10•2001
http://radio-mir.com
Индексы: 48925, 45995

радиомир

В НОВЫЙ ВЕК — С НОВЫМ НАЗВАНИЕМ!

МУЛЬТИЗАГРУЗОЧНЫЙ

Ваш компьютер

Adobe Photoshop ELEMENTS

РУССКАЯ и АНГЛИЙСКАЯ ВЕРСИИ

ЛЕТО-ОСЕНЬ
REANIMATOR 2001

ИМПОРТНЫЕ И ОТЕЧЕСТВЕННЫЕ ЭЛЕМЕНТЫ

ЭЛЕКТРОННЫЙ СПРАВОЧНИК РАДИОЛЮБИТЕЛЯ

ПАРАМЕТРЫ И ХАРАКТЕРИСТИКИ РАДИОЭЛЕКТРОННЫХ КОМПОНЕН

- Диоды
- Светодиоды
- Варикапы
- Транзисторы
- Микросхемы

+ Схемы на бытовую и промышленную аппар

КОМПАС-3D LT v5.10

AUTOCAD ? НЕТ !
ПРОЩЕ, УДОБНЕЕ, МОЩНЕЕ !

+ КНИГА А. ПОТЕМКИНА "Инженерная графика. Просто и доступно"
+ БОЛЕЕ 1000 ЧЕРТЕЖЕЙ

Профессиональный пакет для работы с цифровым фото и видео

МУЛЬТИЗАГРУЗОЧНЫЙ
ОПЕРАЦИОННЫЕ СИСТЕМЫ

РЕШЕНИЕ ВСЕХ ПРОБЛЕМ!

ALEX SOFT

КОМПЬЮТЕР на ладони

GOLD

WinCE

ПРОГРАММЫ ДЛЯ КАРМАННЫХ КОМПЬЮТЕРОВ

- ПРОГРАММЫ ДЛЯ WINDOWS CE
- ПРОГРАММЫ ДЛЯ CASIOPEA, SHARP и
- ИНТЕРНЕТ - ПРИЛОЖЕНИЯ
- ИГРУШКИ ДЛЯ WINDOWS CE
- ИГРУШКИ ДЛЯ PALM-PILOT

Alex Soft GOLD

НОВЕЙШАЯ ЭНЦИКЛОПЕДИЯ ПСИХОЛОГА 2000

Этот диск будет вам про

PROFESSIONAL CorelDRAW 10
полная русская версия

ЛУЧШАЯ ЛОКАЛИЗАЦИЯ

- CorelDRAW 10 Русская версия
- CorelDRAW 10 Английская версия
- Моментальная смена версий
- Обзор возможностей CorelDRAW 10
- Суперклипарты, фотографии, Обучающие

ALEX SOFT

ВЫПУСК 3

АППАРАТНЫЕ СРЕДСТВА КОМПЬЮТЕР
ЭЛЕКТРОННЫЙ СПРАВОЧНИК + ДРАЙВЕРЫ

ОСНОВНЫЕ КОМПОНЕНТЫ КОМПЬЮТЕРА

- CD-ROM
- DVD-ROM
- HDD
- МОНИТОРЫ
- ПРИНТЕРЫ
- МОДЕМЫ

+ ВИДЕОКУРС ПО СБОРКЕ КОМПЬЮТЕРА

PREMIUM

МАСТЕР-САМОУЧИТЕЛЬ
ПО ЗАПИСИ ЛАЗЕРНЫХ ДИСКОВ

Запись, клонирование, копирование CD

CD-R

- VOB Instant CD DVD v6.0
- Roxio WinOnCD v3.8
- Roxio Easy CD Creator v5.0.1 Platinum
- Nero Burning ROM v5.5.2.3
- Учебник по записи лазерных дисков

CD


MP3

Учредитель и издатель:
ООО "НТК ИНФОТЕХ"

Свидетельство о регистрвции
ПИ №77-7399 от 19.02.2001

Главный редактор
Ольга СТРЫЖАНКОВА

Зам. гл. редактора
Иван БЕЛЬСКИЙ

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

Контактные телефоны:
(095) 105-99-89,
+(37517) 249-41-47

Компьютерная верстка —
Янинв БЕЛЬСКАЯ

Техническая графика —
**Наталья БЕЛЬСКАЯ,
Александр ГОРАЮТИН,
Мария КАЛАБУХОВА,
Игорь ШЕВКУН**

Оформление обложки —
**Наталья БЕЛЬСКАЯ,
Надежда БОГОМОЛОВА**

Коммерческий отдел
(095) 139-75-77,
+(37517) 221-01-10

Адреса для писем:
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

Наши платежные реквизиты:
получатель : ООО "НТК ИНФОТЕХ",
ИНН 7703155561,
р/с 40702810100022120172
в АКБ "Межтопэнергобанк"
корр. счет 30101810900000000237
БИК 044585237
Адрес банка: 107078, г.Москва,
ул.Садовая-Черногрязская, 6

Матариалы для публикации принимаются в рукописном, печатном и электронном вариантах

Требования к графическим материалам рекламного характера в электронном виде: CorelDRAW 6.0, 7.0 все шрифты в кривых, bitmaps 300 dpi; TIFF, 300 dpi; CMYK в сопровождении печатной копии

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

Адрес редакции: 141406, г.Москва, Химки-6,
ул.Библиотечная, 18-84, тел. (095) 105-99-89

Подписано к печати 4.09.2001 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободнвя

Отпечатано в типографии ЗАО "Красногорская типография"

Закез 2823. Тираж 1700 экз.

ВНИМАНИЕ!

НАЧИНАЯ С №7/2001 ЖУРНАЛ ВЫХОДИТ ПОД НОВЫМ НАЗВАНИЕМ
(подписка на "Радиолюбитель. Ваш компьютер" действительна. Подписные индексы сохранились)

радиомир Ваш компьютер

Ежемесячный массовый журнал.
№10/2001. Издается с сентября 1995 г.

ЧИТАЙТЕ В НОМЕРЕ:

В.ЕРМОЛЕНКО.

Запись звуковой информации на CD

Проигрыватели звуковых компакт-дисков распространены гораздо шире, чем MP3-плееры, которые все еще неудобны в пользовании. А вот технология записи компакт-дисков за последнее время перестала быть "профессиональной" экзотикой. Поэтому многие любители музыки записывают аудио-CD с подборками собственных музыкальных программ для прослушивания.

На первый взгляд, запись аудио-CD (CD-DA — CD Digital Audio) выполняется аналогично и даже с помощью тех же самых компьютерных программ, что и запись CD-ROM с компьютерными данными. Однако здесь имеются и свои тонкости, некоторые из которых мы сейчас рассмотрим.

Источники звука

Наиболее распространенным форматом хранения музыкальных файлов является MP3. Звуковые файлы, в принципе, мало отличаются от других типов компьютерных данных, разве что своим объемом. На одном записываемом компакт-диске (CD-R) помещается копия одного аудио-CD или примерно до 10 таких CD, закодированных в формате MP3 со скоростью потока данных 128 Кбит/с. Для хранения коллекции музыки MP3-файлы можно записывать на CD-ROM, как и обычные данные, группируя их в подкаталоги по исполнителям, жанрам, темам и т.д. Сейчас начали появляться проигрыватели CD, позволяющие наряду с CD-DA проигрывать CD-ROM с MP3-файлами. Чтобы вас не подстерегла неприятность в таком проигрывателе, рекомендуем все же придерживаться ряда ограничений при записи собственной MP3-коллекции [1]. Так, не следует использовать русские буквы и знаки препинания в именах файлов и подкаталогов: безопаснее ограничиться набором из латинских букв, цифр, знаков "минус" и подчеркивания. Более того, этих ограничений на используемые символы желательно придерживаться при формировании ID3-тегов MP3-файла. Кстати, расширенные версии тегов (ID3v2) не воспринимаются многими плеерами, а потому из CD-версии звукового файла их лучше удалить.

