимизации на размер кода и при использовании динамических процедур в качестве макросов, может заменять оператор return/RETURN на goto/GOTO;
- выполнена оптимизация связки IF()RETURN, if()return и связок else с операторами передачи управления (goto, break, BREAK ...);
- выдаются предупреждения о возможно неисполнимом коде;
- допускается использовать метки вне процедур;
- сделана поддержка наследования классов;
- введена поддержка битовых полей структур.

Язык Си-- был задуман как *высокоэффективный* системный язык, поэтому он не любит терять время и байты. Генерируемый код в большинстве случаев получается более компактным и более быстрым, чем в C/C++. Имеет ли смысл сравнивать его с Си? С BC 3.1 уже сравнивать можно, для сравнения с другими он еще слабоват. C-- пытается занять промежуточное положение между ассемблером и Си, и поэтому его сравнение с Си, а тем более с C++, немного неуместно — другие задачи, другое применение. К тому же, рассматриваемый нами C--, в отличие от C/C++, полностью машиннозависим, и кроссплатформенная разработка приложений и решение некоторых других задач на нем затруднены или невозможны — увы, Sphinx C-- умеет только то, чему его научили. Следует отметить, что имя "C--" носят несколько непохожих друг на друга языков, спроектированных так, чтобы Си-программистам не требовалось приложения усилий для изучения совершенно нового и незнакомого языка. Единственное, что их объединяет — попытка занять место между ассемблером и Си. Об альтернативных проектах C-- можно узнать на [4-6], на [5], кроме того, имеется информация о многоплатформенном компиляторе.

Таким образом, бесспорно, что есть резон пополнить свой инструментарий программиста этим компилятором. Изучить C-- не просто, а очень просто. Если вы знаете (а в особенности если вынуждены иногда применять) ассемблер, то безусловно оцените краткость и выразительность синтаксиса C--. Чтобы быстро освоить программирование в среде Sphinx, необходимо написать буквально несколько крохотных программок, чтобы проникнуться низкоуровневой идеологией этого языка, так близкого по духу системным программистам.

Ну а для решения специфических задач (типа БД, сложных расчетов или сетевизмов) есть специфические (и почти всегда высокоуровневые) средства.

12 причин, по которым я использую Sphinx C--

1. Компилятор позволяет создавать файлы под DOS с 16-битным кодом для реального режима. Допускаются типы .com, .exe, .obj, .sys, .rom. Все это дает возможность написания сверхкомпактных программ, использующих один (модель памяти tiny) или два (small) сегмента (возможно создание симбиоза .com-файлов — первый файл будет присоединен к другому таким образом, что второй файл получит управление только после выхода из процедуры main() первого), а также системных драйверов DOS. Есть совместимость со стандартным форматом линкера для использования функций C-- в программах на Си, Паскале и ТурбоПрологе. Допустимо, кстати, создание кода для экстендеров ROM-BIOS;

2. Присутствует возможность создания файлов с 32-битным кодом для защищенного режима DOS (для 32-разрядной программы можно выбрать DOS-экстендер (WDOSX, DOS4GW) или отказаться от него). Допустимые типы файлов — .exe и .obj. Допустима модель памяти flat (в ней существует только один сегмент, и этим она похожа на модель tiny), но для доступа к данным используются 32-разрядные смещения. Модель flat типична для 32-битных программ под DOS и Windows. Также генерируемые .exe-файлы могут поддерживать несколько сегментов.

3. Можно создавать файлы под Windows с 32-битным кодом для защищенного режима. Допустимые типы — .exe (.exe-файлы формата PE) и .dll. Допустима модель памяти flat. Наряду с оконными приложениями и динамически подключаемыми библиотеками, компилятор может генерировать и консольные приложения Windows. И еще, как известно, в программах под Windows есть заглушка, называемая stub, управление которой передается при запуске программы из "чистого" DOS. Обычно stub выводит сообщение о том, что эту программу надо запускать в среде Windows. Вы можете вместо стандартной заглушки использовать свою, т.е. имеется возможность создавать исполняемые файлы, работающие как под DOS, так и под Windows.

4. Многочисленная оптимизация по скорости исполнения и по размеру генерируемого целевого кода. Применение всего множества машинных команд процессоров 8086/88, 80186, 80286, 80386, 80486, Pentium I-III.

5. Наличие очень неплохой программной оболочки Phoenix Work Bench (Copyright (c) Keeper 2000, E-mail: hd_keeper@mail.ru), обеспечивающей различные удобства в работе. Что она может дать нам? Привычный интерфейс a la Turbo Vision, достаточно удобный многооконный текстовый редактор, замечательную справочную систему по процедурам и функциям языка C-- и встроенному ассемблеру. Обеспечиваются контекстные подсказки. Дополнительные "вкусности": подсветка синтаксиса C-- цветом; встроенный калькулятор программиста, поддерживающий различные системы счисления; таблица символов; возможность настроить все цвета по своему усмотрению. Разумеется, оболочка и вся справочная информация полностью на русском языке. На [3] выложена новая 2.4 версия IDE для C-- (файл wbr.zip размером 119 Кб) и справочник-документация по C-- на русском языке для использования в IDE (файл wb_help.zip размером 240 Кб).

6. Развитый аппарат параметров. Дело в том, что компиляторы Паскаля и Си/Си++ для передачи параметров процедурам и функциям используют специфическую "модель вызова", которая заключается в получении параметров с помощью адресации через регистр базы BP (EBP). Для локальных переменных и рекурсии это, безусловно, очень здорово, но операции по формированию и удалению стекового кадра "кушают" память и такты. Для некоторой оптимальности в процессорах 80x86, начиная с 80286, появились инструкции enter и leave, но этого ой как мало. Механизм все равно довольно громоздкий, и не является максимально подходящим для быстрой передачи параметров всем процедурам (стек — потеря эффективности). Что может предложить нам C--? Он позволяет передавать параметры прямо в регистрах! Замечательное решение! Помимо этого, есть возможность применить вызов стековых процедур как в стиле Си, так и в стиле Паскаль.

7. Встроенная вещественная арифметика на основе математического сопроцессора (тип float). Этому типу соответствует действительное число одинарной точности FPU. Использование операций над числами с плавающей запятой требует обязательного наличия сопроцессора, причем не только для выполнения скомпилированных программ, но и для работы самого компилятора. Сейчас C-- поддерживает основные действия над переменными: сложение, вычитание, умножение и деление, а также инкремент (var++), декремент (var--), смену знака (-var) и обмен значениями (var1><var2). Остальные математические операции реализованы во внешних библиотеках. Вы можете возразить, что это есть в любом Си, и я отвечу: да, но нет в ассемблере!

8. Возможность явного использования флагов процессора в инструкциях (в т.ч. и в качестве результатов функций), а отсюда — высокая эффективность программного кода. Архитектура C-- получилась более эффективной и адекватной машинному коду, чем существующие, именно за счет возможности явного использования стека, флагов и распределения регистров. Не есть ли это путь к созданию идеального императивного языка программирования? Ведь флаги, регистры и стек есть практически в любом процессоре, и практически любой ЯВУ этот факт игнорирует, кроме, возможно, самых зачаточных действий типа переменных register в Си или явных вставок в тело программы на ассемблере...

9. В C-- пока еще слабовата работа с указателями, но присутствует развитая типизация данных — имеется семь типов переменных. Причем для работы с типами float, dword и long компилятор использует 32-разрядные целочисленные команды (это позволило намного ускорить вычисления), следовательно, для их выполнения нужен процессор не хуже 80386. В табл.1 приведены размер и диапазон предоставляемых величин каждого типа.

Также для некоторой совместимости с языком Си введены зарезервированные слова short, signed, unsigned. Тип int в 32-разрядном режиме имеет размер 32 бита, что отражено в таблице предупреждающим знаком "!". ВНИМАНИЕ! Составные математические операции выполняются в том порядке, в котором они записаны (приоритет всех операций равный, выражения вычисляются слева направо), невзирая на правила арифметики! — тоже очень здравое решение, т.к. даже опытные программисты иногда становятся жертвой неправильно взвешенных приоритетов.

Табл. 1

Тип	Размер, байтов	Используемый регистр	Диапазон значений (dec)	Диапазон значений (hex)
byte	1	AL	0...255	0x00...0xFF
word	2	AX	0...65535	0x0000...0xFFFF
dword	4	EAX	0...4294967295	0x00000000...0xFFFFFFFF
char	1	AL	-128...127	0x80...0x7F
int	2/4	AX	-32768...32767 (!)	0x8000...0x7FFF (!)
long	4	EAX	-2147483648...2147483647	0x80000000...0x7FFFFFFF
float	4	EAX	-3,37E38...+3,37E38	0xFF7FFFFF...0x7FFFFFFF

10. Наряду с использованием локальных и глобальных, а также динамических переменных, есть возможность применить в своих задачах все оперативные, индексные и сегментные регистры процессора, динамическую память, указатели, массивы, структуры, объединения и битовые поля.

11. Большая (по сравнению с ассемблером) наглядность и легкость модификации кода, его сопровождение. Например, можно наглядно записывать короткие, ближние и дальние (без)условные конструкции; необходимые циклы транслируются непосредственно в высокоэффективные машинные команды цикла. Заметьте, для этого вовсе не обязательно прибегать к встроенному ассемблеру. Помимо большого количества ярко выраженных средств для оптимизации (динамический код — inline-вставки и макросредства, модель вызова fastcall и т.д.), наиглавнейшая приятность: язык не запрещает "оптимизацию программистом" — самый мощный из всех известных мне способов оптимизации.

12. Ну, и наконец, проект Sphinx C-- — полностью некоммерческий :-).

Еще о достоинствах Sphinx C--. Во встроенном ассемблере можно использовать полное множество команд MMX плюс все инструкции математического сопроцессора, вместе с тем не возбраняется писать программы и для устаревших процессоров. Есть возможность применения такого мощного средства как рекурсия. Допустим любой, сколь угодно большой размер проекта, чего не было раньше в C--. Получение ассемблерного листинга указанных фрагментов программы. Очень эффективные и проверенные библиотеки для использования их в собственных проектах (о них мы поговорим во второй части статьи).

Типы кода

Объявление процедур и функций введено в язык для того, чтобы сообщать компилятору о типе возврата из процедур, способе передачи параметров и их числе. Итак, какие возможности может предоставить программисту C-- для типизации кода?

Стековые процедуры. В C-- введено понятие стековых процедур, которые используют тип передачи параметров через стековый кадр. По умолчанию они объявляются при помощи идентификатора, содержащего по крайней мере один символ строчных букв. Параметры для стековых процедур могут иметь любой тип и передаются в соответствии с выбранной моделью вызова. Если процедура не имеет объявления, то компилятор не следит за числом и типом передаваемых параметров. В этом случае появляется свобода в их описании, но вы должны осознавать все последствия неверного задания параметров.

Регистровые процедуры определяются по умолчанию при помощи идентификатора, который не содержит символов строчных букв, или с помощью ключевого слова fastcall явным указанием на то, что это регистровое определение. Параметры (если они есть) передаются через регистры. Регистровые процедуры могут иметь не более 6 параметров. Если в процедуре сделать объявление порядка и типов используемых регистров, то возможно произвольное использование регистров.

Для того чтобы использовать регистровую процедуру как макрокоманду, она должна быть объявлена как динамическая.

Динамические процедуры — процедуры, которые определены, но вставляются в код программы только если есть обращение к ним. Определение динамической процедуры начинается с символа двоеточия. В динамических регистровых процедурах, которые будут использоваться как макрокоманды, нельзя использовать команду return() и локальные параметры.

Inline-процедуры. Ими могут быть динамические процедуры, которые можно использовать как макросы. Но, в отличие от макросов, inline-процедуры при включенной оптимизации на скорость автоматически вставляются в код, а при оптимизации на размер делается вызов их как динамических процедур.

Макрокоманды — просто регистровые процедуры, код которых вставляется всякий раз, когда происходит вызов. Символ "@", помещенный перед именем регистровой процедуры, указывает компилятору на то, что код должен быть вставлен в тело программы, а не вызван с помощью CALL. В качестве макрокоманд могут быть использованы только регистровые процедуры, объявленные как динамические.

Interrupt-процедуры — это процедуры обработки прерываний. Они не сохраняют никаких регистров автоматически, и никакие регистры сами по себе не загружаются перед передачей управления обработчику прерывания. Следовательно, сохранение значений регистров в стеке и последующее их восстановление, а также загрузка регистра DS нужным значением — на вашей совести. При завершении процедуры прерывания будет автоматически сгенерирована инструкция выхода из обработчика прерывания IRET.

Модели вызова

Сейчас C-- поддерживает четыре модели вызова определений: cdecl, pascal, stdcall и fastcall. Вот вкратце характеристики этих типов.

Cdecl — обычно применяется в языке Си. Характеризуется тем, что параметры передаются в порядке, обратном их записи. Процедуры заканчиваются командой RET. Освобождение стека от параметров происходит в том месте, откуда была вызвана процедура. Обычно это делается командой ADD SP,пппп. Т.е. компилятор точно знает, сколько и каких параметров вы передаете в данном случае, и соответственно освобождает стек после завершения процедуры. Это очень удобно для процедур с переменным числом параметров (например, определений типа printf). Чтобы компилятор воспринял стековую процедуру как процедуру в Си-стиле, первые два символа в имени процедуры должны быть "c_". Например: test(); — компилятор будет считать процедурой в стиле pascal, c_test(); — уже будет процедурой в стиле cdecl.

Pascal — этот тип вызова предполагает, что параметры передаются в том же порядке, в котором они записаны. Освобождение стека от параметров производит сама вызываемая процедура. Раньше C-- делал вызовы стековых процедур только в этом стиле. Преимуществом модели является компактность и более простой механизм генерации кода. К недостаткам, а соответственно, и преимуществам Си-стиля, можно отнести жесткую привязку паскалевских процедур к числу и типу передаваемых параметров (попробуйте при вызове процедуры в стиле pascal опустить один параметр — получите 100% зависание).

Stdcall — эта модель является гибридом первых двух. Параметры передаются в порядке, обратном тому, в котором они записаны, а освобождение стека от параметров производится в самой вызываемой процедуре.

Fastcall — самая быстрая модель вызова. Предполагает, что параметров нет, или их передача производится через регистры. Тем самым отпадает необходимость в освобождении стека от парамотров.

Примеры программ на C--

Давайте-ка будем учиться программированию замечательного кода на замечательном языке :-). Старый и добрый "Hello, World!" на C-- для DOS (.com-файл размером 36 байтов — и это вовсе не предел!):

```
#jumptomain NONE
#include "WRITE.H--"

main() {@WRITESTR("Hello World!\n");}
```

А вот для Windows:

```
#pragma option w32
#include "windows.h--"

void main(){
 MessageBoxA(0, "Hello World!", "C-- hello", MB_OK);
}
```

Простенько и со вкусом. Размер выходного кода этого оконного приложения — 1 Кб, Delphi "отдыхает" :-). Так же легко возможна генерация динамически линкуемых библиотек для Windows.

Рассмотрим мини-демо Георгия Ломсадзе (Wega) "Mirror kaleidoscope". С одной стороны, вроде бы, и не очень уместный здесь пример кодинга именно на C--, т.к. в этой интро много ассемблерных вставок, но с другой — показывается использование C-- в реальных условиях. А реально ассемблер приходится применять довольно часто, хотя, уверен, можно написать очень мощную программу вообще без обращений ко встроенному ассемблеру. По словам автора, тут все очень просто: "Если разобраться, то весь этот узорчик генерируется элементарнейшим образом — через всем известный алгоритм "прыгающего мячика", который был мною просто малость извращен."

Начнем со служебных процедур, находящихся в файле .h— (тип библиотечного модуля в C--, отличается от "сишного" .h тем, что включает не только прототипы объявлений, но и реализующий их исходный код). Практически все процедуры здесь базируются на ассемблерных построениях.

```
Файл: mirkldsp.h--
// "Mirror kaleidoscope" intro.
// (c)2001 Wega (2:5055/141.10@FidoNet)

// Текстмод
:fastcall TextMode()
{
 AX=03h; $int 10h
}
```

Динамическая регистровая процедура TextMode при обращении к ней как к макрокоманде займет в теле программы 5 байтов. В C-- по умолчанию, если имя процедуры написано большими буквами, считается, что эта процедура имеет тип вызова fastcall — очень интересное решение, но, увы, не способствует должной наглядности в применении идентификаторов. Поэтому если вы хотите изменить модель вызова процедуры на любую другую, то эту процедуру надо обязательно объявить с указанием желаемого типа вызова, как это и сделано в данном случае.

```
// Графмод
:fastcall GraphMode()
{
 AX=13h; $int 10h
}

// Вывод видеобуфера
:fastcall PutVideoBuf()
{
 $push DS,ES
 $push ES
 $pop DS
 ES=0A000h;
 SI=0;DI=0;
 CX=320*200/2;
 $rep $movsw
 $pop ES,DS
}
```

Естественный прием — видеоизображение сначала создается в видеобуфере, а потом быстро перебрасывается в экранный сегмент и отрисовывается. Обратите внимание, если бы кусок $push ES; $pop DS был критичным по времени исполнения, стоило бы заменить его на менее компактное, но более быстрое $mov AX, ES; $mov DS, AX. Но в данном случае это не нужно, т.к. код практически одноразовый, к тому же, оптимизируется в основном по размеру.

```
// Очистка видеобуфера
:fastcall ClrVideoBuf()
{
 DI=0;AX=0;
 CX=320*200/2;
 $rep $stosw
}
```

Естественно, конструкция CX=320*200/2 будет оптимизирована до CX=32000. Впрочем, такую оптимизацию можно и отключить.

В последних версиях компилятора есть возможность описывать ассемблерные куски блочно в виде asm {}, это иногда удобней, чем через знак $.

```
// Ожидание перерисовки экрана видеоадаптером
:fastcall WaitRetrace()
{
 asm{
  mov DX,03DAh
V1:
  in AL,DX
  and AL,08H
  jnz V1
V2:
  in AL,DX
  and AL,08H
  jz V2
 }
}
```

Эта процедура ожидает, пока электронный луч дисплея отрисует все точки изображения. Делается это для того, чтобы синхронизировать вывод и избежать нежелательного мерцания или одновременного появления на экране двух различных фреймов. Кстати, это как раз тот случай, когда применение метки в программе оправдано. Этот участок кода можно переписать чисто на C-- с использованием оператора цикла do {} while, но это здесь выглядело бы неуклюже и не соответствовало бы духу программы :-).

```
// Вывод пиксела с зеркальным отражением
// word x, y; byte color
:fastcall PutPixels(AX,BX,CL)
{
 SI=BX<<6;
 AX+=SI;
 SI=BX<<8;
 SI+=AX;
 ESBYTE[SI]+=CL;
 DI=64000-SI;
 ESBYTE[DI]+=CL;
}
```

Насколько нагляднее, чем на ассемблере, выглядят эти конструкции! Здорово!

```
// Начало
:fastcall Begin()
{
 @GraphMode();

 // Сегмент видеобуфера
 AX=DS;
 AX+=4096;
 ES=AX;

 @ClrVideoBuf();
}
```

```
// Конец
:fastcall End()
{
 @TextMode();
 AX=0; $int 16h
}
```

Перейдем непосредственно к телу интро. Если рассмотренная библиотека — в основном ассемблер, то это — чистый C--! Чистый-то чистый, да не совсем ;).

```
Файл: mirkldsp.C--
// «Mirror kaleidoscope» intro.
// (c)2001 Wega (2:5055/141.10@FidoNet)
// Compiled on SPHINX C-- Compiler v0.236.7 (Jan 28 2001)

#usePentium
#codesize
#resize FALSE
#jumptomain SHORT
#pragma option X

byte COPYRIGHT[7]={'(','c',')','W','e','g','a'};

#include "mirkldsp.h--"

main()
{

 @Begin();

 BP=0;

 do{

 AL=BP;AH=0;BX=0;CX=1;DX=CX;
   SI=1193;
   loop(SI){
    AX+=CX;BX+=DX;
     DI=199;
   IF(AX==319)||(AX<1)-CX;
   IF(BX==DI)||(BX<1)-DX;

   $pusha
    CX=CX|BP;
    $pusha
     PutPixels(AX,BX,CL);
    $popa
   DI-=BX;BX=DI;
   PutPixels(AX,BX,CL);
   $popa
  )

  BP-=2;

  SI=63999;
  loop(SI){
   ESBYTE[SI]-;
  }

  SI=320;
  loop(SI){
   AL=9;
   ESBYTE[SI-1]=AL;
   ESBYTE[SI+63679]=AL;
   IF (SI<200){
    DI=SI*320;
    ESBYTE[DI]=AL;
    ESBYTE[DI+319]=AL;
   }
  }

  CL=2;
  loop(CL) @WaitRetrace();
  @PutVideoBuf();

  $in AL,60h
 }while(AL!=1); // Пока не ESC

 @End();
}
```

Сначала я хотел рассмотреть две мини-демо, написанные Георгием Ломсадзе, но объем журнальной статьи... Увы.

Эти интро были выбраны мною совсем не случайно. Стиль их написания можно с уверенностью назвать хорошим стилем (впрочем, есть мнение, что толковый программист напишет хорошую программу на чем угодно). Эти маленькие интрочки написаны действительно *профессионально*. К сожалению, они не участвовали ни в каком конкурсе, а очень зря!

Не включенное в статью интро Micro3D представляет собой демонстрацию в динамике сложных и красивых трехмерных объектов, построенных из пикселов. И все это втиснуто в 508 байтов! Wega пишет: "Все фигурки описаны простейшими формулами, которые я сочинял на ходу, перебором". Поверьте, Micro3D вполне достойна того, чтобы ее увидеть [7].

Подведем итоги

Итак, почему же Sphinx C-- — язык не для всех? Все просто.

Во-первых, многие программисты вообще не слышали о C--, а некоторые считают его чем-то вроде еще одного компилятора C++.

Во-вторых, для того чтобы полноценно использовать такого рода инструменты, надо все же довольно хорошо знать архитектуру и "железо" платформы, а также внутреннюю структуру используемой ОС.

Ну а в-третьих, сам автор последних версий компилятора говорит: "Не могу я лапшу людям вешать — говорить, что все хорошо, когда проблем море...". Никаких особенных глюков я при компиляции не заметил, хотя написал несколько мелких и один довольно большой проект. Но автору, как говорится, виднее.

Кто заставляет вас использовать C-- вместо своего любимого C++? Используйте его вместо ассемблера! Просто потому, что C-- умеет все, что и асм, а листинги программ выглядят намного лучше — проще, логичней и нагляднее.

Итак, резюмируем. Sphinx C-- ни капельки не устарел (да и вряд ли устареет в ближайшее время), развивается, а эффективное программирование все еще востребовано (хотя уже и не в моде?). Так вперед!

На сайте журнала [7], помимо исходных текстов упомянутых выше интрочек, выложена небольшая графическая библиотека для CGA (скроллинг и "декоративная" очистка экрана, вывод спрайтов и литер) из игрушки Bo(u)lder Dash © Владимир Мутель, переписанная мною на C--. Прилагается также небольшой пример ее использования.

Выражаю благодарность Георгию Ломсадзе за предоставленный для препарации :-) программный код, написанный им, Peter'y Cellik'y за прекрасное начинание, а также Михаилу Шекеру за превосходный компилятор, библиотеки, консультации по языку C-- и ценные идеи для написания статьи.

Следует особо отметить, что Михаил чутко прислушивается к багрепортам, замечаниям и пожеланиям по улучшению Sphinx C--, поэтому здесь нелишним будет его адрес: 2:5021/3.111@FidoNet, E-mail: sheker@mail.ru

Литература

1. К. Хилько. Компилятор C--. Обзор возможностей языка C--. Встроенные функции в C--. — Радиолюбитель. Ваш компьютер, 1998, №№9-11.
2. Электронная конференция fido7.ru.C--
3. http://sheker.chat.ru (зеркало: http://C--sphinx.narod.ru, http://www.dev0.de/cmm)
4. http://www.dcs.gla.ac.uk/~reig/C--
5. http://www.cse.ogi.edu/PacSoft/projects/C--
6. http://www.cminusminus.org
7. http://radio-mir.com

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

Случайные числа упрощают алгоритм

В данной статье под случайными числами понимается возможность получения псевдослучайной последовательности символов в заданном диапазоне значений. И только. Математические теории их получения и прочие вопросы не рассматриваются. Я попытался с помощью случайных чисел упростить алгоритм поиска файлов в OS DOS (Windows) и провести небольшое исследование поведения программы, обладающей таким алгоритмом.

Суть исследований

Сначала коротко о поиске файлов и каталогов в DOS (Windows). Как известно, для поиска файлов и каталогов используются два сервиса прерывания int 21h — FindFirst и FindNext (функции 4eh и 4fh). Для указания цели поиска следует передать атрибут файлов в регистре cx, указатель на путь до каталога поиска и маску в файла в виде строки (в регистрах ds:[dx]). Например:

```
MyPath db 'c:\tmp\*.bat',0
```

В этом случае поиск выполнится в каталоге c:\tmp, будут (или не будут) найдены все файлы с расширением "*.bat". Сначала для инициализации поиска следует вызвать сервис FindFirst, а последующие найденные файлы будут возвращатся после вызова FindNext. Папки (директории) ищутся аналогично — например, по маске "*.*" и атрибуту 16 (директория).

Результаты поиска DOS помещает в так называемую DTA (Data Transfer Area). К примеру, при запуске программы она по умолчанию находится в PSP программы (для com-файла по смещению 80h). Но, при желании, DTA можно переопределить, чтобы не портить содержимое PSP. В этой области, помимо имени найденного файла, находятся такие атрибуты файла как дата и время.

Таким образом, чтобы найти нужные файлы в подкаталогах различного уровня вложения, требуется вначале найти все каталоги первого уровня вложения, затем в каждом из них — подкаталоги второго уровня, ..., последнего уровня, и при этом в каждом подкаталоге искать нужные файлы.

Данная задача — не из легких. В простой реализации ее решением может быть использование очередей (динамические связанные элементы, реализуемые, например, в Паскале через указатели). Более сложной реализацией (но и более красивой!) является рекурсия (к примеру, процедуре передается в стеке стартовый каталог).

Я реализовал алгоритм, отличный от вышеприведенных, требующий только статического буфера длиной 256 слов (с запасом!). Возможно, есть и другие варианты алгоритмов, но по общей оценке все они грешат следующими недостатками:

- сложность и большой размер кода (даже на asm!);
- низкое быстродействие;
- самое главное — если нам нужно найти любой подходящий объект во множестве каталогов (не все, а первый попавшийся!), то удручает применение столь мощной "артиллерии" (выше) для решения достаточно простой задачи.

Где же и зачем может быть поставлена задача поиска ***любого первого*** объекта? Я для примера рассмотрел обычный файловый нерезидентный вирус, способный заражать exe-файлы.

Коротко о алгоритме таких вирусов. Сразу после запуска зараженной программы вирус получает управление, и у него есть незначительное время, чтобы выполнить задачу всего живого на этой планете — размножиться, т.е. чтобы выполнить закон Дарвина.

Время незначительное, потому что он не хочет быть похожим на операционную систему с ее сообщением "Install... Please Wait...". Можно схалявить, и посмотреть "под ноги" — в текущий каталог, откуда запустилась программа. Для этого просто нужно выполнить поиск файлов прямо здесь с маской "*.exe". Но так мы далеко не уйдем — максимум, что можно, это заразить все файлы в одной директории, и точка. Не говоря уж о всем диске и других драйвах... Конечно, если пользователи не будут постоянно копировать друг у друга файлы и вдобавок раскидывать их по всем своим папкам. Правда, в этом случае сектор поражения тоже будет невелик — пациенты психушки имеют ограниченное число контактов с внешним миром.

Таким образом, приходим к вирусу, который должен искать мишени в других местах (каталогах, дисках). Например, можно получить текущий диск, перейти в его корень, и заражать файлы, лежащие в корне. Или пойти дальше — поискать во всех каталогах корня exe-файлы одним из вышеперечисленных алгоритмов.

Первичная реализация другого подхода

Однако можно посмотреть на проблему не так прямолинейно. Ведь нам на самом деле нужен всего лишь ***любой случайный*** объект. Заразив его, наш код удваивается, а, следовательно, в следующий раз уже будет выбрано два случайных объекта, потом — четыре, и так по нарастающей.

Мной был реализован такого рода алгоритм. При начальной реализации я задал три случайно выполняющихся действия, которые вирус может запустить при своем старте:

- случайным образом сменить диск (логический);
- случайным образом перейти в верхний каталог (команда "cd ..");
- случайным образом найти подкаталог со случайным номером и перейти в него.

После выполнения одного или нескольких действий из этого списка, текущим каталогом для поиска становится в общем случае произвольный, но другой! каталог, возможно, на другом логическом диске.

Более подробное описание и первые результаты

Итак, необходимо определиться — как будет активизироваться каждое из трех действий.

Для начала необходимо задать случайное число (числа), чтобы разыграть вероятность каждого действия. Для этих целей был взят сервис того же int 21h — функция получения системного времени 2ch, которая возвращает в регистре dl достаточно случайное число сотых долей секунды.

Далее необходимо определиться со стратегией — с какой вероятностью будем выполнять каждое из действий. Почему не выполнять их с максимальным приоритетом (всегда пытаться сменить диск, выйти наверх и найти что-нибудь "под ногами")? Так мы сузим себе круг поиска и, самое главное — будем негибки к различным конфигурациям пользователя (скажем, у одного много подкаталогов, а другой все скинул в корень). С этой целью в программе задается три параметра вероятностей активизации:

- VarOfDrive(0...99) — вероятность перехода на другой диск;
- VarOfUpDr(0...99) — вероятность перехода в верхний каталог;
- VarOfDnDr(0...99) — вероятность поиска подкаталога с произвольным номером.

Итак, с вероятностью VarOfDrive начинаем искать другой диск в пределах, поддерживаемых DOS (0...31). Далее, не-

зависимо от результата предыдущего действия, с вероятностью VarOfUpDr выполняем команду "cd...", и с вероятностью VarOfDnDr, опять-таки независимо от предыдущих событий, ищем каталог с произвольным в разумных пределах номером (скажем, не более 24) "под ногами" и, если он найден, переходим в него. При этом в момент поиска подкаталога происходит сверка с DOS-временем файла в DTA, и если время и текущий номер поиска кратны 4, то каталог с таким номером досрочно признается годным.

Надо признать, что в первичной реализации я действовал скорее по наитию — параметры выбрал достаточно наобум: VarOfDrive = 33%, VarOfUpDr = 50% и VarOfDnDr = 50%. Причем последние оказались зависимыми друг от друга. Для розыгрыша во всех событиях бралось одно и то же число — всегда число сотых долей секунды, полученное при старте.

Тем не менее, испытания показали, что, будучи запущен из какого-либо каталога, вирус, на первый взгляд, очень активно "снует" по каталогам и дискам в поисках мишеней.

Стратегия с умом

Однако можно задаться вопросом — а насколько эффективно был сделан выбор вероятностей VarOfDrive = 33%, VarOfUpDr = 50% и VarOfDnDr = 50%, и что же будет со всеми-всеми программами на всех-всех дисках в общем случае? Заразит ли вирус(ы) их все или забьется в какой-нибудь каталог и по тем или иным причинам не сможет вырваться из него в силу неправильно выбранной стратегии? Если в случае вируса, который умеет заражать только в текущем каталоге, проверка тривиальна, то в нашем случае вот так все не проверишь... Не будешь же, в самом деле, последовательно обходить десятки а то и сотни каталогов в непонятно какой последовательности!

С целью получения хоть какого-то ответа на эти вопросы мною была написана специальная программа, моделирующая распространение вируса по заранее выбранной конфигурации дисков и каталогов.

Моделирование процесса размножения вируса

Для начала зададим параметры идеальной модели конфигурации дисков и каталогов в них:

- DriveNumber — число логических дисков;
- DirsPerDrive — число корневых каталогов на каждом диске;
- DirsLevel — уровень вложенности каталогов. Это означает, к примеру, что если задан уровень вложенности 2, то у корневого каталога может быть не более двух вложенных директорий. Например: C:\TEMP — корневой каталог, в нем есть две папки — DIR1 и DIR2 (C:\TEMP\DIR1 и C:\TEMP\DIR2), у папок DIR1 и DIR2 могут быть подкаталоги, но в этих подкаталогах не может быть других подкаталогов;
- LocDirs — сколько папок в любой директории, включая корневые;
- ExecutableProgs — число исполняемых модулей в любой папке (в корне у меня их нет!), которые можно заразить.

Будем задавать также различные параметры стратегии распространения вируса — все те же VarOfDrive,VarOfUpDr и VarOfDnDr.

Для моделирования поведения вируса и получения результатов общего заражения воспользуемся методом Монте-Карло, проще говоря — методом случайного розыгрыша событий. В данном случае можно было, конечно, попытаться вывести аналитическую зависимость количества заражаемых модулей от количества запуска произвольных программ, но данный способ представляется мне гораздо более сложным, чем тот, который описан ниже.