Большинство записывающих программ заносят звуковые данные на диск из файлов формата WAV (44,1 кГц, 16 бит, стерео). Создать такие файлы можно с помощью звуковых редакторов, подавая на линейный вход звуковой карты сигнал с магнитофона, проигрывателя или другого источника. Выбор подходящих редакторов достаточно широкий. Многие программы могут сами выполнять преобразования форматов "на лету" (из MP3 в WAV и др.), а также вносить "коррекцию", направленную, с точки зрения авторов программ, на улучшение качества звучания или на защиту авторских прав путем искажения копии по сравнению с оригиналом. Однако пользоваться такими режимами следует с большой осторожностью. Гораздо надежнее перед записью собственного CD-DA сначала преобразовать исходный материал в WAV-формат, проверить звучание, а при необходимости — и отредактировать.

Как ни странно, наибольшие сложности представляет качественное копирование собственно CD-DA в цифровые файлы, и это связано не только (и не столько) с защитой авторских прав производителей. Дело в том, что чтение CD-DA и CD-ROM выполняется принципиально различным образом. Чтение CD-DA имеет целью непрерывную подачу звукового сигнала, пусть даже и ценой анесения искажений при ошибочном считывании диска. Для этого как в контроллер привода, так и в программные драйверы встроены алгоритмы интерполяционной коррекции формы сигнала при выпадении одного или нескольких отсчетов. Программы более высокого уровня и пользователь обычно не получают уведомлений о том, что звучание скорректировано.

Чтение CD-ROM, наоборот, рассчитано на точную подачу данных, возможно, ценой повторения попыток позиционирования и чтения дефектного участка диска. Если после нескольких попыток правильно прочитать блок данных не удается, программа бракует его целиком и выдает сообщение об ошибке. Таким образом, часто данные с некачественных CD-DA не читаются, хотя в CD-плеере эти диски проигрываются без перебоев.

Соответственно, содержимое аудио-CD, в принципе, можно прочитать в цифровой файл двумя способами: либо подав аналоговый звуковой сигнал с привода на вход звуковой карты, где он оцифровывается, либо собрав цифровые данные через интерфейс привода с помощью специальных программ. Некоторые читающие приводы оснащены также последовательным цифровым звуковым интерфейсом, который подключается к соответствующему цифровому входу "продвинутых" звуковых карт, тем самым соединяя преимущества этих двух способов.

Если сигнал преобразуется в аналоговую форму, а затем вновь в цифровую, качество звука сильно зависит как от качества звуковой карты, так и от аккуратности в сборке компьютера. Мешающие наводки могут возникать из-за расположения звуковой карты рядом с сильно шумящей видеокартой или сетевой картой, из-за неудачного расположения звукового кабеля или из-за плохой экранировки. Причиной фона переменного тока, кроме перегрузки блока питания после нескольких "апгрейдов", может служить так называемая "земляная петля". Чтобы избавиться от этого эффекта, автор [2] рекомендует отключить экран звукового кабеля с одной стороны — либо у CD-привода, либо у звуковой карты. Для этого достаточно аккуратно вынуть земляной контакт из пластмассовой фишки разъема.

Цифровое чтение (DAE — Digital Audio Extraction) правильно выполняется далеко не всеми приводами. Основная погрешность — нарушение синхронизации адресных меток между кадрами [3]. Различные приводы по-разному позиционируются на начало дорожки, и в результате либо цифровой файл начинается чуть раньше, либо дорожка оказывается обрезанной. Речь идет о неразличимых на слух сотых долях секунды, но ведь при копировании сделанной копии такие погрешности могут накапливаться. Для определения дисков при запросах к централизованной базе данных (CDDB), которые выполняют определенные лицензионные программы, используется индексная карта диска, а создание корректной копии с идентичной индексной картой требует совпадения не только длин звуковых дорожек, но и адресной информации. Для получения точной цифровой копии диска требуются значительные затраты времени на настройку системы копирования. Тренироваться в подборе оптимальных режимов лучше всего на матрицах CD-RW. Автор [3] применяет программу Exact Audio Copy (EAC),

ООО "НТК ИНФОТЕХ"

которая, кроме цифрового чтения, позволяет и записывать диски (хотя этот режим в версии 0.9 недоработан).

Носители записи

Записываемый диск содержит слой органического красителя, в котором лазером выжигаются следы (питы), образующие спиральные дорожки с информацией. Долговечность такой записи оценивалась изготовителями в 50 лет для дисков с цианиновыми красителями (голубых, зеленых) или в 100 лет для дисков с фталоцианиновыми (золотистых). Среди пользователей бытуют и более скептические оценки долговечности — несколько лет. Впрочем, поскольку технология CD-R еще "молодая", надежных данных по сохранности информации просто нет. Дефекты на дисках накапливаются, и в какой-то момент избыточного кодирования информации на CD-ROM уже окажется недостаточно для ее полного восстановления. Деградация же CD-DA носит более плавный характер — изменение звучания за счет коррекции, начиная с некоторого момента, будет становиться все более заметным.

Во всяком случае, к дискам, рассчитанным на длительное хранение, следует относиться бережно — не подвергать их нагреву, действию прямого солнечного света, изгибу, ударам, загрязнениям и т.п. Брать диски руками следует только за края, стараясь не притрагиваться к поверхностям. Особенно осторожно следует применять для CD-R и CD-RW имеющиеся в продаже химические средства для чистки, рассчитанные на обработку штампованных (алюминиевых) компакт-дисков.

Наряду со стандартными матрицами CD-R емкостью 74 мин (650 Мб), некоторое время назад появились новые — на 80 мин (700 Мб). 80-минутные диски поддерживаются практически всеми CD-плеерами и новыми CD-рекордерами. На таких матрицах рабочая поверхность для спиральной дорожки с записью начинается ближе к центру диска, чем в 74-минутных. Этим они отличаются от записей объемом более чем 650 Мб на 74-минутных дисках, когда путем различных ухищрений дорожка 74-минутного диска продлевается ближе к его краю. Поскольку цены 74- и 80-минутных матриц различаются не сильно, пользоваться последним способом не стоит — это работает не на всех приводах. Надо сказать, что довольно многие алюминиевые CD содержат более 74 минут записи, и для их копирования лучше применять 80-минутные матрицы.

Некоторые матрицы CD-R маркированы как "CD-R Digital Audio" или "CD-R for music". Несмотря на утверждения изготовителей, будто бы такие матрицы "оптимизированы" для записи музыки, можно считать, что надежность записей на них ниже. Как уже говорилось, запись музыки отличается от записи данных нечувствительностью к "выпадениям" отдельных битов или отсчетов. Использовать их для хранения данных нежелательно.

Существуют матрицы, специально маркированные как пригодные для записи на повышенной скорости (8x и выше). Слой красителя в таких матрицах тоньше, что позволяет лазеру прожигать его за более короткое время. Однако запись CD-DA выполнять на повышенной скорости (более чем 2x) не рекомендуется. "Смазанная" запись вызовет необходимость более интенсивной коррекции формы сигнала, что приведет к искажениям звука, по-разному заметным при проигрывании этого диска в разных плеерах.

Среди фирм-производителей матриц наиболее уважаемыми считаются Verbatim, BASF/Emtec, Mirex, Memorex, Philips, TDK. Не следует применять так называемые "технологические" матрицы, поставляемые без упаковки и фирменной маркировки.

Рекордеры

Поскольку цены на рекордеры, работающие как с однократно записываемыми (CD-R), так и с перезаписываемыми (CD-RW) дисками, сильно снизились, имеет смысл покупать именно рекордер CD-RW, несмотря на то что для записи CD-DA будут использоваться только матрицы CD-R (об этом ниже).

Основное решение, которое надо принять при выборе привода для записи CD — это тип интерфейса: SCSI, IDE или USB. Интерфейс SCSI, считающийся более профессиональным, предполагает наличие в компьютере соответствующего адаптера. SCSI проще конфигурировать, и он потенциально более эффективен, чем даже последние версии IDE.

IDE-интерфейс имеется в любом PC, где он чаще всего используется для подключения жестких дисков. Нужно учесть, что подключать CD-рекордер желательно к свободному каналу IDE, и лишь в самом крайнем случае — на один шлейф с быстрым винчестером.

Выбор приводов CD-R с интерфейсом USB не так широк, поскольку они начали появляться недавно. Тем не менее, USB-рекордер — практически единственный вариант для портативных компьютеров, а также в тех случаях, когда все четыре возможных IDE-канала заняты.

Скорость записывающих приводов описывается, так же как и читающих, множителем по отношению к скорости стандартного аудио-CD (1x, что соответствует 150 Кб/с). Любой привод проигрывает аудио-CD только на этой скорости; правда, более быстрый привод осуществляет поиск дорожек на звуковом диске несколько быстрее. Максимальная скорость записи обычно меньше, чем чтения, а для перезаписываемых дисков она может быть еще меньше, чем для CD-R. Распространенные записывающие приводы читают на скорости 32x...40x и записывают CD-R на 8x...10x, а CD-RW — на 4x...8x. Можно заметить, что скорость 10x сопоставима со скоростью магнитного накопителя Zip Iomega, при существенно большей емкости компакт-диска.

Высокая скорость не является однозначным преимуществом модели привода. Запись матриц оказывается более надежной на пониженной скорости, что особенно важно для аудио-CD. Так, перед использованием CD-RW может потребоваться его форматирование, занимающее 45 минут. В итоге выигрыш нескольких минут за счет записи при повышенной скорости, скорее всего, будет нецелесообразным. Скорость записи диска на USB-приводе часто ограничена эффективностью интерфейса компьютера, так что иногда приходится снижать скорость записи до 2x, чтобы избежать порчи диска.

Скорость чтения CD-ROM на записывающих приводах тоже не рекордная, но и это на деле оказывается не очень большим недостатком. Чаще всего чтение дисков выполняется всего лишь один раз — для установки программ. Поэтому в компьютере с пишущим приводом более быстрый читающий практически уже не нужен.

Хорошими марками приводов для копирования CD-DA считаются Panasonic, Pioneer, Sony, Teac, Plextor, наименее удачными — Mitsumi и BTC. Очень многие модели поддерживают flash upgrade, так что имеет смысл время от времени посещать веб-сайт изготовителя. Замена прошивки может потребоваться, например, для введения поддержки 80-минутных дисков в более старых моделях.

(Окончание следует)

Б.КИСЕЛЕВ,
г.Минск, БАТУ, каф.ВТ,
тел.264-70-42.

"Живые" книги в MATLAB

Среди богатых возможностей пакета MATLAB [1-5] мы хотим подробней остановиться на приложении Notebook, которое позволяет создавать в редакторе Word электронные книги (**M-книги**) с действующими примерами. Они оформляются как документы класса *Notebooks* с расширением *.doc*. При этом набор и редактирование текста выполняются по правилам Word, а вычисления — по правилам MATLAB. Такие книги особенно полезны в качестве учебно-методического материала, когда обучающийся может сразу поэкспериментировать с предложенными примерами.

Эта тема уже раскрывалась в [1] и, частично, в [3]. Приведенный ниже материал использует MAT-LAB 6 и русифицированный Word 97. Мы полагаем, что читатель знаком с пакетом MATLAB хотя бы в объеме [5], а также владеет простейшими навыками работы в Word.

Описанный способ хорошо работает только при расположении файлов Word и MATLAB в строгом соответствии с инсталляционными рекомендациями. Поэтому надежней пользоваться приемом, описанным в документации.

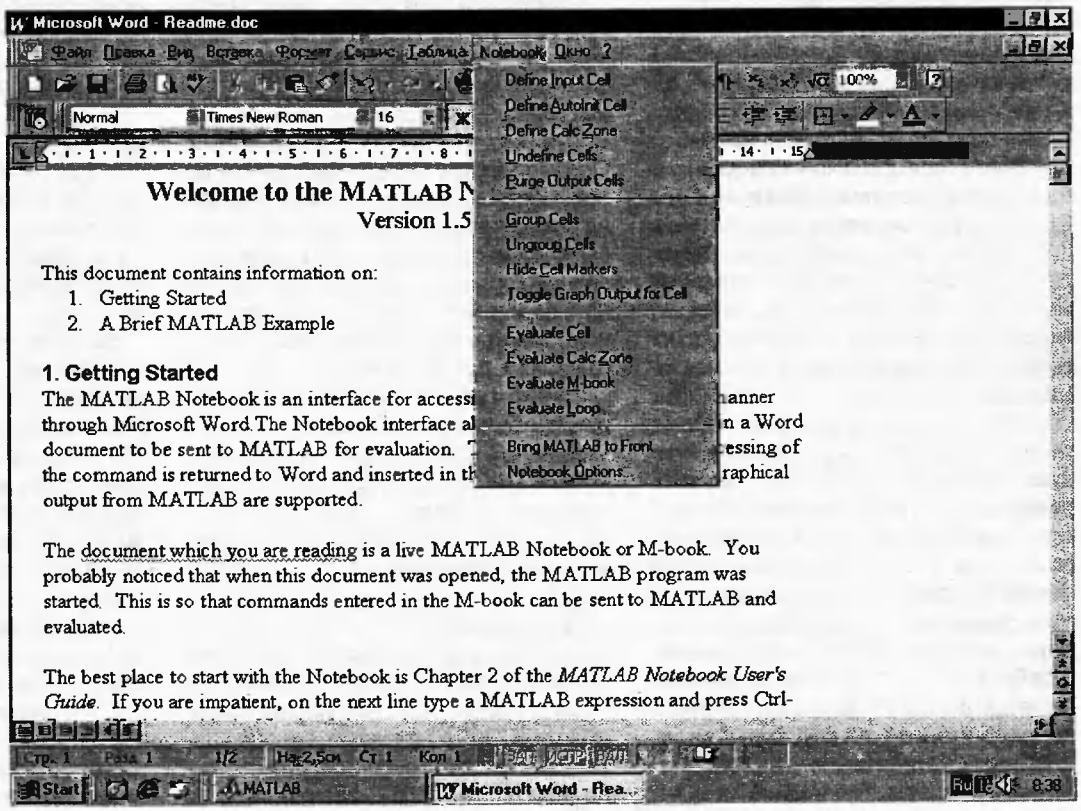
Запустить MATLAB. В командном окне набрать:

```
notebook -setup
```

и нажать **Enter**. Появится предложение выбрать версию редактора Word в виде:

```
Choose your version of Microsoft Word:
[1] Microsoft Word for Windows 95 (Version 7.0)
[2] Microsoft Word 97
[3] Microsoft Word 2000
[4] Exit, making no changes
Microsoft Word Version:
```

Рис. 1



Подготовка к созданию Notebook

Перед началом работы с Notebook (при первом обращении к приложению) необходимо выполнить ряд предварительных действий, чтобы установить связи Word с MATLAB.

Проще всего поступить так, как рекомендуется в [1]. Запустить текстовый редактор Word. В позиции **Файл** главного меню выбрать **Открыть**. В появившемся окне **Открытие документа** найти файл *readme.doc* системы MATLAB, который находится в папке *notebook* или ее дочерних папках, выделить файл и нажать **Открыть**.

В процессе загрузки файла *readme.doc* автоматически произведутся все требуемые настройки для работы с документами класса Notebooks, запустится сам MATLAB, и организуются необходимые связи с ним. В результате в окне редактора появится текст файла *readme.doc*, а в главном меню — дополнительная позиция **Notebook**. На рис.1 представлен вид этого окна с активизированной позицией **Notebook**.

Создание первой книги начинается при загруженном файле *readme.doc* путем выбора нового пункта **New M-book** позиции **Файл** главного меню.