Метод Монте-Карло очень прост. Пусть, к примеру, есть монета, и мы хотим узнать, с какой вероятностью будет выпадать орел. Для большего интереса пусть монета кривая, и, скорее всего, будет не фифти-фифти, а что-то другое. Можно метнуться к талмудам с формулами кривизны поверхности монеты, зависимостями плотности воздуха от расстояния до поверхности земли — но куда проще кинуть монету раз сто и, запомнив число выпадений орла, определить вероятность его выпадения (разделив это число на сто).

В нашем случае будем делать вот что. Доопределим несколько параметров уже самого моделирования:

- MaxDays — число дней испытаний;
- ProgPerDay — число запусков произвольных программ в день;
- AverCounter — число проходов по всем дням (т.е. прогоняем весь процесс заражения AverCounter раз — прямо как с монетой).

Испытания будут заключаться в следующем. Пусть вначале все программы на всех DriveNumber дисках будут незараженными. Заразим произвольную программу. Потом будем выбирать случайную программу (имитируя ее запуск) ProgPerDay раз в день, и так MaxDays дней. Фактически, в одном проходе мы выполняем запуски программ ProgPerDay*MaxDays раз — так сделано просто для наглядности. Если случайно запущенная программа окажется зараженной, то мы можем имитировать активизацию вируса из конкретного каталога и диска. Модельный вирус, в свою очередь, получит случайное число (как бы досовские сотые секунды) в качестве параметра, и с ним запустит собственный генератор случайных чисел (в первичной реализации я не понимал необходимости такого генератора). Далее он случайным образом выполнит действия по смене диска и каталога и, если найдет незараженную программу, выполнит ее заражение. Таким образом, число зараженных программ будет увеличиваться.

Проведя один такой прогон по всем дням, будем записывать число зараженных программ за каждый день, добавляя это число к соответствующему числу зараженных программ за этот день в предыдущем прогоне. Проведя все прогоны, усредним полученные значения по числу прогонов и получим зависимость числа зараженных программ от числа дней существования вируса в нашей системе.

Таким образом, задав те или иные параметры системы, мы получим ответ, насколько эффективно были выбраны параметры стратегии в той или иной конфигурации системы.

Результаты моделирования

Как только я провел серию первых испытаний со стратегией, соответствующей первоначальной реализации вируса, то сразу стало ясно, насколько она оказалась неэффективна! Оказалось, что такой вирус практически ничего не заражает — менее 5% программ! Пришлось дополнить алгоритм стратегии следующими усовершенствованиями:

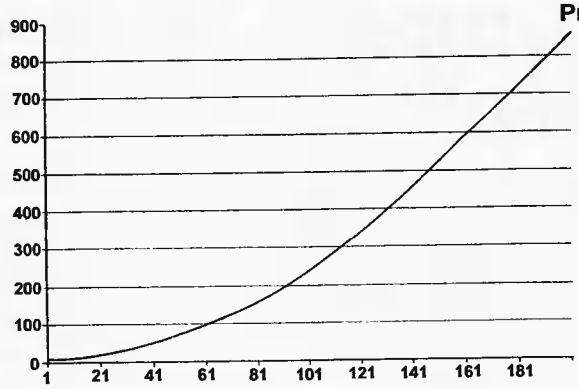
- в вирус был добавлен генератор псевдослучайных чисел;
- параметры VarOfDrive,VarOfUpDr и VarOfDnDr стали по-настоящему независимыми друг от друга — оказалось, что их взаимная зависимость серьезно сужает эффективность алгоритма случайного поиска!

На рис.1 изображена зависимость числа зараженных программ (в % × 100) от числа дней существования вируса в системе для следующих параметров:

- MaxDays = 200 — число дней испытаний;
- ProgPerDay = 50 — число запусков произвольных программ в день;
- DriveNumber = 8 — число логических дисков;
- DirsPerDrive = 12 — число корневых каталогов на каждом диске;
- DirsLevel = 2 — уровень вложенности каталогов;
- LocDirs = 2 — количество папок в любой директории, включая корневые;
- ExecutableProgs = 2 — число исполняемых модулей в любой папке.

Т.е. всего было выполнено 10000 запусков для 1344 про-

Рис. 1



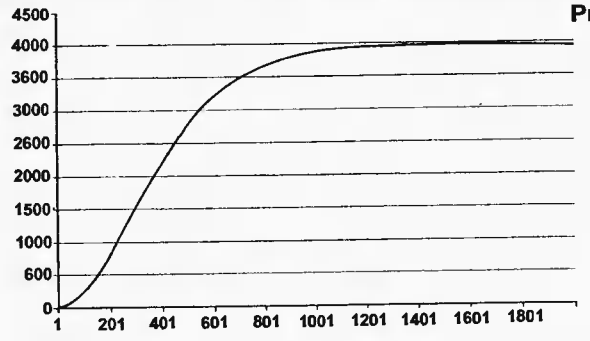
грамм. Каждая программа запускалась примерно 9 раз. Число каталогов в корневом каталоге рассчитывается по формуле: (LocDirs^(DirsLevel + 1) - 1)/(LocDirs-1) — фактически, сумма геометрической прогрессии.

Параметры стратегии были следующими:

VarOfDrive = 50%, VarOfUpDr = 50% и VarOfDnDr = 50%.

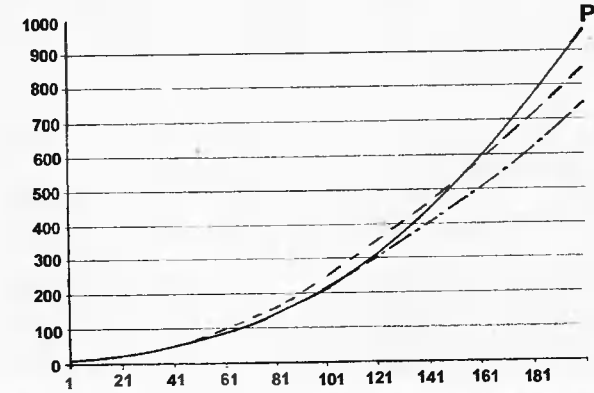
Далее я сразу обнаружил очень интересный эффект — если в системе запускать программы неограниченно долго (число запусков >> числа программ), то вирус никогда не сможет заразить все программы! По крайней мере, это проявлялось для вышеприведенных параметров, с той лишь разницей, что число дней запуска было увеличено до 2000 (рис.2). Насыщение наступает примерно на уровне всего 40% от всего числа программ. Причем повторюсь, первая зараженная программа выбиралась произвольно — а это означает, что режим насыщения наступает независимо от того, какая программа будет заражена первой!

Рис. 2



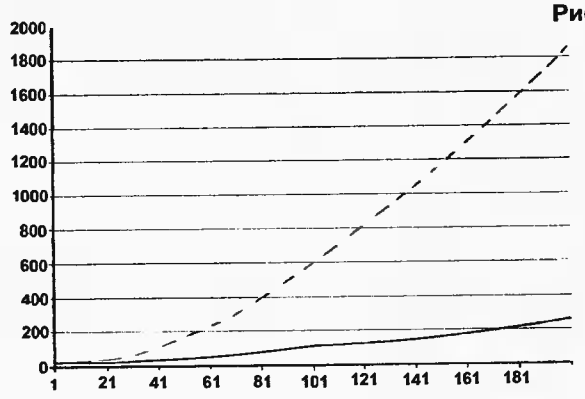
Далее я поэкспериментировал с параметрами стратегии. В одном случае (конфигурация системы прежняя) параметры стратегии были: VarOfDrive = 10%, VarOfUpDr = 50% и VarOfDnDr = 50% (неактивно меняем диски, но достаточно активно лезем в верхние и нижние каталоги), в другом — VarOfDrive = 50%, VarOfUpDr = 50% и VarOfDnDr = 50% (средне-активно меняем диски и каталоги), а в третьем — VarOfDrive = 90%, VarOfUpDr = 50% и VarOfDnDr = 50% (очень активно меняем диски). Результат представлен на рис.3. Вероятности смены диска: "—" — 10%; "– – –" — 50%; "—·—" — 90%.

Рис. 3



В следующем случае я исследовал на тех же параметрах системы что выгоднее: активно менять каталоги (VarOfDrive = 5%, VarOfUpDr = 90% и VarOfDnDr = 30%) или активно менять диски (VarOfDrive = 90%, VarOfUpDr = 10% и VarOfDnDr = 90%), т.е. активно менять текущий драйв и "расползаться" по подкаталогам. Результат — на рис.4 (вероятности смены диска, перехода в верхний каталог и вероятность перехода в нижний каталог: "—" — 5%, 90% и 30%; "– – –" — 90%,10% и 90%.

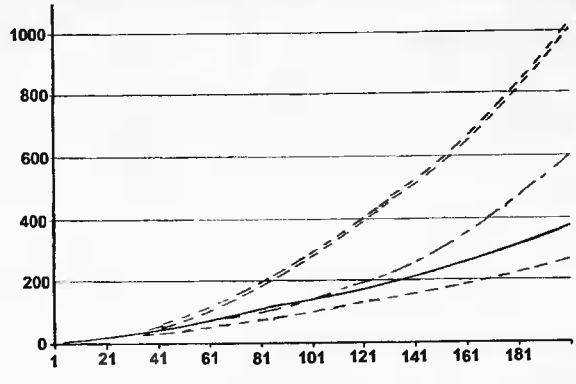
Рис. 4



Далее я решил посмотреть поведение вируса для конфигурации с малым количеством логических дисков и относительно большим количеством подкаталогов. Параметры новой конфигурации:

- DriveNumber = 2 — число логических дисков;
- DirsPerDrive = 10 — число корневых каталогов на каждом диске;
- DirsLevel = 2 — уровень вложенности каталогов;
- LocDirs = 4 — количество папок в любой директории, включая корневые;
- ExecutableProgs = 2 — число исполняемых модулей в любой папке.

Рис. 5



На рис.5 — то, что получилось для четырех различных стратегий размножения. Уровень вложенности равен 4, в каждом каталоге — по две директории. Вероятности смены диска, перехода в верхний каталог и вероятность перехода в нижний каталог: "===" — 5%, 20% и 90%; "—·—" — 5%, 30% и 60%; "—" — 5%, 90% и 90%; "– – –" — 5%, 60% и 30%.

Немного о генераторе случайных чисел

Для моделирования мною был выбран генератор псевдослучайных чисел из книги С.В.Зубкова "Assembler — язык неограниченных возможностей". Это так называемый сдвиговый генератор в простейшей реализации для получения псевдослучайных чисел в размере одного байта.

От редакции. Исходные тексты программ к статье выложены на `http://radio-mir.com`

Боремся — с однообразием

А.КОРБИТ,
г. Минск, БГУИР,
каф. ВМиП.

При создании приложений в среде C++ можно разнообразить внешний вид рабочего окна приложения за счет использования окон не только прямоугольной формы. Напомню, что при работе с прямоугольными областями окна используют объект типа **CRect**, который и определяет окно с прямоугольной рабочей областью. Тип **CRect** порождается от структуры типа **RECT**, которая объявлена следующим образом:

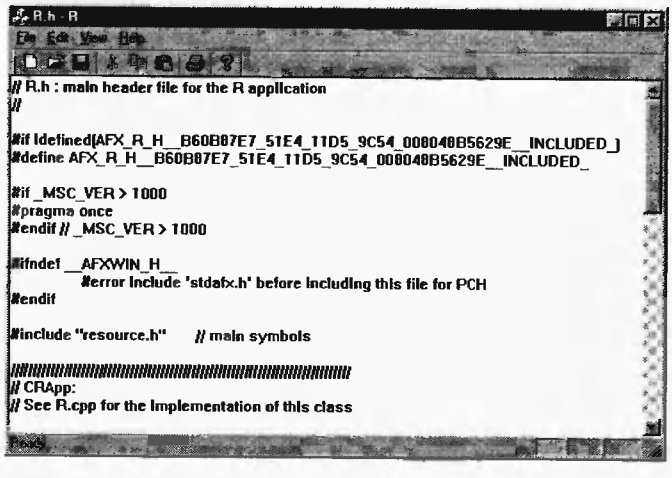
```
typedef struct tagRECT {
   LONG left;
   LONG top;
   LONG right;
   LONG bottom;
} RECT;
```

Для явного использования данного типа в конструкторе класса окна Win32Application-приложения нужно создать объект типа **RECT** и явно задать значения его полей. Например:

```
Class CMyWnd:CFrameWnd
{
public:
CMyWnd();
...
};

CMyWnd::CMyWnd()
{
RECT r;
r.top = 10; r.left = 10;
r.bottom = r.right = 200;
Create(NULL, " A Small Window", WS_OVERLAPPEDWINDOW, r);
}
```

Рис. 1



Приложение создаст стандартное окно, показанное на рис.1, верхний угол которого имеет координаты 10, 10, а нижний правый — 100, 100.

Если же SDI-окно создано при помощи AppWizard, то для установки собственных размеров объекта необходимо в файле MainFrm в начале метода OnCreate вызвать метод **SetWindowPos**. Этот метод изменяет размеры окна и задает его позицию на экране.

Декларация метода имеет следующий вид:

BOOL SetWindowPos(const CWnd* pWndInsertAfter, int x, int y, int cx, int cy, UINT nFlags);

Данный метод возвращает ненулевое значение — если функция отработала успешно, и ноль — в противном случае.

круглое окно!!!

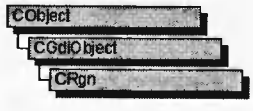
Первый параметр метода — **pWndInsertAfter** — идентифицирует CWnd-объект, после которого будет размещен текущий CWnd-объект. Поскольку в данном случае устанавливаются размер и положение текущего SDI-объекта, этот параметр нужно установить в NULL.

Параметры **x** и **y** определяют новую позицию верхнего левого угла окна, а параметры **cx** и **cy** — новые ширину и высоту окна.

Последний параметр — **nFlags** — это набор флажков, определяющих режимы отображения окна. Этот параметр может быть комбинацией нескольких значений:

- SWP_DRAWFRAME — создать фрейм (рамку) вокруг окна;
- SWP_HIDEWINDOW — скрыть окно;
- SWP_NOACTIVATE — не активизировать окно;
- SWP_SHOWWINDOW — отобразить окно и т.д.

Например: **SetWindowPos(NULL, 100, 100, 300,100, SWP_SHOWWINDOW)**



Для создания непрямоугольных рабочих областей используют понятие **регион**. Это, как и объект типа CRect — **область окна**. Чем же регион отличается от класса **CRect**?. Только тем, что **CRect** всегда прямоугольник, а регион может принимать разные формы, например, эллипсоидные.

Регион является объектом типа **CRgn**. Этот класс наследуется непосредственно от класса **CGDIObject** — базового класса интерфейса различных устройств графики (GDI), таких как побитовые изображения, области, кисти, перья, палитры, шрифты и т.д. От данного класса непосредственно объектов не создают, а используют классы, производные от него.

Класс **CRgn** инкапсулирует область интерфейса графического устройства, которая может быть или эллиптической, или многоугольной областью в пределах окна.

Для работы с регионом используют функции класса **CRgn**, при этом действия функций класса CDC за границей региона автоматически блокируются. Функции класса **CRgn** создают, а также позволяют изменять или получать информацию об объектах, для которых они вызываются.

Регион можно получить через функцию **GetWindowRgn**. Этот метод вызывается для того, чтобы получить регион окна, который определяет ту область в пределах окна, где операционной системе доступна, например, перерисовка окна. При этом операционная система не отображает область окна, которая лежит вне его региона.

Метод принадлежит классу CWnd, и возвращает величину, определяющую тип области, которую функция получает.

Возможные значения возвращаемого значения:

- NULLREGION — пустой регион;
- SIMPLEREGION — простой прямоугольник;
- COMPLEXREGION — многоугольник;
- ERROR — ошибка, региона нет.

Единственный параметр метода hRgn типа HRGN — дескриптор региона, который получает данный метод.