После слова "...Version" надо, в нашем случае, ввести цифру 2 и нажать **Enter**. Появится сообщение о необходимости выбрать в окне-диалоге файл *winword.exe* (обычно *c:\Program Files\Microsoft Office\Office\winword.exe*):

```
You will be presented with a dialog box.  Please use it to select
your copy of the Microsoft Word 97 executable (winword.exe).
Press any key to continue...
```

После нажатия любой клавиши откроется окно для выбора (на рис.2 показан возможный вид этого окна после окончания поиска файла). Необходимо выбрать файл *Winword.exe* и нажать **Open.** В результате появится предложение найти в аналогичном окне папку шаблонов (Templates) Word и выбрать файл *normal.dot* (обычный путь — *c:\Program Files\Microsoft Office\normal.dot*):

```
You will be presented with a dialog box.  Please use it to
select a Microsoft Word template (.dot) file in one of your
Microsoft Word template directories.  We suggest that you specify
your normal.dot file.

Press any key to continue...
```

После нажатия любой клавиши откроется окно-диалог, возможный вид которого представлен на рис.3. Следует выбрать файл *normal.dot* и нажать **Open**. Если все было

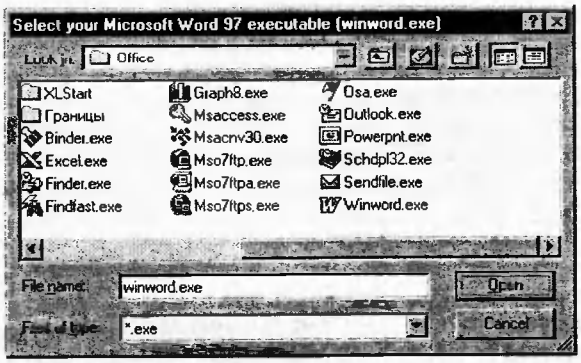


Рис. 2

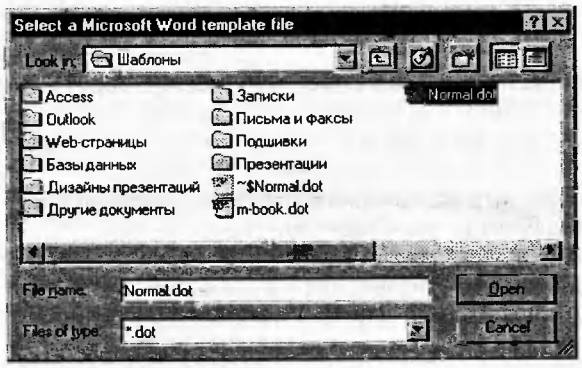


Рис. 3

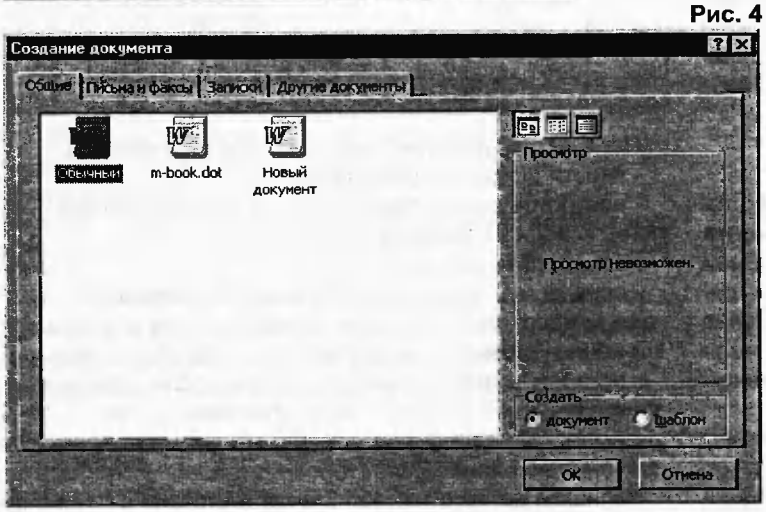


Рис. 4

сделано правильно, то вскоре должно появиться сообщение:

```
Notebook setup is complete.
```

Это говорит о том, что предварительные установки для связи Word и MATLAB выполнены. Теперь, выбирая в главном меню позицию **Файл** и пункт **Создать**, можно увидеть, что появился новый ярлык *m-book.dot* (рис.4), который используется для создания новой М-книги.

На некоторое время вернемся к рис.1 и кратко рассмотрим характеристики опций **Notebook**.

Define Input Cell (Alt+D) — определить ячейку ввода. Для этого необходимо выделить участок текста, предназначенный для ввода в MATLAB, и выбрать пункт **Define Input Cell** или нажать комбинацию клавиш Alt+D. В результате текст будет изображен темно-зеленым жирным шрифтом Courier New размером 10 и заключен в жирные квадратные скобки. Эти скобки называют маркерами. Естественно, текст ячейки должен соответствовать правилам входного языка MATLAB.

Содержимое ячеек ввода отправляется на выполнение выбором пункта **Evaluate Cell**. При этом результат помещается в ячейки вывода. Эти ячейки автоматически появляются при вычислениях после тех ячеек ввода, которые предполагают выдачу некоторого результата. Ячейки вывода выделяются синим цветом и тоже обрамляются жирными квадратными скобками.

Define AutoInit Cell — определить ячейку автостарта. Определенные таким образом участки текста будут выполняться сразу после загрузки М-книги в редактор. После определения они выглядят подобно ячейкам ввода, но выделены темно-синим цветом.

Define Calc Zone — определить зону вычислений. Выделяется часть текста, которая может содержать более одной ячейки ввода, и выбирается **Define Calc Zone**. Появятся маркеры перед зоной и после нее. При выборе соответствующей команды вычисления будут выполняться только для ячеек, попавших в зону выделенного текста.

Заметим, что маркеры могут находиться непосредственно в тексте, но это не всегда удобно, поэтому зону вычислений лучше отделить парой пустых строк. Можно использовать пункт **Hide|Show Cell Markers**.

Undefine Cells — отменить определения ячеек. Текст ячеек преобразуется в обычный текст Word.

Purge Output Cell — удалить ячейки вывода. Удаляются все выделенные ячейки вывода.

Group Cells — сгруппировать ячейки. Все ячейки ввода выделенной части текста объединяются в группу ввода. При этом весь выделенный обычный текст, который находится между ячейками, располагается после группы.

Ungroup Cells — разгруппировать ячейки. Команда, обратная предыдущей. При этом удаляются ячейки вывода, связанные с преобразуемыми ячейками ввода.

Hide|Show Cell Markers — спрятать или показать маркеры ячеек на экране. Отметим, что при печати документа маркеры не печатаются независимо от этой установки.

Evaluate Cell (Ctrl+Enter) — вычислить ячейку. Содержимое текущей ячейки ввода отправляется в MATLAB на выполнение. Текущей является выделенная ячейка, или та, в пределах которой находится курсор мыши, либо он располагается сразу за закрывающим маркером. Если, в соответствии с правилами MATLAB, должны выводиться результаты, то они появятся в ячейке вывода.

Evaluate Calc Zone — выполнить действия в зоне вычислений. Произвести вычисления во всех ячейках ввода и группах ячеек в текущей зоне вычислений.

Evaluate M-book — выполнить М-книгу. Произвести вычисления во всех ячейках ввода и группах ячеек данного файла.

Evaluate Loop — выполнить циклически. Действия в текущей ячейке или группах ячеек выполнить несколько раз. После выбора пункта **Evaluate Loop** появляется одноименное окно (рис.5), где в поле **Stop After** можно задать число повторений.

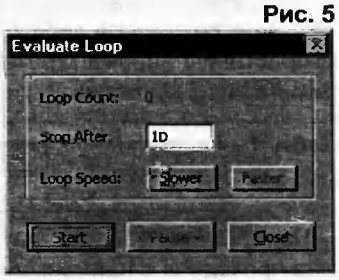


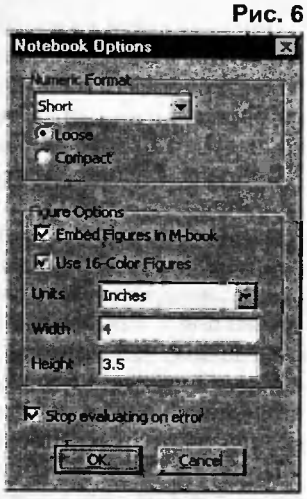
Рис. 5

Кнопками **Slower** и **Faster** можно задать скорость вычислений. Начинаются вычисления кнопкой **Start**, приостанавливаются кнопкой **Pause**. Кнопка **Close** закрывает окно.

Bring MATLAB to Front — вывести на передний план командное окно MATLAB. На экране появляется командное окно MATLAB, и можно выполнять все его функции.

Notebook Options — опции notebook. Выводит соответствующее окно (рис.6), содержащее две панели, и позволяет устанавливать опции, связанные главным образом с выводом результатов.

Рис. 6



Панель **Numeric Format** задает форматы чисел. Опция **Loose** добавляет пробел между ячейками, а опция **Compact** убирает его. Панель **Figure Options** обслуживает графики и рисунки. Опция **Embed Figures in M-book** при установленной галочке означает размещение рисунков прямо в тексте книги. При ее отсутствии — они будут выведены в отдельные окна. Отсутствие галочки в опции **Use 16-color Figures** позволяет отобразить 256 цветов. **Units, Width** и **Height** задают единицу измерения, ширину и высоту рисунков соответственно. Наличие опции **Stop evaluating on error** означает остановку вычислений в случае ошибки.

Создание M-книги

В позиции **Файл** главного меню необходимо выбрать пункт **Создать** — откроется окно **Создание документа** (рис.4). Выделить ярлык *m-book.dot* и нажать **OK**. Если вы продолжаете начатый ранее сеанс работы, то откроется новый документ. Если вы начинаете новый сеанс работы, то после нажатия кнопки **OK** сначала загрузится MATLAB, затем откроется новый документ, и в главном меню появится позиция **Notebook**.

Далее по обычным правилам редактора Word набирается необходимый текст. Можно сначала набрать весь текст, а затем преобразовать программные фрагменты в ячейки ввода, можно в процессе набора текста сразу формировать ячейки ввода и вести их отладку — дело вкуса. Однако необходимо помнить — все, что затем попадет в ячейки ввода, должно удовлетворять правилам языка MATLAB.

В качестве примера рассмотрим фрагменты изучения темы "Метод наименьших квадратов". Тот, кто не знаком с методом, для начала может обратиться к [6].

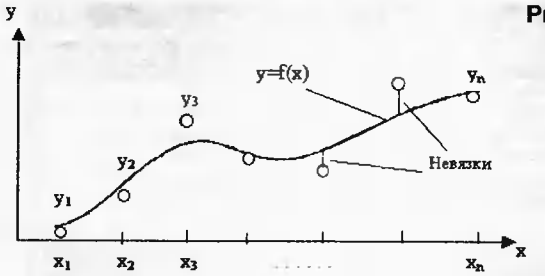
Допустим, нужно определить зависимость (в виде формулы) количества продаваемого товара (*y*) от объема рекламы (*x*). Для этого мы зафиксировали свои наблюдения объема реализованных товаров для различных объемов рекламы в табл.1, расположив результаты по возрастанию объема (*x*). Пары точек x_i, y_i *(i=1,n)* отобразим на рис.7 кружками:

Табл. 1

X	x_1	x_2	...	x_n
Y	y_1	y_2	...	y_n

Наша задача — определить кривую $y=f(x)$ такую, чтобы она проходила с наименьшим отклонением от заданных

Рис. 7



точек. За меру отклонения *S* берут сумму квадратов разностей между значениями функции $f(x_i)$ и y_i.

$$S = \sum_{i=1}^{n} (f(x_i) - y_i)^2. \qquad (1)$$

В качестве искомой зависимости *f(x)* обычно берут многочлены

$$y = f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_mx^m, \qquad (2)$$

где $m < n$.

Коэффициенты многочлена a_j *(j=0,m)* определяются так, чтобы *S* принимало минимальное значение, т.е.

$$S \to min \qquad (3)$$

Отклонения $f(x_i)-y_i$ называются ***невязками***.

Рассмотрим пример. Пусть исходные данные имеют вид, как в табл.2

Табл. 2

X	0,5	1,0	1,5	2,0	2,5	3,0	3,5
Y	2,6	3,8	4,2	4,0	3,6	3,0	2,2

Перечислим действия, которые будут выполнены:

- ввести исходные данные (табл.2);
- построить многочлены первой, второй и третьей степени, исходя из условий (1) и (3);
- определить невязки;
- определить среднеквадратичные отклонения $S_0 = \sqrt{\frac{S}{n}}$ для каждого многочлена, и по ним оценить степень отклонения каждого из многочленов от табличных точек;
- построить графики многочленов в табличном диапазоне изменений *x* с указанием табличных точек;
- вывести таблицу невязок для наших многочленов и значения величин S_0.

Запишем данные из табл.2 в соответствии с правилами входного языка MATLAB, т.е. определим векторы *x* и *y*:

```
x=[0.5 1.0 1.5 2.0 2.5 3.0 3.5]
y=[2.6 3.8 4.2 4.0 3.6 3.0 2.2]
n=7
```

Выделим эти три строчки. В позиции **Notebook** главного меню выберем **Define Input Cell**, получим:

```
[x=[0.5 1.0 1.5 2.0 2.5 3.0 3.5]
 y=[2.6 3.8 4.2 4.0 3.6 3.0 2.2]
 n=7]
```

Напомним, что содержимое ячеек ввода выделено зеленым цветом, а ячеек вывода — синим. Затем выберем **Evaluate Cell** (Ctrl+Enter), и появится ячейка вывода со значениями введенных элементов:

```
[x =
    0.5000    1.0000    1.5000    2.0000    2.5000
3.0000    3.5000
 y =
    2.6000    3.8000    4.2000    4.0000    3.6000
3.0000    2.2000
 n =
     7]
```

Если при определении *x*, *y* и *n* в конце выражений поставить ";", то после выбора **Evaluate Cell** ячейка вывода

не появится, хотя вычисления выполнятся (значения будут располагаться в рабочей области), т.е. все действия происходят в соответствии с правилами MATLAB.

Определим коэффициенты многочленов, удовлетворяющих условиям (1) и (3). Это можно сделать с помощью встроенной функции *polyfit(x,y,m)*, где *x*, *y* — массивы точек таблицы, а *m* — степень многочлена:

```
cd1=polyfit(x,y,1);
cd2=polyfit(x,y,2);
cd3=polyfit(x,y,3);
```

Здесь *cd1*, *cd2*, *cd3* — массивы коэффициентов соответствующих многочленов. Поместим эти три оператора в ячейку ввода с помощью **Define Input Cell**, получим:

```
[cd1=polyfit(x,y,1);
 cd2=polyfit(x,y,2);
 cd3=polyfit(x,y,3); ]
```

Можно было бы вычислять каждую ячейку ввода и выполнять отладку, но мы большинство следующих ячеек объединим в одну зону вычислений.

Теперь определим значения наших многочленов в табличных точках, чтобы можно было вычислить невязки. Для этого используем встроенную функцию *polyval(p,x)*, где *p* — массив коэффициентов многочлена, *x* — массив точек, в которых нужно вычислить многочлен. Сформируем ячейку ввода:

```
[Ym(1,:)=polyval(cd1,x);
 Ym(2,:)=polyval(cd2,x);
 Ym(3,:)=polyval(cd3,x); ]
```

Здесь `Ym` — двумерный массив значений многочленов в табличных точках: первая строка массива содержит значения многочлена первой степени, вторая — второй степени и т.д.