Для установки региона используют метод класса **CWinApp:** — **SetWindowRgn**. Этот метод объявлен следующим образом:

int SetWindowRgn(HRGN *hRgn*, **BOOL** *bRedraw* **= FALSE);**

Первый параметр hRgn — дескриптор региона, устанавливаемого в области окна. Т.е. после успешного вызова метода SetWindowRgn система получает регион, вид которо-

го определяет дескриптор hRgn. Для получения овального региона в качестве первого параметра воспользуемся методом, задающим эллиптическую область. Объявление данного метода:

```
BOOL CreateEllipticRgn( int x1, int y1, int x2, int y2 );
```

создает эллиптическую область, которая создается для объекта типа CRgn. Эта область определяется связанным прямоугольником, который задается координатами верхнего левого (*x1*, *y1*) и правого нижнего (*x2*, *y2*) углов. Максимальный размер области — 32767 на 32767 логических единиц или до 64 Кб оперативной памяти. Данный метод возвращает ненулевое значение в случае успешного создания области и нулевой результат в случае возникновения ошибки.

Второй параметр — *bRedraw* (логического типа) — определяет, перерисует ли система окно после установки региона. Если значение этого параметра — "**истина**", то система сделает перерисовку окна, в противном случае перерисовка не производится. Так как регион будет отображен на экране, то второму параметру нужно задать значение "**true**". В случае успешной установки региона метод возвращает положительное значение, а при возникновении ошибки — нулевое значение. При вызове данного метода окну посылаются два сообщения — **WM_WINDOWPOSCHANGING** и **WM_WINDOWPOSCHANGED**. Окно, для которого устанавливается регион, получает их через функцию WindowProc. Координаты устанавливаемого региона определяются относительно верхнего левого угла окна, а не по отношению его клиентской части.

Для вызова данных методов воспользуемся указателем, который возвращает метод, объявленный как `CWinApp* AfxGetApp( );`.

Указатель, возвращаемый этой функцией, используют для того чтобы получить доступ к информации о приложении, например — код сообщения или основное окно.

Итак, мы создаем приложение как **MFC AppWizard**. Пусть оно будет **SDI**, и имя у него будет **Tt**.

1. В классе приложения заведем переменную типа "регион". Переменная типа "регион" объявляется в файле **Tt.h** — основном заголовочном файле класса приложения.

```
class CTtApp : public CWinApp
{
public:
      HRGN rgn;
      CTtApp();
........
};
```

2. Сохраним регион окна при инициализации окна. Для этого поместим файл **Tt.cpp** в самый конец метода **InitInstance()**, отвечающего за инициализацию приложения.

```
/////////////////////////////////////////////////////////////////////////
// CTtApp initialization

BOOL CTtApp::InitInstance()
{
      .......
      GetWindowRgn(m_pMainWnd->m_hWnd, rgn);
      return TRUE;
}
```

Вот теперь окно можно будет восстановить.

3. Обработчики щелчка левой и правой кнопками мыши размещаем в файле **TtView.cpp**. В начале файла в карту сообщений необходимо вставить макросы **ON_WM_RBUTTONDOWN()** и **ON_WM_LBUTTONDOWN()**.

В самый конец файла после фразы "**// CTtView message handlers**" добавляем тексты обработчиков событий.

Код при нажатии на правую кнопку мыши:

```
void CTtView::OnRButtonDown(UINT nFlags, CPoint point)
{
 AfxGetApp()->m_pMainWnd->SetWindowRgn
(CreateEllipticRgn(40, 40, 300, 200),true);
 CView::OnRButtonDown(nFlags, point);
}
```

Код при нажатии на левую кнопку мыши при востановлении:

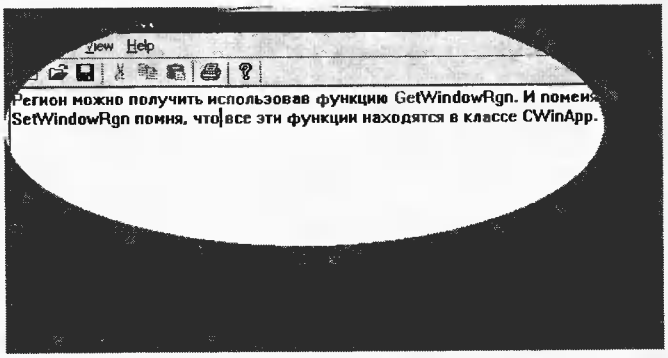
```
void CTtView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CTtApp* tt=(CTtApp*)AfxGetApp();
    AfxGetApp()->m_pMainWnd->SetWindowRgn(tt->rgn,true);
    CView::OnLButtonDown(nFlags, point);
}
```

4. В файл **TtView.h** сразу после декларации конструктора класса **CTtView()** добавляем прототипы этих обработчиков:

```
CTtView();
afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
DECLARE_DYNCREATE(CTtView)
```

В результате всех этих действий, запустив наше приложение, мы получаем простейший графический редактор со стандартным SDI-окном, приведенным на рис.1. Щелчок правой кнопкой мыши на клиентской части окна приведет к созданию региона, показанного на рис.2, а щелчок левой клавишей мыши в области региона вернет привычный вид SDI-объекта.

Рис. 2



И в заключение отмечу, что кроме метода, который создает регион типа окружности, в классе CRgn есть следующие методы для создания регионов:

- CreateRectRgn — инициализирует CRgn-объект с прямоугольной областью;
- CreateRectRgnIndirect — инициализирует CRgn-объект с прямоугольной областью, определенной структурой RECT;
- CreateEllipticRgnIndirect — инициализирует CRgn-объект с эллиптической областью, определенной структурой RECT;
- CreatePolygonRgn — инициализирует CRgn-объект с многоугольной областью. Система замыкает многоугольник автоматически и, в случае необходимости, достраивает линию из его последней вершины в начало многоугольника;
- CreatePolyPolygonRgn — инициализирует CRgn-объект с областью, состоящей из серии закрытых многоугольников. Многоугольники могут быть непересекающимися или могут перекрыться;
- CreateRoundRectRgn — инициализирует CRgn-объект с прямоугольной областью и с закругленными углами.

В.ВАСИЛЕНКО,
г.Свердловск, Луганской обл.

Устранение некоторых неисправностей лазерного принтера HP Laser Jet 1100

HP Laser Jet 1100 — это лазерный принтер формата А4 со скоростью печати 8 листов в минуту. Он с успехом заменяет более ранние модели этого семейства (HP Laser Jet 5L, HP Laser Jet 6L), превосходя их по скорости печати. Кроме того, он может комплектоваться съемной приставкой (модель 1100A), позволяющей выполнять сканерно-копировальные работы. Ресурса картриджа обычно хватает на 2500 копий (при 5% заполнении листа тонером, что соответствует текстовому режиму).

Опишем некоторые неисправности, которые могут иметь место в этом принтере. Но сначала разберемся с индикацией на передней панели принтера (рис.1). Большой зеленый индикатор-кнопка 1 — это индикатор включения и кнопка распечатки теста, маленький зеленый индикатор 2 — индикатор готовности печати, маленький желтый индикатор 3 — индикатор наличия или отсутствия бумаги.

Рис. 1

Итак, неисправности и методы их устранения.

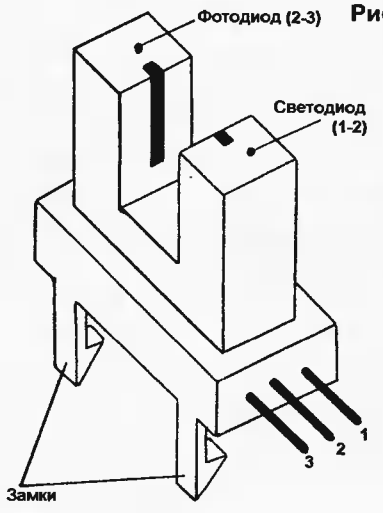
1. Принтер не реагирует на попытки что-то распечатать, при этом распечатать тест невозможно, мигает желтый индикатор 3.

Мигание желтого индикатора 3 означает, что либо произошел затор бумаги в принтере, либо неплотно вставлен (или отсутствует) картридж, либо неплотно прикрыта передняя дверца. Первым делом надо, вынув картридж, посмотреть, нет ли в принтере застрявших листков бумаги, и если есть — осторожно вынуть их. Это надо сделать аккуратно, стремясь не порвать лист, иначе придется вынимать и обрывки (наиболее частая причина заторов — использование слишком тонкой бумаги, наилучшие результаты получаются при работе с бумагой плотностью 60...80 г/м). Затем надо плотно вставить картридж. В нижней части картриджа имеются две контактные площадки, а на соответствующих местах в принтере — две пружинящие скобки из упругой проволоки. Между скобками и контактными площадками должен быть надежный контакт. Контакт может нарушаться из-за загрязнения контактных площадок тонером, появления жировых или окисных пленок на контактных пружинах или из-за ослабления их пружинных свойств. Загрязненные места необходимо протереть изопропиловым или этиловым спиртом. При ослаблении пружинящих свойств контактных скобок следует слегка отогнуть их навстречу контактным площадкам картриджа. Картридж может вставляться неплотно, в частности, и из-за того что этому препятствует подвижная шторка, закрывающая снизу фоторецепторный барабан (фирменные картриджи обычно имеют барабан голубого цвета). Поэтому, вставляя картридж, следует оттянуть подвижную шторку максимально вперед и вверх, задвинуть картридж до упора и отпустить шторку. После этого необходимо плотно прикрыть переднюю крышку. Передняя крышка в своей нижней части имеет выступ, который через систему рычагов нажимает микропереключатель на верхней плате принтера (в принтере имеются две платы: на нижней расположена микропроцессорная схема управления, на верхней — блок питания, схема обработки сигналов датчиков и остальные детали). Если картридж плотно вставлен, передняя дверца плотно прикрыта, а индикатор 3 все равно мигает, то либо неисправен микропереключатель на верхней плате принтера (что, однако, бывает редко — микропереключатель коммутирует малые токи и отличается высокой надежностью), либо загрязнены или неисправны оптопары в тракте движения бумаги.

В тракте движения бумаги расположены три оптопары (фотодиод — светодиод), две из них — на входе (когда в любой из входных лотков ставится бумага, отрабатывает первая из них, находящаяся в самом начале тракта движения) и третья — на выходе, после блока закрепления (печки). Загрязнение оптопар может иметь место из-за попадания частиц тонера или бумажной пыли (особенно если используется бумага некачественных сортов), причем наиболее вероятно загрязнение двух первых. Третья оптопара, расположенная на верхней плате принтера, окружена поролоновой прокладкой, что уменьшает вероятность загрязнения.

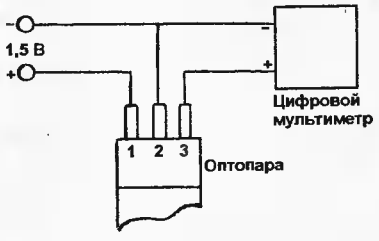
Для того чтобы добраться к микропереключателю или оптопарам, необходимо произвести разборку принтера. Порядок разборки следующий. Сначала необходимо снять небольшие декоративные накладки зеленого цвета в нижней части по бокам принтера. Для этого надо положить принтер на заднюю стенку (или набок) и со стороны донной части отжать отверткой или длинным тонким предметом пружинящие выступы — замки декоративных накладок. Две входные оптопары расположены на общем основании под входными лотками. Каждая из оптопар имеет вид, изображенный на рис.2.

Рис. 2



При подозрении на неисправность следует снять оптопары с основания (каждая оптопара имеет в своей нижней части небольшие пластмассовые выступы-замки, посредством которых она крепится к основанию, пространство между фото- и светодиодом может перекрываться подвижной пластмассовой шторкой) и проверить их. Сначала следует протереть щели в оптопарах изопропиловым спиртом или продуть их сильным потоком воздуха. Для этих целей можно использовать пылесос, включив его на реверс. Продувать оптопары прямо в принтере не рекомендуется — частицы тонера и бумажная пыль могут при этом попасть в оптическую систему принтера и загрязнить ее. Оптическая система расположена в верхней части принтера и конструктивно выполнена в виде плоского пластмассового коробка, в котором расположены лазерный излучатель, подвижная зеркальная призма с приводом и двигателем и система неподвижных призм и зеркал. Сверху оптическая система хорошо защищена от пыли, места, где расположены разъемы, дополнительно герметизированы небольшими поролоновыми прокладками. Но снизу имеется длинная узкая прямоугольная щель, через которую лазерный луч сканирует фоторецепторный барабан, создавая скрытое изобра-

Рис. 3



жение. Через эту щель пыль с потоком воздуха может попасть в оптическую систему. Поэтому при необходимости удалить пылевые загрязнения с внутренних деталей принтера необходимо использовать пылесос, включенный на всасывание, и мягкую кисточку.

Проверить исправность оптопары можно следующим образом. Необходимо собрать схему, изображенную на рис.3, подать на светодиод оптопары (выводы 1 — 2) напряжение не более 1,5 В (это важно!) с учетом полярности и прикоснуться щупами цифрового мультиметра (также с учетом полярности) к выводам 2 — 3 фотодиода. Мультиметр при этом используется в режиме измерения сопротивления. Сопротивление освещенного фотодиода составляет величину в несколько килоом, неосвещенного — несколько сот килоом. Впрочем, оптопара отличается высокой надежностью, и вероятность выхода ее из строя невелика.

2. Принтер "не видит" бумагу, постоянно горит индикатор 3.

В этом случае загрязнена или неисправна первая оптопара. Следует провести вышеуказанные операции. В нормальном состоянии, при загрузке бумаги в любой из лотков индикатор 3 должен погаснуть, а индикатор 2 — загореться.

3. Нет подачи бумаги, или подается сразу несколько листов.

В этом случае загрязнены подающий ролик и детали, соприкасающиеся с ним. Получить доступ к подающему ролику можно, вынув картридж. Подающий ролик расположен у дна загрузочных лотков. Конструктивно он выполнен в виде резинового эллипса, который с эксцентриситетом закреплен на пластмассовой оси и может быть снят без полной разборки принтера (для снятия ролика достаточно открутить два винта по бокам). Резиновую поверхность ролика надо протереть изопропиловым спиртом или теплой водой с мылом. Если после этого неисправность не исчезла, следует протереть и другие части тракта подачи, соприкасающиеся с подающим роликом. В некоторых случаях может потребоваться полная разборка принтера.

4. Снижена контрастность изображения.

Здесь может быть несколько причин:

- возможно, в драйвере принтера установлен режим economode (пониженный расход тонера);
- фоторецепторный барабан выработал свой ресурс;
- заканчивается тонер в картридже;
- загрязнена оптическая система.

Режим economode в принтере HP1100 имеет свои особенности. При работе в этом режиме внешние контуры знаков прорисовываются тонкой черной линией, а внутреннее пространство заливается более светлым серым цветом. В других моделях лазерных принтеров может использоваться другой способ организации этого режима, при этом и внешние контуры, и внутреннее пространство заливаются одним светлым серым цветом. При использовании первого способа текст получается более читабельным.

Если режим economode отключен, а текст бледный — причина либо в картридже, либо в оптической системе. Оптическая система может загрязниться не только частицами тонера или бумажной пылью, но и из-за того, что вблизи принтера курят. При подозрении на загрязнение оптической системы следует протереть ее детали (пластмассовые призмы, стеклянные зеркала и подвижную металлическую призму на валу двигателя) хлопчатобумажной тканью (не ватой!), смоченной теплой водой. Наиболее вероятная причина бледной печати — картридж. Фирма HEWLETT-PACKARD рекомендует использовать только фирменные картриджи для обеспечения высокого качества печати. Однако, для снижения стоимости печатной копии картриджи часто заправляют повторно (один или несколько раз). Существует целая индустрия, поставляющая тонер, фоторецепторные барабаны, ракели и другие детали.

Следует запомнить одно простое правило. В большинстве случаев фирменный картридж можно заправить еще три раза, т.е. теоретически можно на одном фоторецепторном барабане "откатать" 10000 копий. Однако при использовании некачественной бумаги (у которой повышенная абразивность, вызывающая ускоренный износ фоторецепторного слоя) и некачественного тонера (размер и форма частиц которого отличаются от тех же показателей фирменного тонера) фоторецепторный барабан изнашивается быстрее, и все большая часть тонера начинает уходить в бункер отработки. Поэтому реальное количество отпечатанных копий составляет меньшую величину. Качество печати с каждой последующей заправкой немного снижается, но в большинстве случаев это вполне приемлемо. Заправить три раза можно только тот картридж, в котором установлен фирменный фотобарабан (он, как правило, имеет голубой цвет). Множество фирм производят поставки так называемого ремонтного комплекта, состоящего из фоторецепторного барабана и ракеля (пластины, снимающей остатки тонера с барабана). Как правило, в этих комплектах фоторецепторный барабан имеет светло- или темно-зеленый цвет и несколько пониженный, по сравнению с фирменным, ресурс. Этого ресурса зачастую не хватает на четыре заправки, как в случае с фирменным картриджем. Поэтому при заправке или ремонте картриджа необходимо вести статистику. Общее количество отпечатанных копий и другую служебную информацию можно получить, нажав на короткое время кнопку-индикатор 1. Если при печати на листе присутствуют продольные (параллельные длинной стороне листа) более светлые полосы — это говорит о том, что в картридже заканчивается тонер. В этом случае следует слегка встряхнуть картридж в горизонтальном направлении несколько раз. Тонер при этом более равномерно распределится по магнитному барабану, и можно отпечатать с приемлемым качеством еще несколько копий.

5. Посторонние черные точки и кляксы на отпечатанном листе.

Здесь может быть несколько причин. Возможно, при заправке картриджа не был удален тонер из бункера отработки. При заправках картриджа вследствие износа поверхности фоторецепторного барабана все большая часть тонера идет в отработку. Если не удалять отработанный тонер из бункера, тот переполняется, и тонер начинает сыпаться на печатаемый лист и внутренние детали принтера. Тонер представляет собой мелкие гранулы, состоящие из ферромагнитных частиц, окруженных полимерной оболочкой. При прохождении листа с перенесенным с фоторецепторного барабана тонером через блок закрепления (печку), тонер расплавляется и крепко сцепляется с поверхностью бумаги. При этом нагревается не только печка, но и другие детали принтера. Если попавший на них тонер не будет сразу удален, он намертво сцепится с их поверхностью и придаст им неряшливый вид. Отработанным тонером нельзя заправлять картридж, он подлежит утилизации.

Другая причина — повреждение в отдельных местах поверхности фоторецепторного барабана. В этом случае черная точка периодически (3...4 раза) повторяется на одном листе на воображаемой линии, параллельной оси движения бумаги. Это повреждение можно обнаружить визуально, внимательно осмотрев поверхность барабана. Поврежденное место отличается цветом — если вся поверхность имеет равномерную голубую (зеленую) окраску, то поврежденное мес-

то — светло-желтую или белую. Повреждение такого рода может быть вызвано некачественной бумагой (продуктом переработки отходов). В такой бумаге могут находиться чужеродные вкрапления разного рода, вплоть до металлических. Так что не стоит экономить на бумаге.

Еще одна причина — загрязненность тонером рабочих поверхностей в блоке закрепления (печке). Блок закрепления расположен в передней части принтера и в обычном состоянии закрыт декоративной накладкой. Он состоит из пустотелого цилиндра из термопленки, внутри которого расположен нагревательный элемент и датчик температуры, и резинового вала. Бумага с налипшим тонером при выходе из картриджа проходит между термопленкой и резиновым валом ("лицом" к термопленке). Включается нагревательный элемент, изображение закрепляется, и дальше бумага подается в выходной лоток (или наружу, если используется бумага высокой плотности). Если производить печать на обеих сторонах листа, то расплавляется тонер и на той стороне листа, которая соприкасается с резиновым валом. Дело осложняется в том случае, если печатаются текст жирным шрифтом или насыщенное изображение. Часть тонера переходит с листа на резиновый вал. Со временем загрязняется и термопленка. Если обе поверхности сильно загрязнены, то наслоения тонера время от времени отрываются и переходят на бумагу, снижая качество изображения. В этом случае необходимо почистить резиновый вал и термопленку (желательно делать это после печати каждых 10000 копий). Для этого необходимо разобрать принтер и снять блок закрепления (он крепится двумя защелками по краям). Удалить наслоения тонера можно хлопчатобумажной тряпочкой, смоченной керосином. При этом надо быть внимательным — термопленка (она обычно темно-коричневого цвета) достаточно тонкая, и ее легко повредить.

6. Фрагментарное отсутствие изображения.

Если фрагменты с отсутствующим изображением лежат на оси, параллельной направлению движения бумаги, то это свидетельствует о повреждении термопленки в блоке закрепления. Повреждение термопленки может быть вызвано, в частности, неумелым устранением заторов. Если термопленка и резиновый вал в блоке закрепления загрязнены налипшим тонером, и произошел затор (лист бумаги остался при этом в блоке закрепления), вытаскивать лист надо очень осторожно. При возникновении затора тракт движения блокируется, и нагреватель блока закрепления отключается. Тонер, перенесенный на бумагу и налипший ранее на термопленку и резиновый вал, быстро остывает, и бумага прилипает к термопленке и резиновому валу. Если резко дернуть лист, можно повредить термопленку. Поэтому лист в этом случае следует очень осторожно и медленно вытащить по направлению движения бумаги. В некоторых случаях может понадобиться частичная разборка принтера.

При замене термопленки надо внимательно осмотреть поверхность нагревательного элемента и сам нагревательный элемент. Нагревательный элемент представляет собой узкую длинную керамическую пластину с нанесенной на ее поверхность пленкой сплава с высоким удельным сопротивлением (сопротивление нагревательного элемента в холодном состоянии — около 100 Ом). На противоположной стороне закреплен термодатчик, с помощью которого осуществляется управление нагревательным элементом. В держателе нагревательного элемента (непосредственно под элементом) расположен термопредохранитель. Он служит для предохранения нагревательного элемента от разрушения в случае выхода из строя схемы управления нагревателем. Может случиться так, что схема управления выйдет из строя, и на нагревательный элемент будет постоянно подаваться напряжение. Нагреватель разогреется до температуры выше обычной (рабочей), но при этом сработает термопредохранитель и прервет цепь питания. Температура срабатывания термопредохранителя выбрана несколько выше рабочей, так что в обычном режиме он не оказывает никакого воздействия на работу, и его сопротивление практически равно нулю.

При повреждении термопленки капельки тонера могут попасть на нагревательный элемент и держатель. Их надо тщательно и аккуратно удалить. В противном случае вновь установленная термопленка может прилипнуть с обратной стороны к нагревательному элементу или держателю и легко повредиться.

И в заключение несколько советов:

- используйте бумагу только качественных сортов и подходящей плотности. Вы таким образом сэкономите на техническом обслуживании и ремонте и продлите ресурс фоторецепторного барабана. То же можно сказать и о тонере;

- не курите вблизи принтера, этим можно загрязнить оптическую систему;

- не принимайте пищу вблизи принтера. Этим вы можете привлечь тараканов или других насекомых, которые обоснуются в укромных местах, в частности, в оптической системе, и со временем обязательно загрязнят ее.

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

С момента выпуска фирмой Microsoft своего очередного продукта — операционной системы Windows Millennium Edition — прошло не так уж много времени. Споры о ее достоинствах и недостатках продолжаются. В этой статье я решил вернуться к этой теме и высказать свою критическую точку зрения по этому поводу.

Думаю, что среди читателей найдутся такие пользователи, которые хотели бы попробовать установить на своем компьютере Windows Millennium, чтобы стать "крутым" и идти в ногу со временем, но еще твердо не решили, и находятся в некотором раздумье по поводу того, стоит это делать или нет. Ну что же, дерзайте, если есть желание, вкусите очередной продукт "мелкомягких".

Я тоже решил установить у себя Windows Me, но месяца через два снес ее и вернулся назад к Windows 98SE. Может быть, я консервативен, или сыграла роль сила привычки, не знаю. Лучше расскажу о своих впечатлениях относительно Windows Me. Возможно, кому-то это пригодится и убережет от лишних ошибок и ненужной нервотрепки.

Для начала о системных требованиях. Ставить Windows Me стоит только на мощные машины, второй "пень" и AMD K6-2 — это минимум, хотя сама фирма Microsoft опустила планку нижней границы до 150 МГц. Перед установкой убедитесь, что имеется достаточно свободного места на жестком диске. Учитывайте то, что после инсталляции система с папкой Windows вместе с "Programs Files" плюс "_Restore" займут на винчестере больше 1 Гб. Оперативной памяти для Windows Me потребуется минимум 32 Мб, лучше — 64 Мб и более. Перечислю сразу все те негативные моменты, с которыми я столкнулся, а потом для ясности остановлюсь на них подробнее. Итак, все по порядку.

1. "Пожирающая" свободное место папка "_Restore".

2. Громоздкий Media player.

3. Слишком большой размер файла подкачки.

4. Плохая дозвонка к провайдеру.

5. Отсутствует режим эмуляции MS-DOS.

6. Нет возможности отредактировать конфигурационные файлы.

7. Звуковая карта после выхода из режима приостановки работы молчала.

8. Word из Office 2000 после загрузки документа долго "думает".

Девять причин не ставить Windows Millennium

9. Просмотрщик графики Cads 32 v2.41 стал "глючить".

Теперь более подробно.

1. После установки Windows Me, на винчестере сразу же появилась "не заказанная" мною директория (с атрибутами hidden) — "_Restore", которая стала разбухать с каждым днем как на дрожжах. Удалось выяснить, что она будет постоянно увеличиваться, и может занимать место от 200 Мб (минимум) до 12% от общей емкости жесткого диска (максимум). В этой папке программа восстановления системы System Restore сохраняет системные "снимки" на тот случай, чтобы пользователь мог произвести восстановление системы, если будет нарушена ее работа, т.е. произвести корректный откат к тому рабочему состоянию, когда система была полностью работоспособна. К тому же, System Restore "притормаживает" производительность системы, контролируя все наши действия по удалению файлов. К счастью, бороться с этой бедой можно.

2. Привычные проигрыватели "Универсальный" и "Лазерный" бесследно исчезли, и вместо них появился неповоротливый и громоздкий монстр под названием Media Player, причем довольно "заторможенный". Для проигрывания какого-нибудь WAV-файла приходится долго ожидать загрузки этого исполина. Такое ощущение, что как будто бы грузится нечто глобальное, типа CorelDraw или AutoCAD последних версий. Отменить при инсталляции установку Media Player не представлялось возможным, и он явно не шел ни в какое сравнение с более "шустрым" и популярным WinAmp.

3. С каждой новой версией Windows файл подкачки набирает свой "вес". Каждый пользователь знает, что чем меньше размер этого файла, тем лучше. В Windows 95 после загрузки этот файл занимал 8,2 Мб, в Windows 98 SE он вырос до 32 Мб, в Windows Me раздулся уже до 60 Мб. Складывается такое впечатление, что разработчики Microsoft исходят из принципа — чем больше, тем лучше. А это не всегда бывает верно. Можно только предположить, какого же размера будет файл WIN386.SWP в следующей версии.

4. Хотя в рекламных буклетах и пишут, что система Windows Me максимально ориентирована на подключение и работу в сети Интернет, дозвониться к провайдеру почему-то стало намного труднее. Постоянно возникали какие-нибудь сложности с дозвонкой. То программа не может определить наличие гудка в линии, то никак не может договориться с модемом на сервере провайдера. В общем, "звонилка" в Windows Me оставляет желать лучшего.

5. В Windows Millennium отсутствует возможность перезагрузиться в режим эмуляции MS-DOS. Этот режим все же иногда был нужен, чтобы, например, обновить прошивку во Flash BIOS, запустить утилиту Norton Disk Editor или какую-нибудь другую DOS'овскую программу, которая не хочет нормально работать в обычных условиях.

6. Нужен мне, к примеру, виртуальный электронный диск — этакий удобный "карман" для временного хранения файлов. Я попытался соответствующим образом отредактировать конфигурационные файлы CONFIG.SYS и AUTOEXEC.BAT, но не тут-то было (кстати, я был немного удивлен, когда увидел, что CONFIG.SYS имеет нулевую длину). Так вот, Windows Millennium, конечно же, позволил мне покопаться в этих файлах, но после перезагрузки системы все вернулось на исходные позиции, т.е. все стало как было до редактирования. Система дает возможность только добавить дополнительные пути к используемым каталогам — строка Path в файле AUTOEXEC.BAT. В общем, свобода настройки конфигурации под свои нужды довольно сильно ограничена.

7. Почему-то Millennium не понравилась моя звуковая карта. После выхода компьютера из спящего режима "звуковуха" отказывалась просыпаться. Чтобы она зазвучала, приходилось каждый раз "будить" ее значком, расположенным в System tray, изменяя положение ползунка регулятора громкости. Возможно, дело только в моей звуковой карте (у меня Vortex 8810) или в драйверах к ней, но другую карточку я не пробовал ставить.

8. Было замечено, что Word из пакета Office 2000, установленный из-под Millennium, работает со странностями — после загрузки документа в Word, последний долго о чем-то "думает", прежде чем допустить пользователя к редактированию документа. При установке пакета Office 2000 из-под Windows 98SE подобных вопросов никогда не возникало.

9. Ну и, наконец, последнее. Кроме Word, из-под Millennium не совсем корректно стал работать мой любимый графический просмотрщик ACDSee 32 v2.41. Как известно, в состав этой популярной программы входят браузер и вьюер. Так вот, вся проблема в том, что после каждого запуска браузера последний выдает на экран только список файлов графического формата в текущей директории, но на экране монитора их не показывает. Чтобы все-таки заставить браузер показывать графические файлы, надо перейти в любую другую директорию, а потом вернуться назад. Получается, что браузер не может с ходу выполнить свои функции, что никогда не наблюдалось при его запуске из-под Windows 98SE.

Конечно, я не хочу сказать, что новая операционная система Windows Me полностью "кривая" и "заглюченная", а также сплошь и рядом состоит из одних "багов". Есть, конечно, и положительные моменты. Мне, например, понравилось, как проведена в Windows Me организация главного меню и улучшен пользовательский интерфейс при работе с мышью. Но, взвесив все "за" и "против", я все же предпочел вернуться к проверенной временем и более надежной "старушке" Microsoft Windows 98 Second Edition. Как говорится, старая лошадка борозды не портит. А глубже "пахать" я ее заставлю, периодически обновляя через Интернет, используя при этом утилиту Windows Update, которая позволяет скачивать с сервера фирмы Microsoft самые свежие обновления и "заплатки" к системе.

Рисунок О.Попова

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

Автоматическое определение кодировки текста-2

В [1] я рассказывал о способе автоматического определения кодировки текста для случая, когда текст может быть в одной из двух кодировок — ALT или WIN. Кратко напомню, в чем этот способ заключался.

Рис. 1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
9	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
B	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
D	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
F	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Рис. 2

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
A	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

Рис. 3

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
A	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C	ю	а	б	ц	д	е	ф	г	х	и	й	к	л	м	н	о
D	п	я	р	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ
E	Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О
F	П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ

Кодам E0h...EFh в кодировке ALT соответствуют буквы "р"..."я" (см. таблицу на рис.1), а в кодировке WIN — "а"..."п" (см. таблицу на рис.2). Можно выбрать такие два кода из этого промежутка, чтобы для одной кодировки буква, соответствующая первому коду, в среднем чаще встречалась в текстах, чем буква, соответствующая второму коду (то есть первый код встречался в текстах чаще второго), а для другой кодировки — наоборот. Тогда, просто посмотрев, какой из этих двух кодов чаще встречается в анализируемом тексте, можно сделать вывод о его кодировке.

Несмотря на то что такой способ прост, быстр и достаточно надежен, область его применения ограничена.

Во-первых, анализируемый текст должен состоять в основном из строчных букв, ведь именно на подсчете их количества основано определение кодировки. А если текст записан прописными буквами, способ ничем нам не поможет.

Во-вторых, способ "привязан" к кодировкам ALT и WIN, и адаптировать его для другой пары кодировок можно не всегда. Пусть, например, мы хотим различать тексты в кодировках ALT и KOI-8R (далее просто KOI). В кодировке ALT русским строчным буквам соответствуют коды A0h...AFh и E0h...EFh, а в KOI — C0h...DFh (см. таблицы на рис.1 и рис.3). Как видите, эти два множества не пересекаются, а значит, способ в данном случае непригоден.

А вот пару кодировок WIN и KOI (см. таблицы на рис.2 и рис.3), несмотря на то что там множества кодов строчных букв также не пересекаются, различить можно. Для этого данный способ придется несколько изменить. Заметим, что в кодировке WIN коды C0h...DFh соответствуют прописным буквам, а E0h...FFh — строчным, причем порядок следования прописных и строчных букв один и тот же. В кодировке KOI — точно то же самое, только сначала идут строчные буквы, а затем прописные (разумеется, порядок букв в этих кодировках различен). Так вот, выберем из промежутка C0h...DFh два кода по такому же принципу, как для пары кодировок ALT и WIN. Обозначим их c1 и c2, а их количество в анализируемом тексте — n(c1) и n(c2). При определении кодировки будем подсчитывать общее количество прописных и строчных букв, т.е. n(c1)+n(c1+20h) и n(c2)+n(c2+20h). Сравнив эти две величины, мы сможем сделать вывод о кодировке текста. При этом правильность определения кодировки не будет зависеть от того, прописными или строчными буквами записан текст!