Определим ячейку ввода для вычисления массива `S0` среднеквадратичных отклонений:

```
[for i=1:3; % значение i соответствует степени многочлена
  S0(i)=sqrt(sum((y-Ym(i,:)).^2)/n);
 end]
```

Для наглядности построим графики многочленов. Предварительно определим массив `Xgr`, который содержит значения *x* в табличном диапазоне (от 0,5 до 3,5), и массив `Ygr`, содержащий значения многочленов в точках `Xgr`. Чтобы кривые на графике были гладкие, точек надо взять достаточно много. Будем, например, менять *x* с шагом 0,1. Определим соответствующую ячейку ввода:

```
[Xgr=0.5:0.1:3.5; % абсциссы графика
 Ygr(1,:)=polyval(cd1,Xgr); % ординаты многочлена 1-й степени
 Ygr(2,:)=polyval(cd2,Xgr); % ординаты многочлена 2-й степени
 Ygr(3,:)=polyval(cd3,Xgr); % ординаты многочлена 3-й степени]
```

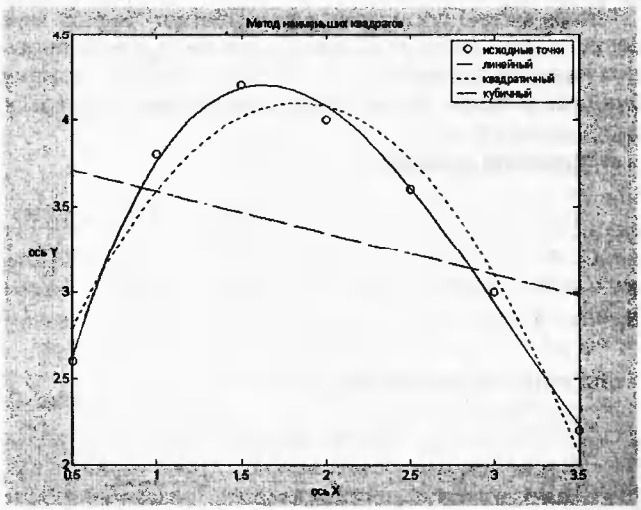
Построение графиков осуществим функцией *plot* [1-5]. Соответствующая ячейка:

```
[plot(x,y,'ko',Xgr,Ygr(1,:),'k-',Xgr,Ygr(2,:),'k:',...
      Xgr,Ygr(3,:),'k-'); % первые три параметра - точки таблицы
 legend('исходные точки','линейный','квадратичный','кубичный');
 title('Метод наименьших квадратов');
 xlabel('ось X'); ylabel('ось Y'); ]
```

Выполним действия, набранные в ячейках ввода. Для этого выделим весь текст вместе с ячейками от абзаца "Здесь cd1, cd2, cd3..." до данного абзаца, и сформируем зону вычислений, выполнив **Define Calc Zone**. Затем выберем **Evaluate Calc Zone**. В результате получим график, представленный на рис.8. Заметим, что в цвете его можно сделать намного эффектней.

Если размеры изображения графиков нас не устраивают, их можно менять обычными приемами работы в Word.

Рис. 8



Вывод оформим с помощью команды *disp* и функции *sprintf* [1, 4] (она работает аналогично функции printf языка Си), поместим его в ячейку ввода, и выполним **Evaluate Cell**:

```
[disp('                НЕВЯЗКИ:');
 disp('      x      y-Ym(1)   y-Ym(2)   y-Ym(3)');
 for i=1:n;
   str=sprintf('%8.4f %8.4f %8.4f %8.3f',...
       x(i),Ym(1,i)-y(i),Ym(2,i)-y(i),Ym(3,i)-y(i));
   disp(str);
 end;
 disp('              ср.кв.отклонение:');
 disp('       S1          S2          S3');
 disp(S0); % ср.кв.отклонение]
```

```
[          НЕВЯЗКИ:
     x      y-Ym(1)   y-Ym(2)   y-Ym(3)
  0.5000    1.1071    0.1905    0.024
  1.0000   -0.2143   -0.2143   -0.048
  1.5000   -0.7357   -0.1857   -0.019
  2.0000   -0.6571    0.0762    0.076
  2.5000   -0.3786    0.1714    0.005
  3.0000    0.1000    0.1000   -0.067
  3.5000    0.7786   -0.1381    0.029
              ср.кв.отклонение:
        S1          S2          S3
    0.6551      0.1608      0.0452 ]
```

Видно, что среднеквадратичное отклонение уменьшается с увеличением степени многочлена.

Если требуется более эстетично оформить расчеты, то проще отредактировать результат вывода средствами Word, чем изощряться возможностями MATLAB. Например, можно выделить последнюю ячейку вывода и выполнить **Undefine Cells**. Получившийся текст легко преобразовать в таблицу, используя позицию **Таблица** главного меню и ее пункт **Преобразовать в таблицу.** Один из таких вариантов может выглядеть следующим образом:

НЕВЯЗКИ:

X	Y-YM(1)	Y-YM(2)	Y-YM(3)
0.5000	1.1071	0.1905	0.024
1.0000	-0.2143	-0.2143	-0.048
1.5000	-0.7357	-0.1857	-0.019
2.0000	-0.6571	0.0762	0.076
2.5000	-0.3786	0.1714	0.005
3.0000	0.1000	0.1000	-0.067
3.5000	0.7786	-0.1381	0.029

ср.кв.отклонение:

S1	S2	S3
0.6551	0.1608	0.0452

Работа с М-книгой

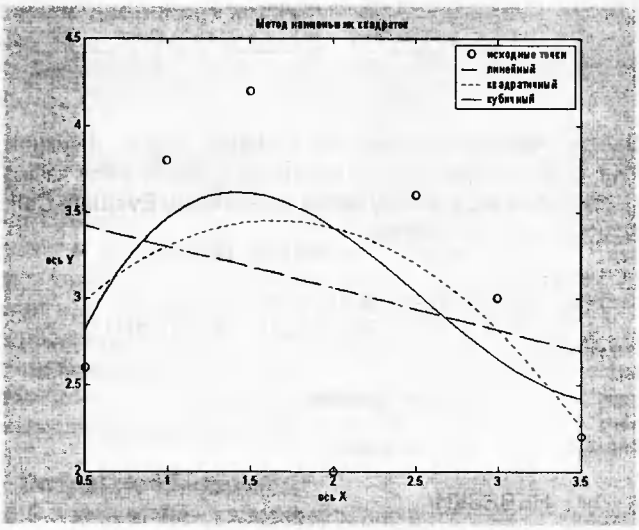
Теперь можно посмотреть, как работает "живая" книга целиком. Для этого, например, изменим в самой первой ячейке ввода значение *y* с 4,0 на 2,0 и выполним команду **Evaluate M-book.** В результате содержимое ячеек вывода поменяется на

для первой ячейки:

```
x =
    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000
3.5000
y =
    2.6000    3.8000    4.2000    2.0000    3.6000    3.0000
2.2000
n =
     7
```

для второй ячейки (рис.9):

Рис. 9



для третьей ячеки:

НЕВЯЗКИ:

x	y-Ym(1)	y-Ym(2)	y-Ym(3)
0.5000	0.8214	0.3810	0.214
1.0000	-0.5000	-0.5000	-0.333
1.5000	-1.0214	-0.7571	-0.590
2.0000	1.0571	1.4095	1.410
2.5000	-0.6643	-0.4000	-0.567
3.0000	-0.1857	-0.1857	-0.352
3.5000	0.4929	0.0524	0.219

ср.кв.отклонение:

S1	S2	S3
0.7372	0.6711	0.6531

Надеемся, что эта статья поможет без особых проблем освоить оставшиеся возможности создания М-книг самостоятельно. Также заметим, что этот материал был выполнен в виде М-книги, и затем незначительно отредактирован.

Литература

1. Дьяконов В.П., Абраменкова И.В. MATLAB 5.0/5.3. Система символьной математики. — М.: Нолидж, 1999, 640 с.

2. Дьяконов В.П. MATLAB: учебный курс. — СПб.: Питер, 2001, 510 с.

3. Лазарев Ю.Ф. MatLAB 5.x. — К.: Издательская группа BHV, 2000, 384 с.

4. Мартынов Н.Н., Иванов А.П. MATLAB 5.x. Вычисления, визуализация, программирование. — М.: КУДИЦ-ОБРАЗ, 2000, 336 с.

5. Нечаев С. MATLAB. — Радиолюбитель. Ваш компьютер, 1998, №№9-12; 1999, №№3, 6.

6.Турчак Л.И. Основы численных методов: Учеб.пособие. — М.: Наука. Гл.ред. физ.-мат. лит., 1987, 320 с.

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

Работа в системе Mathematica 3.0/4.0.

Операции математического анализа

В практических расчетах львиная доля вычислений приходится на операции интегрирования и дифференцирования. В данной статье мы их рассмотрим в той же последовательности и с теми же вспомогательными элементами, что и в любом стандартном курсе математического анализа: пределы, дифференцирование, интегрирование, ряды.