Когда я решил реализовать в своей программе BestView автоматическое определение кодировки при просмотре текстовых файлов (для трех наиболее часто встречающихся кодировок — ALT, WIN и KOI), возникла необходимость в отыскании нового, свободного от вышеописанных недостатков способа определения кодировки текста.

Размышляя над этой проблемой, я вспомнил о прочитанной достаточно давно статье [2]. Там упоминалось о простейшем способе орфографического контроля текста, заключающемся в следующем: не все возможные сочетания букв встречаются в словах русского языка, и если в слове обнаружено такое не встречающееся или, иными словами, недопустимое сочетание букв — значит, слово написано с ошибкой. И я подумал — может быть, удастся приспособить это для определения кодировки?

Вы, наверное, неоднократно замечали — если кодировка текста определена неправильно, он превращается в какую-то абракадабру. Логично предположить, что при этом появляются недопустимые сочетания букв. Тогда получается, что для оп-

Рис. 4

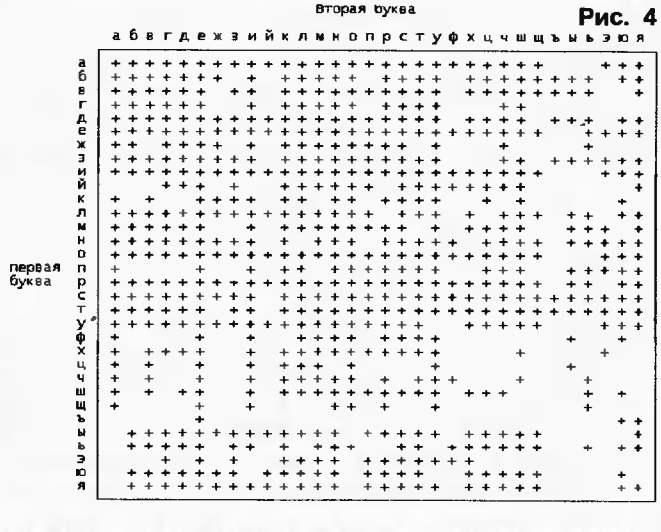
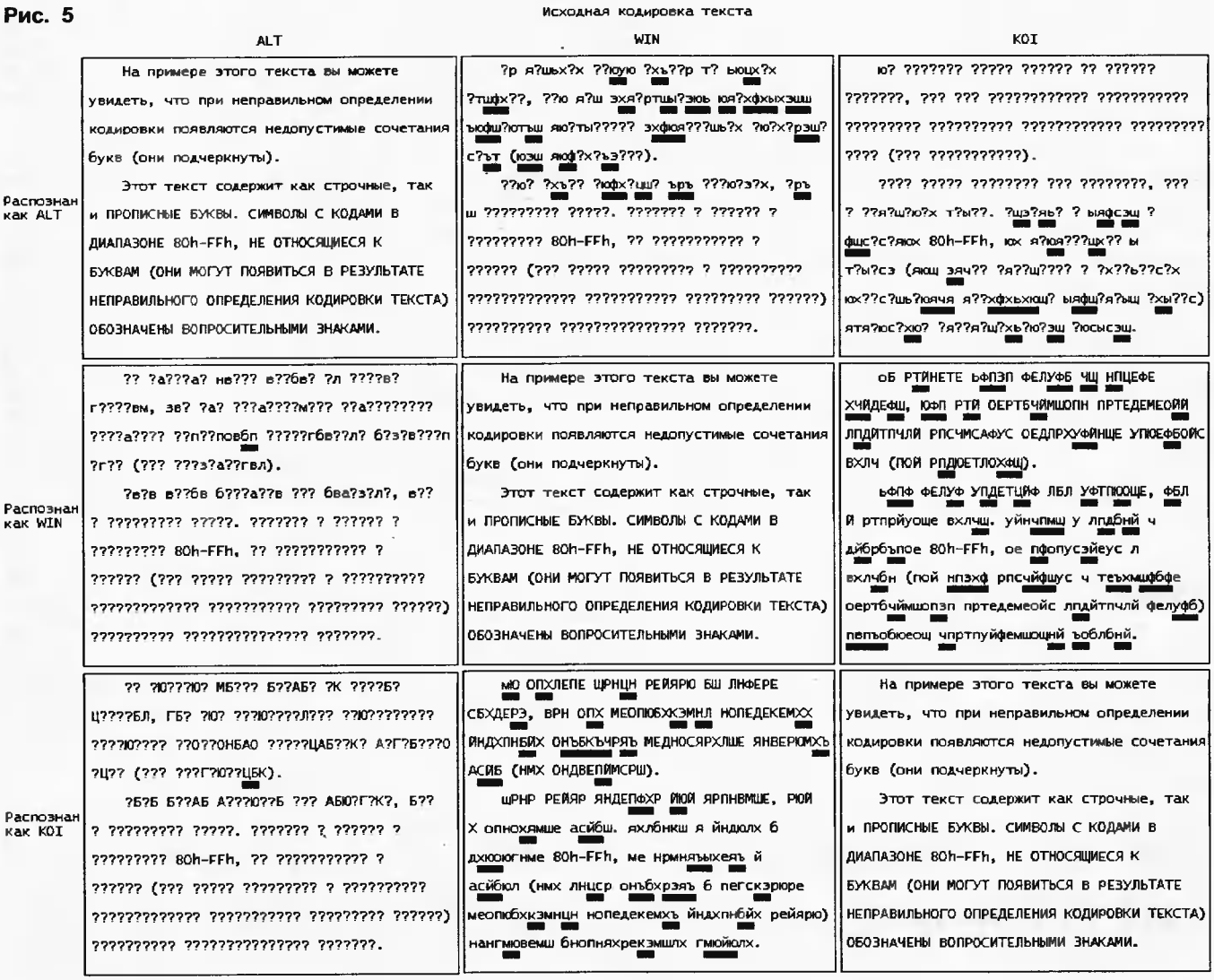


Рис. 5



определения кодировки достаточно выбрать из трех возможных вариантов — ALT, WIN и KOI — тот, при котором в тексте не будет недопустимых сочетаний.

Решив проверить это на практике, я начал с того, что построил таблицу допустимых сочетаний из двух букв (таблица на рис.4), обработав для этого 860-килобайтный текстовый файл.

Оказалось, что из всех возможных сочетаний (32*32=1024) лишь 709 (чуть больше 69%) являются допустимыми.

С помощью этой таблицы уже нетрудно было проверить, действительно ли при неправильном определении кодировки появляются недопустимые сочетания букв (таблица на рис.5).

Как можно видеть, недопустимые сочетания действительно появляются, но не всегда. Так, если анализируемый текст — в кодировке ALT и записан прописными буквами, то в нем не появится недопустимых сочетаний, если считать, что его кодировка — WIN или KOI. Дело в том, что коды прописных букв в кодировке ALT (80h...9Fh) не являются кодами букв в этих кодировках. Аналогичная ситуация возникает, если мы имеем текст в кодировке WIN, записанный прописными буквами — если считать, что его кодировка ALT, недопустимых сочетаний также не появится. То же самое будет и в случае, когда текст в кодировке KOI записан строчными буквами, а мы рассматриваем вариант, что его кодировка — ALT.

В таких случаях метод определения кодировки не даст однозначного ответа — ведь он рассчитан на то, что недопустимых сочетаний не будет лишь в одном варианте из трех. Поэтому добавим еще одно условие — если недопустимых сочетаний не оказалось в нескольких вариантах, выбираем из них тот, в котором больше допустимых сочетаний.

Метод рассчитан на то, что при правильном определении кодировки в тексте не будет недопустимых сочетаний. Но они все-таки могут появиться по разным причинам: ошибки, опечатки в тексте, использование каких-либо специальных терминов или очень редко встречающихся слов. Это надо учесть. При рассмотрении всех трех вариантов может не найтись ни одного, где не было бы недопустимых сочетаний — тогда выбираем тот вариант, где их меньше (а если таких вариантов несколько — выбираем из них тот, где больше общее количество сочетаний).

Может случиться так, что при рассмотрении "правильного" варианта недопустимые сочетания будут, а при рассмотрении "неправильного" их не будет за счет того, что коды букв текста не будут в этом варианте кодами

букв. В результате кодировка будет определена неправильно. Чтобы избежать этого, введем еще одно условие — если при рассмотрении какого-либо варианта в тексте оказалось не более одного недопустимого сочетания, скажем, на 32, считаем, что недопустимых сочетаний вообще нет.

Итак, сформулируем окончательный вариант алгоритма определения кодировки текста.

1. Предположив, что текст — в кодировке ALT, подсчитываем в нем количество всех 2-буквенных сочетаний (all_1) и недопустимых сочетаний (bad_1). Аналогично поступаем для кодировок WIN (получаем all_2 и bad_2) и KOI (получаем all_3 и bad_3).

2. Производим коррекцию — обнуляем bad_1, если bad_1*32≤all_1. Аналогично для bad_2 и bad_3.

3. Сравнив bad_1, bad_2 и bad_3, выбираем из них наименьшее. Если это можно сделать однозначно, то на этом определение кодировки закончено — если меньше всех оказалось bad_1, значит, текст в кодировке ALT; если bad_2 — WIN; если bad_3 — KOI.

4. Если же среди bad_1, bad_2 и bad_3 оказалось несколько одинаковых наименьших значений, сравниваем соответствующие им количества всех сочетаний (например, если наименьшими оказались bad_1 и bad_3 — сравниваем all_1 и all_3). Если это можно сделать однозначно, то на этом определение кодировки закончено — наибольшее из сравниваемых значений и соответствует кодировке (скажем, если наибольшим оказалось all_1 — значит, текст в кодировке ALT).

5. Если же среди all оказалось несколько одинаковых наибольших значений, выбираем из них одно случайным образом. По нему и выдаем ответ.

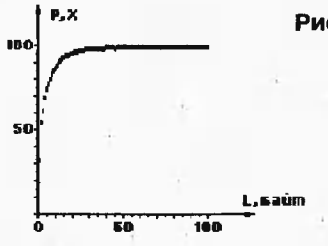


Рис. 6

На рис.6 показана зависимость вероятности правильного определения кодировки (P) от длины текста (L). В анализируемых текстах русские буквы составляли в среднем 70%, тексты в кодировках ALT, WIN и KOI встречались с равной вероятностью, тексты из прописных и строчных букв встречались с равной вероятностью. Как видно, точность определения кодировки с помощью этого алгоритма зависит от длины текста.

Если вы сравните этот график с графиками, приведенными в [1], то увидите, что точность определения кодировки гораздо выше — график гораздо быстрее выходит на уровень 100% правильного распознавания! И это при том, что выбор одной кодировки из трех (к тому же, для текстов как из прописных, так и из строчных букв!) — более сложная задача, чем выбор одной кодировки из двух.

Литература

1. И.Рощин. Автоматическое определение кодировки текста. — Радиомир. Ваш компьютер, 2001, №10, С.34.

2. А.Шкроб. Я не любитель, я другой.... — Компьютерра, 1998, №№24-25.

(Окончание следует)

По страницам электронных изданий

AUTOFIRE

(c) DELTA/PHG

Хочу рассказать поклонникам аркадных стрелялок типа FLY-SHARK, MARADEUR и т.д. об одном простеньком устройстве, которое позволяет сохранить свои пальцы и клавиатуру. Это устройство именуется в народе "AUTOFIRE". Подобные устройства уже несколько раз описывались в "Радиомире", но их основным минусом было отсутствие синхронности с чтением состояния джойстика.

Алгоритм стрельбы:

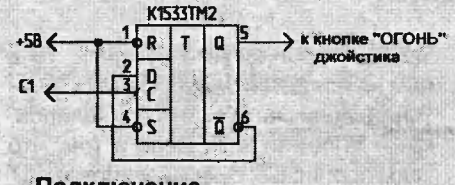
1. Если нажата кнопка "ОГОНЬ", то производится выстрел.

2. Программа проверяет, нажата ли кнопка "ОГОНЬ".

3. Если кнопка не нажата, то переход на п.1.

4. Если кнопка нажата, то переход на п.3.

Все прошлые схемы работали, так сказать, "от балды", и находились непосредственно внутри джойстика. Предлагаемая схема работает именно с сигналами компьютера, что позволяет достигнуть максимальной частоты переключения.



Подключение

Так как я являюсь счастливым обладателем компьютера "БАЙТ", то искать, где расположены сигналы у разных там "ПРОФИ" или других машин, придется вам самим. Сигнал С1 должен изменяться при чтении данных от джойстика. У "БАЙТА" это первый вывод микросхемы 1533АП3 (она одна в компьютере). Сигнал может быть как "1", так и "0" — главное, чтобы он сменялся при чтении из порта джойстика. Сигнал "ОГОНЬ" джойстика идет в "БАЙТе" на вывод 11 микросхемы АП3, в других компьютерах его можно снять с разъема джойстика. В данной схеме я использовал микросхему 1533ТМ2, но, в принципе, подойдет и другой триггер, можно даже счетчик попробовать всунуть, ну, это уже кто как хочет. Устанавливается эта микросхема на любой логике, ведь выводы питания у них уже совпадают. Блокировать AUTOFIRE можно "разрывая" проводник, идущий на кнопку "ОГОНЬ". Я прикольнулся и поставил у себя AUTOFIRE на клавиатурном джойстике, ведь у "БАЙТа" две кнопки "ОГОНЬ". Левая кнопка у меня обычная, а правая — "AUTOFIRE". Можно микросхему и в джойстик засунуть, но тогда придется провести туда сигнал С1.

Схема лично у меня заработала сразу, без всяких там "подборок".

Проверить сей простой девайс можно, набрав в Бейсике:

1. PRINT IN 31: GOTO 1

При нажатой кнопке на экране должно появиться следующее:

```
0
16
0
16
0
... й т.д.
```

Если хотите, чтобы частота выстрелов была постоянной, то можете тактовый вход С1 соединить с сигналом CLK или INT процессора.

Подготовил Е.Зарецкий.

А.ПЕХИМЕНКО,
459120, Казахстан,
г.Рудный, ул.Чкалова, д.47,
E-mail: palex@rambler.ru

Режим мультиколор для Spectrum

Я являюсь спектрумистом со стажем. Хотя дома у меня есть Pentium 2, Spectrum я не бросаю. Хочу предложить для тех, кто умеет держать в руках паяльник, схему расширения графики обычного Spectrum'а — режим "мультиколор 1 байт — 1 цвет". Данная схема была опробована на компьютерах "Ленинград-48, -128", и должна также пойти на "KAY-48, -128", "Скорпион-256, -1024".

Все началось с того, что я прочитал в одном из электронных журналов ("Миракл"), что режим мультиколор можно сделать с помощью трех "перерезов" на плате и одной микросхемы КП11. Я пробовал искать соответствующую схему в электронных журналах для Spectrum, но, кроме подобных слов, ничего не нашел. Попробовал поискать схему в Интернете — тоже ничего не нашел.

Для сведения спектрумистов — огромную коллекцию электронных журналов, программ, игр, музыки, и т.д., а также ссылки на другие спектрумовские сайты можно найти на сайте http://www.zx.da.ru, схему ZX-совместимого компьютера "Ленинград" — на сайте http://www.chat.ru\zrm\.

Тогда взял я схему "Ленинграда", хорошо разобрался в ней, и привожу вам один из вариантов схемы такой доработки с использованием одной микросхемы КП11 и разрывом трех дорожек на печатной плате, а также принцип ее работы. На рис.1 приведен фрагмент схемы "Ленинграда-48" до доработки, на рис.2 — после, с включенным режимом "мультиколор 1 байт — 1 цвет".

Функциональное назначение сигналов:

- H1 — переключение обращения к ОЗУ СРУ и видео (0 — СРУ, 1 — видео);
- H2 — выборка из ОЗУ байта пикселов и байта атрибутов (1 — пикселы, 0 — атрибуты).

Согласно схеме на рис.1, сигналом H1 производится поочередное обращение к ОЗУ СРУ и видео. В цикле обращения видео к ОЗУ при H1=1 производится поочередная выборка из ОЗУ байта пикселов по адресу 16384 (6144 байта) и байта атрибутов по адресу 22528 (768 байтов).