Вычисление пределов

Вычисление пределов функций — простой, но вместе с тем и наглядный пример превосходства систем символьной математики. Численные математические системы, например, не считают, что выражение 0/0 несет какую-либо смысловую нагрузку. При вычислении выражения *sin(x)/x* при *x=0* как правило появляется сообщение об ошибке вида "деление на 0". Для вычисления пределов в Mathematica существует команда **Limit**. Эта команда работает как с простейшими пределами (**Limit[x^2, x→1]** возвращает результат 1), так и с неопределенностями вида 0/0 (**Limit[Sin[x]/x, x→0]** возвращает 1). Так же легко вычисляются пределы и в не столь очевидных ситуациях,например — **Limit[(x^x-a^a)/(a^x-x^a), x→a]**. Еще одна особенность систем символьной математики — переменная может стремиться к значению "бесконечность": **Limit[n/(n+1), n→∞]**. Для набора символа ∞ можно воспользоваться палитрой, или набрать **Esc i n f Esc**, или просто **Infinity**. Разрывные функции в точке разрыва могут иметь различные значения пределов — т.н. левосторонние и правосторонние пределы. Левосторонний предел в Mathematica задается опцией **Direction→+1**, правосторонний — **Direction→ -1**. Пример использования: **Limit[1/x, x→0, Direction→+1]** возвращает **-∞**, **Limit[1/x, x→0, Direction→-1]** возвращает **∞**.

Дифференцирование

К числу наиболее часто используемых математических операций относится также и дифференцирование функций. При наборе с палитры первая производная выглядит как ∂_x**Sin[x]**. Эта же команда может быть записана в виде **D[Sin[x], x]**. Для вычисления второй (или более высокой степени) производной используется та же команда, только вместо указания переменной используется более сложная конструкция — **{имя_переменной, порядок}**. Пример вычисления второй производной: $\partial_{\{x,\,2\}}$**Sin[x]** — с использованием палитры; **D[Sin[x], {x, 2}]** — при наборе с клавиатуры.

Для функции многих переменных предусмотрена возможность вычисления смешанных производных. И в этом случае Mathematica проявляет последовательность и логичность в обозначениях. Несложно догадаться, что выражение $\partial_{\{x,\,2\},\{y,\,3\}}$**Sin[x+y]** или **D[Sin[x+y], {x, 2},{y, 3}]** означает вторую производную функции **Sin[x+y]** по переменной **x** и третью — по **y**.

Список команд дифференцирования этими примерами, естественно, не исчерпывается. Для вычисления полной производной существует команда **Dt**. В этом случае подразумевается, что все символьные переменные зависят от той переменной, по которой вычисляется производная. Так, **Dt[x*y, x]** приводит к результату **y+x Dt[y, x]**. По прежней схеме вычисляются производные высших порядков — **Dt[x*y, {x, 2}]**. Если же вы хотите, чтобы какие-либо из величин все-таки рассматривались как константы, это можно указать следующим образом — **Dt[x^2 + z^2, x, Constants → {z}]**.

Интегрирование

Вычисление интегралов — предмет особой гордости системы Mathematica. В рекламных целях фирма Wolfram Research предлагает на сайте integrals.wolfram.com испытать возможности системы в режиме on-line. По числу вычисляемых интегралов Mathematica сравнима с самыми полными математическими справочниками. Конечно, никто не застрахован от ошибок, но расхождение результатов не всегда говорит об ошибках системы — на сайте фирмы-разработчика www.wolfram.com собраны десятки примеров, свидетельствующих об обратном.

При наборе с палитры (BasicInput) команда для вычисления неопределенного интеграла выглядит следующим образом:

$$\int Sin[x]dx.$$

Ту же команду можно набрать и с клавиатуры — **Integrate[Sin[x], x]**. В качестве первого аргумента указывается подынтегральное выражение, второй аргумент — переменная, по которой производится интегрирование. Вычисление определенного интеграла в Mathematica производится аналогично, только либо на палитре выбирается шаблон с полями для верхнего и нижнего пределов интегрирования, либо пределы указываются вместе с переменной — **Integrate[Sin[x], {x, 0, 1}]**.

Особого внимания заслуживает интегрирование разрывных функций. В некоторых случаях, таких как **Integrate[x/(1-x), {x, 0, 3}]**, интеграл, строго говоря, не определен, и Mathematica возвращает сообщение об ошибке. Однако в этом случае может быть определено главное значение интеграла по Коши. Система справляется с вычислением данной величины при задании дополнительной опции — **Integrate[x/(1-x), {x, 0, 3}, PrincipalValue → True]**.

Встречаются ситуации, когда сходимость интеграла зависит от значения некоторого параметра. Например, интеграл $\int_0^\infty e^{ax}dx$ сходится, если действительная часть a меньше нуля. Со свойственной ей дотошностью Mathematica выдает полное сообщение — **If[Re[a] < 0, -(1/a), Integrate[Exp[a x], {x, 0, ∞}]]**. При желании можно отказаться от генерации сообщения об условиях сходимости: **Integrate[Exp[a x], {x, 0, ∞}, GenerateConditions → False]**. Хотя более корректный способ — указать ограничения, налагаемые на параметр — **Integrate[Exp[a x], {x, 0, ∞}, Assumptions → {a < 0}]**.

Работа с кратными интегралами также ничем принципиально не отличается от рассмотренных выше примеров: **Integrate[x^2+y^2, {x, 0, 1}, {y, 0, 1}]**. Отметим только, что в качестве пределов могут использоваться не только числа и символьные переменные, но и функции — **Integrate[x^2+y^2, {x, Sin[y], Cos[y^2]}, {y, 0, 1}]**.

Естественно, далеко не все интегралы выражаются через известные функции. Поэтому в системе предусмотрены команды численного интегрирования —**NIntegrate[Sin[Sin[x]], {x, 0, 1}]**. Mathematica самостоятельно выбирает метод интегрирования и производит контроль точности. Однако, можно явно указать системе метод численного интегрирования, **NIntegrate[Sin[Sin[x]], {x, 0, 1}, Method →MonteCarlo]** — вычисление с использованием метода Монте-Карло.

Приведем некоторые другие методы, используемые системой Mathematica: **GaussKronrod** (Гаусса-Кронрода), **MultiDimensional** (Генца-Малика) — используются по умолчанию, **DoubleExponential** — алгоритм двойной экспоненциальной сходимости. Для задания точности вычислений используются опции **WorkingPrecision** — задание рабочей точности (числа значащих цифр в промежуточных вычислениях) и **PrecisionGoal** — задание требуемого числа значащих цифр в окончательном ответе (естественно, оно не может превышать **WorkingPrecision**). Пример вычисления интеграла с повышенной точностью — **NIntegrate [Sin[Sin[x]], {x, 0, 1}, WorkingPrecision → 30, PrecisionGoal → 20]**.

При вычислении неопределенных интегралов с использованием систем компьютерной математики часто встречается ситуация, когда производная от первообразной функции не совпадает с исходной функцией. Простой пример: **D[Integrate[1/(1-Exp[-x]), x], x]**. Но в подавляющем большинстве случаев это свидетельствует не об ошибке, а о разных формах записи одной и той же функции (часто ситуация проясняется после применения команды **FullSimplify**).

Работа с рядами

Разложение функции в ряд осуществляется при помощи команды **Series**. Разложение экспоненты в окрестности точки **x=0** с точностью до члена девятого порядка включительно выглядит как **Series[Exp[x], {x, 0, 9}]**. Аналогично получается разложение в ряд по двум перемен-

ным — **Series[Exp[x y], {x, 0, 9}, {y, 0, 9}]**. Особый случай — разложение в окрестности точки x=∞. На выходе получается асимптотический ряд, например, при разложении факториала можно получить формулу Стирлинга с поправочными членами **Series[x!, {x, ∞, 3}]**. Но в таких разложениях аргумент следует выбирать максимально простым — вычисление **Series[(a x)!, {x, ∞, 3}]** вызывает сообщение об ошибке.