Согласно схеме на рис.2, режим мультиколор включается установкой в состояние логической "1" 4-го бита

Рис. 1

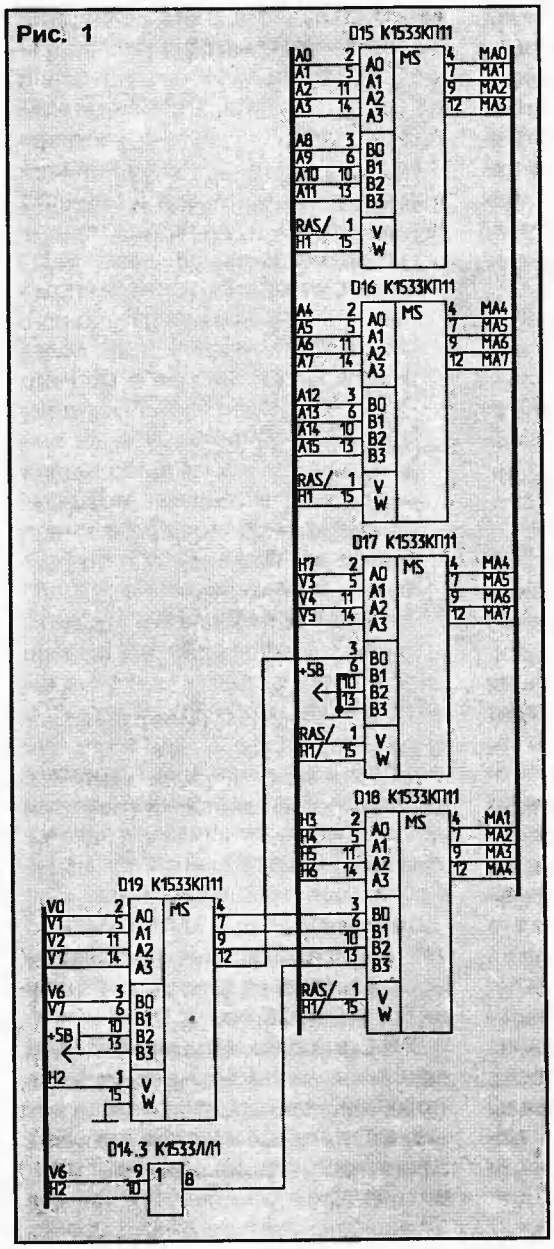
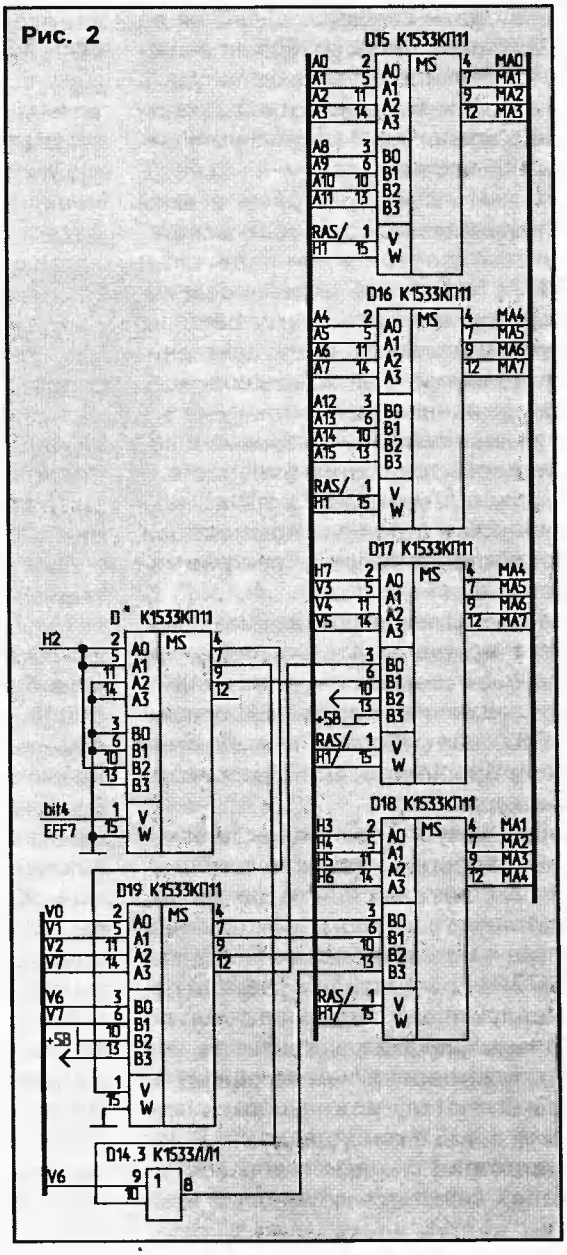


Рис. 2



порта EFF7, и производится поочередная выборка байта пикселов по адресу 16384 (6144 байта) и байта атрибутов по адресу 24576 (6144 байта). Это проверяется сопоставлением адресов СРУ и видео. В принципе, режим мультиколор можно "воткнуть" на любой свободный порт, но я рекомендую использовать для этого стандартный порт EFF7.

На оригинальной плате "Ленинграда" разрываются три проводника — отрезаются дорожки, идущие к выводу 1 микросхемы D19, к выводу 10 микросхемы D14.3 и к выводу 6 микросхемы D17 (позиционные обозначения элементов приводятся в соответствии с оригинальной схемой компьютера). Производятся соединения согласно схеме на рис.2.

Также я рекомендую установить на вашем компьютере режим уровня черного (если он не установлен), иначе на экране будет плохой переход цветов.

Если вам что-то будет непонятно, пишите мне на обычный или электронный адрес.

Обменяюсь программами и схемами по "железу".

И.ЦАПЛИН,
г.Краснодар.

Кодер для ППЗУ

Эта статья может оказаться полезной для читателей, занимающихся конструированием автоматов световых эффектов на ППЗУ.

Автоматы световых эффектов на ППЗУ и счетчиках [1-3] просты в изготовлении и позволяют получать широкий набор эффектов без каких-либо ограничений со стороны аппаратной части.

Ответственным этапом в этом случае является разработка световых эффектов и их перенос в ППЗУ. Побитовое "моделирование" переключений источников света на бумаге отнимает много времени, да и результат не всегда совпадает с ожиданиями, поскольку не исключены ошибки, связанные с переводом данных из бинарного в гексадецимальный формат или пропуском отдельных адресов при составлении таблицы программирования.

Предлагаемая программа позволяет моделировать световые эффекты и сохранять данные для последующего программирования ППЗУ. Она работает на любой модели Spectrum-совместимого компьютера.

При запуске программы необходимо ввести количество комбинаций эффектов в одном цикле. Как правило, при программировании даже таких малоемких ППЗУ как К573РФ2, 3-4 старших разряда резервируют для получения дополнительных эффектов. Учитывая, что в цикле переключения эффекты повторяются, можно ограничить цикл 16...32 комбинациями. В соответствии с количеством комбинаций создается двумерный массив a$. При вводе элементов массива a$ в нижней строке экрана выводится побитная "линейка", упрощающая ввод данных в бинарном строковом формате (например, 10000001, 11111111, 00000000 и т.д.), которые затем преобразуются в числовой десятичный формат (соответственно — 129, 255, 0 и т.д.) и сохраняются в памяти, начиная с адреса 50000. Выбор адреса — произвольный, поскольку закодированные данные занимают небольшой сегмент памяти.

По окончании ввода данных экран очищается. Затем на экран выводятся 8 ярко-красных прямоугольников, цвет которых автоматически меняется на ярко-белый в соответствии с включеными битами. Цикл повторяется до нажатия клавиши "BREAK".

Если необходимо перенести последовательность переключений в ППЗУ, следует набрать команду GOTO 270. В этом случае содержимое блока памяти с адресами 50000...(49999+c) еще раз будет выведено на экран и затем сохранено в файле "cod" на дискете. Чтобы сохранить данные на кассете, из строки 280 необходимо убрать все команды, расположенные перед командой SAVE. Имя файла с данными может быть иным, для этого его следует изменить в последней строке.

Последовательно загружая кодовые блоки в память компьютера, из фрагментов кода можно создать файл для программирования ППЗУ. Необходимо только не допускать пробелов между блоками кода.

Листинг программы:

```
  10 INPUT "Сколько комбинаций в
цикле? ";c: LET p=50000
  20 CLS : PAPER 0: BORDER 0: INK 7:
BRIGHT 1
  30 DIM a$(c,8)
  40 FOR i=1 TO c
  50 PRINT AT 21,1;"87654321"
  60 INPUT a$(i)
  70 LET s=VAL (a$(i,1))*128
  80 LET d=VAL (a$(i,2))*64
  90 LET f=VAL (a$(i,3))*32
  100 LET g=VAL (a$(i,4))*16
  110 LET h=VAL (a$(i,5))*8
  120 LET w=VAL (a$(i,6))*4
  130 LET k=VAL (a$(i,7))*2
  140 LET l=VAL (a$(i,8))
  150 LET z(i)=s+d+f+g+h+k+l+w
  160 POKE p,z(i): LET p=p+1
  170 PRINT:PRINT:
PRINT AT i,0;a$(i),z(i)
  180 NEXT i
  190 CLS
  200 FOR i=1 TO c
  210 FOR n=1 TO 8
  220 LET j=VAL (a$(i,n))
  230 POKE 22848+2*n,80+40*j
  240 NEXT n
  250 NEXT i
  260 GO TO 200
  270 CLS : FOR i=50000 TO 49999+c:
PRINT i;" > ";PEEK i: NEXT i
  280 RANDOMIZE USR 15619: REM :
SAVE "cod" code 50000,c
```

Литература

1. А.Емельянов. Доработка автомата световых эффектов. — Радиолюбитель, 1996, №9, С.26.

2. В.Василенко. Автомат световых эффектов на ППЗУ. — Радиолюбитель, 1997, №7, С.22.

3. В.Сумченко. Автомат световых эффектов. — Радиолюбитель, 1997, №11, С.23.

К.КЛИЩЕНКО,
г.Минск.

Романтика Вампира необыкновенного

Судя по этому зубу, граф Дракула Задунайский был человеком весьма странным и неприятным.
А.Стругацкий, Б.Стругацкий."Понедельник начинается в субботу"

Погребальная контора: милости просим.
И.Ильф, Е.Петров."Двенадцать стульев"

Мне часто жалко свой Word. Ему вполне определенно надоело видеть "взахлебные" отзывы о помогательно-убегательных РПГ и всяческих одиноких мегагероях с холодным оружием и горячей кровью (а в связи с последними, и о моей смертельно наскучившей Ларочке). Но не судьба ему расслабиться и лицезреть мои потуги в расхваливании нового сверхтрехмерного тетриса с долби-звуком и подвижной камерой. Мучайтесь и вы, читатели. Вновь супергерой. Вновь Action, но с оригинальной концепцией (не стоит расстраиваться, когда я говорю так о каждой второй игре). Как вы думаете, откуда он пришел? Откуда, откуда? Звучит грубо, но по смыслу верно, с приставок! Кто главный герой игры, очевидно людям, игравшим в первую часть игры и удосужившимся прочитать название. На мой, извращенный постоянным общением с людьми и не слишком, взгляд, в последнее время вампиров часто романтизируют и превращают в этаких спасителей мира, вынужденных страдать и прятаться от переполнившей мир пищевой субстанции. На этом настаивает и данная игра, и фильм Blade, и интересное детище монструозных игроделов — Vampire: The Masquerade. Видимо, разочарование в человечестве при взгляде в зеркало так велико, что авторам просто необходима отдушина.

За этими размышлениями по поводу и без повода и застала меня судьба в лице игровой Немезиды, которая пришла мстить за скуку томлений от собственной горькой судьбы и бездарности людей, расставляющих даты релизов. Суровая древнегреческая женщина приготовила мне работы на три следующие статьи. Но сначала я твердо решил разобраться с вампирской тематикой в мыслеизвержениях людей игропроизводящих.

Наш герой скрывает свою нижнюю часть лица под тряпкой (ни на что другое это полотнище не похоже). Причина вовсе не в прыщах, покрывающих его подбородок. Под ней Разиэль прячет свою нижнюю челюсть, которая, в отличие от обыкновенных вампиров, у него рабочая. Открывает он ее (этот процесс больше похож на выдвигание тумбочки) не для того чтобы кусать. Он собирает в нее души. Именно они, а не гнусное сборище эритроцитов являются его пищей и основой смертедеятельности (ну все вампиры, вроде как, жмурики). Души можно взять у поверженных врагов и на том свете, причем

тот свет является, как ни странно, именно тем светом.

Именно на взаимопомощи бытия и отрицательного бытия и построен основной процесс. По большому счету, именно спектральный мир и является основой оригинальности этой игры (каюсь, не видел ни первой части, ни еще одной игры, связанной с ними). В обычном, т.е. нашем с вами мире, герой силен и беспощаден, но обыденен. Прыгает в четыре раза выше собственного роста, дерется руками сильнее, чем люди оружием, разговаривает и собирает души. Так как никто души "за так" не отдает, приходится убивать. Для разнообразия авторы позволяют поднимать оружие поверженных врагов. Но и без него весело, учитывая не самое удачное управление. Забавным видится умение врагов отлетать метров на десять при каком-нибудь удачном ударе ногой или оружием потяжелее. Сразу вспоминается трудное детство и видеосалон с однотипными (но, как ни странно, не приедавшимися) боевиками из Сянгана. Каждое орудие производства позволяет нанести последний удар, в узких кругах специалистов и широких адептов именуемый фаталити. Но этот зрелищный способ расправиться играет с вами не самую лучшую из шуток. На половину из арсенала вампиретто хочет насадить врага, как на кол. Но при этом герой оставляет свой "штык" в теле врага. Просто гениально, если учесть, что рядом еще двое жаждут вашей..., ну предположим, смерти. А на изъятие инструмента нужно время. Хотели как лучше... Ну, в общем, как хотели, так и хотели. К тому же, в поэтустороннем мире герой открывает двери и совершает большинство трепательных движений языком.

Но стоит нажать пару кнопок, и вам повезет увидеть, как изменяется мир. От вашего персонажа пойдут волны, искажающие действительность. Мир приобретает непонятную человеческому разуму ясность и окрашивается в сине-зеленые тона. Для вас больше не существует предметов и нельзя воздействовать на объекты. Для вас исчезает вода и сильно изменяется геометрия окружающего пространства. Параллельные прямые пересекаются, а колонны, стены и пол стараются искривиться позамысловатее. В этом загробном существовании, кроме проблем, есть и польза. Далеко не все души здесь имеют владельца, и халявное лечение процветает. Кроме того, в призрачной форме наш

ясно-пустоглазый товарищ может проходить сквозь решетки как терминатор (не Шварценеггер), то есть попадать в ранее недоступные места. Остальные прелести обнаружите сами. Однако подлость или невозможность полностью сбалансировать игру не позволяют нам переходить из одного мира в другой по своему желанию. Туда попасть можно отовсюду, а вот Оттуда — только в определенных точках.

Немалую роль в ваших странствиях играет клинок Soul Reaver (игра, между прочим, называется Soul Reaver 2). Это не совсем обычное оружие (даже и не оружие). Питается, как и вы, душами. Так что приходится выбирать между возможностью полечиться и легко убить врага. Но если честно, мне не очень хочется рассказывать об этом немаловажном, а во многом и определяющем элементе игры. Слишком он связан с сюжетом, а эта часть целиком на вашей совести.

Сражаетесь вы по разные стороны баррикад ради сомнительного удовольствия порешать несложные, но разнесенные в пространстве (к счастью, в пределах одного участка карты) пазлы. А вот бои, решение пазлов и беготня из одного конца мирка в другой дают вам порадоваться умению авторов делать длинные диалоги на движке игры. Вся интересность этих длинных (до двадцати минут) диалогов убивается полным отсутствием субтитров плюс желанием авторов вложить как можно больше интонаций (умоляющую, испуганную, требующую, благосклонную, заискивающую, поясняющую, утомляющую) в невнятный шепот и гневные возгласы. Авторы оставили игроку лишь возможность насладиться изложением на бумаге, после того как диалог состоится. История у авторов получилась если не очень захватывающей (по причине трудности восприятия), то уж точно интересной. Приятно, когда сюжет оставляет в непонятках даже после своего завершения. Хотя не могу сказать, что он завершен. Лично я вижу множественное по вариантам продолжение. Хотя не стоит обольщаться (может, я чего недопонял).

Более того, трактовать концовку можно по-разному (хотя уверен, авторы хотели донести что-то вполне определенное). Честно говоря, уже два дня под впечатлением, а потому чувствую себя глупо, как иголка в стоге сена.

P.S. Не думаю, что игра слишком хороша, и сюжет мог быть посильнее, но судьба Носготов стоит того, чтобы вникнуть.

P.P.S. Так как игра приставочная, то с сохранениями, как всегда, неудачно. Лично мне не нравится система сохранения только в определенных локациях.

Петька 3. Возвращение Аляски

А.ВЕНДИЛОВСКИЙ.

"O! Its Big commander Vasiliy Ivanovish and children Petka!" –
"Никогда, блин, в натуре, я не видел более прикольных чуваков!"
(Закадровый голос переводчика).
Цитата из игры

Жанр квест.
Издатель "Бука".
Разработчик компания "Сатурн".

Итак, наступил конец середины XX века (а если точнее — 1970 г.), когда два великих музыканта были выброшены на музыкальную свалку загнивающего капиталистического мира. Василий Иванович и Петька коротали время в заброшенном музыкальном клубе (а точнее — в вонючем здании на не менее загаженной улице), вспоминая те славные деньки, когда они со всей своей русской душой наяривали русский революционный анархо-рок. Были слава, деньги и девочки... От этих приятных воспоминаний их ничто не могло отвлечь, разве что, небольшой спор с местными панками, в котором каждая из сторон обычно приводила столь веские аргументы как кастеты, цепи, холодное оружие всевозможной длины и пустые бутылки из-под виски. Когда столь увлекательная интеллектуальная беседа практически подошла к своему логическому завершению, когда аргументы красноармейцев, как всегда, оказались более весомыми и конкретными, и самое главное, правильно и в нужный момент приведенными, произошло нечто...

Вспышки молний, внезапная тьма и запоздалое воспоминание о том, что в инструкции говорилось — машину времени нельзя бить ногой, даже в кирзовом сапоге. Иначе... Вот именно это и произойдет, машина начнет выписывать во времени такие кренделя, что даже Эйнштейн со всей своей теорией относительности не сможет определить, что будет через секунду — завтра или позавчера. А вопрос "Где это будет?" покажется сложнее, чем тайна Бермудского треугольника.

Оба красноармейца очутились в 2001 г., возле небезызвестного питейного заведения в мексиканской глуши, которое работает "от заката до рассвета". Несколько затянувшееся знакомство с местными жителями, опять же с использованием "веских аргументов", привело сразу к двум неожиданным последствиям. Первое — возле бара рухнула летающая тарелка. А второе — на месте событий появился Мокс Фалдер, специальный агент ФБР, зани-

мающийся расследованием Ужасных и Секретных Материалов, а также поиском внеземных цивилизаций. И конечно, такая пара "нормальных" объектов не могла не привлечь его внимание. Он вывез Василия Ивановича и Петьку из области повышенного возмущения окружающей среды (вслед им еще долго летели гнусные, но безобидные ругательства и милые, но весьма небезобидные снаряды из гранатомета...) и коротко объяснил им, что мир опять стоит на краю катастрофы, вызванной безответственными действиями с машиной времени. В результате, деревня Гадюкино вместе с ее обитателями дружненько переселилась на... Аляску! И резкое увеличение числа русских в Америке, а самое главное, чисто русского разгильдяйства, к которому американцы ну никак не были готовы, может даже сбросить Землю с ее орбиты. Так что положение срочно нужно исправлять. Но все усложняется тем, что все знакомые героев стали абсолютно другими, и абсолютно не желают узнавать Василия Ивановича и Петьку! Анка — теперь замаскированный агент ФБР, работающая под прикрытием вместе с Моксом Фалдером над Секретными Материалами, и фамилия у нее Скаллир... Небезызвестный Фурманов превратился в проповедника-мормона отца Фурмана, проповедующего Божий Завет и разводящего где-то в американской глуши саблезубых кроликов. Пасечник открыл в себе талант живописца и ведет богемную жизнь на ближайшей к городу свалке, ожидая, когда к нему придет слава. В общем, все вверх дном, да и Аляска почти на другом конце света, и до нее еще добраться надо...

Ну что сказать, нас посетила очередная серия квестовых приключений легендарных красноармейцев, сделанная все той же "Букой". Признаюсь честно, первые две части я с чистой совестью "продинамил", так как абсолютно не находил ничего смешного в пересказывании плоских анекдотов, у которых уже борода до пола выросла.Чего-то подобного я ожидал и от третей части... но я ошибся!

Честное слово, хватит пальцев на одной руке, чтобы перечислить те случаи, когда третьи части удавались разработчикам лучше, чем предыдущие. Можно подумать, что создателям надоело выдавливать из себя нечто тематически-смешное, они взяли все свои прежние наработки и послали их... куда-нибудь по почте, а сами сели и стали прикалываться — над голливудскими штампами, нашими представлениями об Америке, над русскими, над американцами. И самое главное — у них получилось смешно! И смешнее всего не наблюдать за главными героями, а рассматривать окружающий их мир. Мы хохотали всей компанией, когда читали афиши на стенах или разглядывали местные дорожные знаки. А "закадровые" комментарии "аборигенов" уже сами по себе могут заставить вас улыбнуться.

Можно сказать, что чувство юмора у разработчиков выросло настолько, что теперь происходящие на экране события не вызывают у вас ощущения легкого идиотизма, когда приходится изгаляться над собственным разумом, пытаясь понять, что хотят от игрока.

Теперь о технической стороне дела. Квест по-прежнему остался мультяшным — за основу был, видимо, взят движок из прежних частей этой серии и очень серьезно переработан. Это относится и к качеству картинки, цветовому насыщению, а также к тому, что часто "задники" и фон анимированы, причем чаще всего используются вставки "живого видео". Поэтому даже возникли серьезные споры — новый ли это движок, который внешне напоминает старый, или абсолютно переделанный новый.

К достоинствам этой игры нужно причислить и более удобный интерфейс, особенно, что касается "корзинки", в которой персонажи носят всевозможнейшее барахло, найденное ими по дороге. Оглядываясь назад, хочется спросить у программистов — неужели нельзя было сделать это раньше, не дожидаясь третьей части?

Больной вопрос для любого квестомана — пиксельхантинг. Увы, есть одна плохая новость — используемые предметы не подсвечиваются, и их придется отлавливать с помощью курсора. Хорошая новость — все активные предметы имеют большие габариты, так что проблем с "попаданием" быть не должно. По крайней мере больших :).

И последний вопрос, который интересует любого любителя любых игр — системные требования. Будем откровенны — использование новых форматов оцифровки звука и видео (таких как MPEG4) требует и больших мощностей компьютера. Так что владельцам первых "пней" нужно иметь это в виду, а остальные могут не беспокоиться.

Минимальные характеристики

- Windows(r) 95 (не тестировалось), Windows 98;

- Pentium 200+ MMX;

- 32 Мб оперативной памяти (зависит от операционной системы);

- 500 Мб (без компрессии) свободного места на жестком диске;

- графический видеоадаптер 4 Мб, полностью совместимый с DirectX (2 Мб при 16-битном (HiColor) видеорежиме);

- привод компакт-дисков 8x (скорость чтения зависит от конкретной модели);

- любая полностью совместимая с DirectX звуковая карта (конфигурация без звуковой карты проверялась, но детально не тестировалась);

- мышь, клавиатура;

- DirectX 7.

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

Рисунок О.Попова

КУПЛЮ, ПРОДАМ, ОБМЕНЯЮ

*Для публикации бесплатных объявлений **некоммерческого характера** о покупке и продаже радиодеталей, бытовой и радиолюбительской аппаратуры, их текст можно присылать в письме по адресам:*

***117454, г.Москва, а/я 37** или*

***220095, г.Минск-95, а/я 199,** передавать по тел. в Минске **(017) 221-01-10**, либо через E-mail:*

rl@radiopage.by

rm@radio-mir.com

*WWW: **http://radio-mir.com***

Продаю 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

Куплю двухкоординатный графопостроитель ПДП (ЛКД 4-003).
E-mail: angel_of_darkness@pisem.net

Продам акустику "Radiotehnika" (S-30), двухполосную.
Константин.
E-mail: kos_vr@mail.ru

Меняю на комплектующие для IBM **или продам** генераторы ПАЛ/СЕКАМ (габариты 160x60x35 мм), питание КРОНА (9В), литературу и CD-ROM по TV.
620138, г.Екатеринбург, а/я 220. Александр.

Продам коллекцию радиолюбительских CD-ROM (схемы, программы, электронные книги).
620039, Екатеринбург, а/я 172. Сергей.
E-mail: banan@pisem.net

Продаю клавиатуру к ZX-SPECTRUM, плату ZX-SPECTRUM без ПЗУ и кварца, собранную, но не налаженную, платы (5 шт.) программатора MPPS + схему, но нет программы к ней.
Сергей.
E-mail: sergey2b@yandex.ru

15-летний школьник из небогатой семьи с благодарностью **примет в дар** IBM PC-486DX, старые комплектующие, и т.п.
Антон Конопацкий,
ул. Красногвардейская 5, кв.9,
Чернигов-33, 14033,
Украина.
E-mail: netstranger@ukr.net

Ищу информацию об опыте практического применения в радиолюбительских условиях микросхемы КМ1813ВЕ1 (А, Б) — цифровой процессор обработки аналоговых сигналов. Приобрету несколько экземпляров микросхемы КМ1813ВЕ1 и/или готовый модуль с описанием и текстом программы.
Бутов А.Л.,
с.Курба, ул.Юбилейная, 11 — 15,
Ярославская обл., Ярославский р-н,
150533, Россия.

Разыскиваю схему струйного принтера "Электроника МС-6317" и головку к этому принтеру, либо ее возможную замену.
E-mail: saniok@ic.km.ua

Ищу пакеты программ для "ZX-Spectrum": iS-DOS Assembler, iS-DOS ZX-Modem.
E-mail: tzaplin@mail.ru

Куплю ксерокопию принципиальной схемы и описание источника бесперебойного питания BACK-UPS mod.250I ф.APC (American Power Conversion).
Сергей.
E-mail: sergeis@vmail.ru

Уважаемые читатели!

Подписаться на наши журналы (индексы: **48996** — "Радиомир", **48924** — "Радиомир. КВ и УКВ", **48925** — "Радиомир. Ваш компьютер") можно по каталогу Агентства "Роспечать" или по каталогу РО "Белпочта" "Газеты и журналы Республики Беларусь" (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, 4, 5, 7, 8	1 — 5, 7 — 12	1, 4 — 7, 11, 12	25	800	4
1999	2, 3, 9, 10, 11	1 — 6, 10, 11, 12	1 — 3, 5, 6	30	1000	5
2000	1 — 12	1 — 12	4, 6 — 12	35	1200	6
2001	2 — 6	1 — 6	2 — 6	40	1400	6,5
	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	6,5
2002	1 — 12	1 — 12	1 — 12	45	1500	6,5

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

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

При оплате платежным поручением нужно предварительно выписать счет.

Вся информация по тел. в г.Москве (095) 139-75-77; в г.Минске (017) 221-01-10 или по E-mail: rm-sales@radio-mir.com

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.

В МГРСТК: г.Москва, ул.Строителей, 9/10, тел. 131-02-65 (ст. метро "Университет").

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

В интернет-магазине изд-ва **"DMK-Press"**: www.dmkpress.ru с бесплатной доставкой по Москве.

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:
4-й Войковский проезд, 10, тел. 150-09-72 (ст. метро "Войковская").

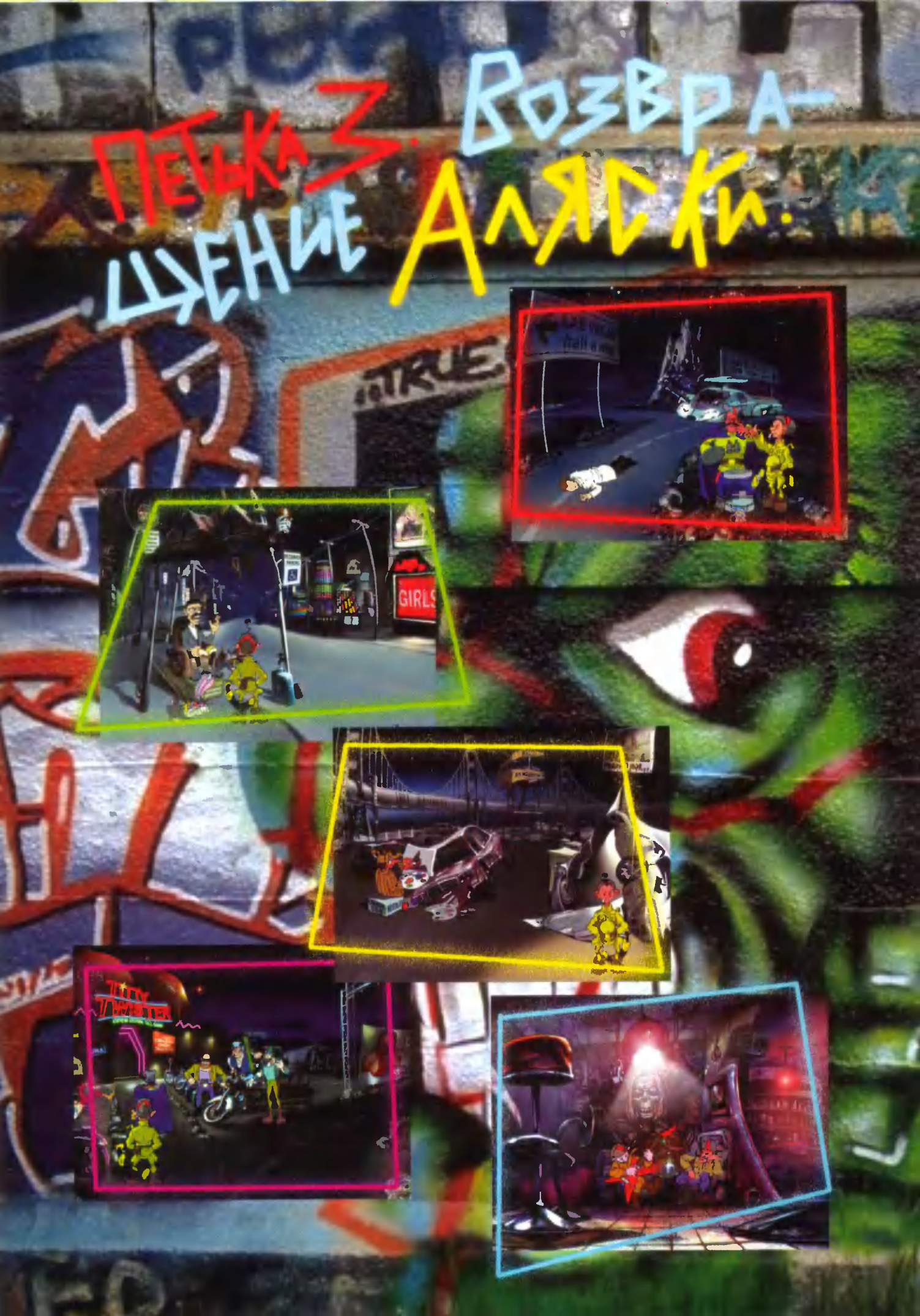
На УКРАИНЕ:

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").

ПЕТЬКА 3: ВОЗВРАЩЕНИЕ АЛЯСКИ.




ЗОЛОТАЯ КОЛЛЕКЦИЯ
ГОНКИ НА ГРУЗОВИКАХ
3В1
ДАЛЬНОБОЙЩИКИ
ДАЛЬНОБОЙЩИКИ II
MERCEDES BENZ
TRUCK RACING
ГОНКИ НА ГРУЗОВИКАХ
Dream Reality Studio
MEDAL OF HONOR
ALLIED ASSAULT
PC CD ROM
EA GAMES
2В1
ПОЛНЫЙ РУССКИЙ ПЕРЕВОД
+ АНГЛИЙСКАЯ ВЕРСИЯ
ALCATRAZ
PRISON ESCAPE
ACTIVISION
Platinum
РУССКАЯ ВЕРСИЯ
EMPIRE EARTH
ВЫПУСК 10
ТАЙНЫ ИГР
КЛАССИКА
1980-2000 ГОДА
БУКЛЕТ СО СПИСКОМ ВНУТРИ
БЕСПЛАТНОЕ
ЭКСФОРС
PC CDROM
SIERRA
ЗОЛОТАЯ КОЛЛЕКЦИЯ
2CD
DIABLO
4В1
РУССКИЕ ВЕРСИИ
Dream Reality Studio
ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
Американский
ГОНЩИК
PC CD ROM
ФАРГУС
НА РУССКОМ ЯЗЫКЕ
COUNTER-STRIKE 1.3
РУССКАЯ ВЕРСИЯ
ПОЛНОЕ
ПРОТИВОСТОЯНИЕ
4 в 1
Противостояние
Опаленный снег
Противостояние III
Внезапный удар
III
ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
Русский Проект
SPIDER-MAN
ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
Русский Проект
RETURN TO CASTLE
Wolfenstein
III