Полученные разложения можно перемножать, использовать в качестве аргумента математической функции (Mathematica корректно обращается с остаточным членом). При этом имеются средства для работы непосредственно с рядами: команда **InverseSeries** — вычисляет по заданному ряду обратный (разложение обратной функции), **Compo-seSeries** — находит композицию рядов (в качестве аргумента в первый ряд подставляется второй ряд).

Для удаления остаточного члена ряда используется команда **Normal**: **Se-ries[Exp[x], {x, 0, 9}]//Normal** или **Normal[Series[Exp[x], {x, 0, 9}]]** — ряд конвертируется в полином.

Кроме разложения функции в ряд, Mathematica выполняет и обратную операцию — суммирование рядов (как конечных, так и бесконечных). Для символьного суммирования ряда используется команда **Sum**, численное суммирование осуществляется командой **NSum**. Пример применения первой из этих команд — **Sum[x^n/(n!), {n, 0, ∞}]** — суммирование по **n** от **0** до ∞; **Sum[x^n/(n!), {n, 0, ∞, 2}]** — то же самое, но с шагом **2** по **n**. Достаточно близки к этим командам команды для вычисления произведений **Prod** и **Nprod** — **Prod[(x+i), {i, 1, 4}]**.

Основные команды, рассмотренные в данной статье, приведены в таблице. Конечно, всем этим вычислениям не хватает наглядности, поэтому тема нашей следующей статьи — работа с графикой, а также импорт и экспорт графических файлов.

Действие	Реализация
Пределы	
1) Вычисление предела	**Limit**[(Sqrt[4 + x + x^2] - 2)/(x + 1), x→-1] **Limit**[n/(n + 1), n → ∞]
2) Вычисление одностороннего предела	**Limit**[Tan[x], x → Pi/2, **Direction** → 1] **Limit**[Tan[x], x → Pi/2, **Direction** → -1]
Дифференцирование	
1) Вычисление производной функции	∂ **Sin[x]** **D**[Sin[x], x]
2) Вычисление производной высшего порядка	$\partial_{\{x,2\}}$ **Sin[x]** **D**[Sin[x], {x,2}]
3) Вычисление смешанных производных	$\partial_{x,\{y,2\}}$ **Sin[x y]** **D**[Sin[x y], x, {y,2}]
4) Вычисление полной производной	**Dt**[x^2*y^3, {x, 2}] — вычисление производной в предположении, что все переменные зависят от x **Dt**[x^2 + z^2, x, Constants → {z}] — вычисление производной в предположении, что все переменные за исключением z зависят от x
Интегрирование	
1) Вычисление неопределенного интеграла	Набрать с помощью палитры $\int x^n dx$ или **Integrate**[x^n, x]
2) Вычисление определенного интеграла	Набрать с помощью палитры $\int_{-1}^{1} \frac{3Cos[5x]}{E^{2x}} dx$ или **Integrate**[3 Cos[5 x]/Exp[2 x], {x,-1,1}]
3) Вычисление повторного или кратного интеграла	Набрать с помощью палитры $\int_{-1}^{1} (x^2 + y^2) dx$ Выделить при помощи мыши полученное выражение Щелчок по кнопке  на палитре BasicInput Ввести пределы интегрироеания и переменную интегрирования или **Integrate**[**Integrate**[x^2+y^2,{x,-1,1}], {y,-1, 1}] или **Integrate**[x^2+y^2, {x,-1,1}, {y,-1, 1}]
4) Интегрирование с условиями	**Integrate**[Exp[a x],{x,0,Infinity}, **Assumptions**→{a<0}]
5) Интегрирование без проверки условий сходимости	**Integrate**[Exp[a*x], {x, 0, Infinity}, **GenerateConditions** → False]
6) Интегрирование разрывных функций (в смысле главного значения по Коши)	**Integrate**[Sqrt[x]/(x - 1), {x, 0, 3}, **PrincipalValue** → True]
7) Численное интегрирование	**NIntegrate**[Sin[Sin[x]], {x, 0, 1}]
8) Численное интегрирование с указанием метода вычисления	**NIntegrate**[Sin[Sin[x]], {x, 0, 1},**Method** → MonteCarlo] — с использованием метода Монте-Карло
	Некоторые другие методы: GaussKronrod (Гаусса-Кронрода), MultiDimensional (Генца-Малика) — используются по умолчанию; DoubleExponential — алгоритм даойной экспоненциальной сходимости
Ряды	
1) Разложение в ряд	**Series**[Sin[x], {x, 0, 9}] **Series**[f[x], {x, x0, n}] — возвращает первые n членов разложения функции f(x) в ряд по x в окрестности точки x0 и остаточный член
2) Разложение в ряд по нескольким переменным	**Series**[Sin[x*y], {x, 0, 7}, {y, 0, 7}]
3) Обращение ряда	s=**Series**[Sin[x], {x, 0, 9}] — разложение в ряд, присваивание переменной s значения "ряд **Series**[...]" **InverseSeries**[s, x] — обращение ряда s (т.е. получение ряда обратной функции, в данном случае — ряд арксинуса)
4) Композиция рядов	sinseries = **Series**[Sin[x], {x, 0, 7}] cosseries = **Series**[x*Cos[x], {x, 0, 7}] **ComposeSeries**[sinseries, cosseries]
5) Отбрасывание остаточного члена ряда	**Normal**[**Series**[Sin[x], {x,0,5}]] или **Series**[Sin[x], {x,0,5}]// **Normal**
Суммы и произведения	
1) Вычисление суммы	$\sum_{n=0}^{7} \frac{x^n}{n!}$ **Sum**[x^n/n!, {n, 0, 7}] **Sum**[x^n/n!, {n, 0, 7, 2}] — вычисление суммы с шагом 2 по n
2) Численное вычисление суммы	**NSum**[x^n/n!, {n, 0, 7}]
3) Вычисление произведения	$\prod_{i=1}^{4} (x+i)$ — набор с палитры **Prod**[(x+i), {i, 1, 4}]
4) Численное вычисление произведения	**NProd**[j, {j, 1, 10}]

В.ЛУТКОВСКИЙ, Д.МИРОНОВ, П.НАЗАРОВ,
220116, Минск-116, а/я 202.

Проектирование нейронной сети для задач аппроксимации

Мы начинали с задач классификации и распознавания образов [1], где преимущества нейронных сетей неоспоримы. В этой статье пойдет речь о задаче аппроксимации экспериментальных данных функциями, алгоритмы решения которых на обычных компьютерах достаточно хорошо разработаны и давно известны. Тем не менее, нейронная сеть, на наш взгляд, решает эту задачу проще и изящнее. Тех, кто начинает грустить при виде формул, мы отсылаем непосредственно к примерам, которые рассмотрим дальше. Они помогут получить результат, не вдаваясь во все тонкости построения и обучения нейронной сети. Тех же, кто уже освоил работу с персептронами [2], мы познакомим с еще одним видом нейронных сетей.

Итак, рассмотрим постановку задачи аппроксимации.

Задача. *По заданному множеству N точек на плоскости $\{x_k, y_k\}$ (k=1...N), полученных из эксперимента, необходимо построить функцию y=f(x), такую, чтобы сумма квадратов отклонений $\Delta_i = y_i - f(x_i)$ была минимальной.*

Таким образом, имеются отсчеты функции в отдельных точках x_i (узлах), система базисных функций и вектор регулируемых весовых коэффициентов w_i при m базисных функциях $F_i(x)$. Необходимо обучить функцию путем настройки весовых коэффициентов таким образом, чтобы их комбинация давала аналитическую зависимость, которая наилучшим образом вписывается в множество экспериментальных данных (или отсчетов функции в узлах):

$$y(x) = \sum_{i=1}^{m} w_i F_i(x).$$

Эту задачу можно переформулировать как задачу контролируемого обучения ("с учителем") нейронной сети. В качестве обучающих пар при этом используются экспериментальные данные или значения функции в узлах. Число входов нейронной сети определяется числом независимых аргументов. Выходом сети как правило является скалярная величина — значение функции (зависимая переменная). В более общем случае входные и выходные величины могут быть векторными. Если мерой отклонения служит среднеквадратичная ошибка, то полученная аналитическая зависимость $y(x)$ называется *линией регрессии*.

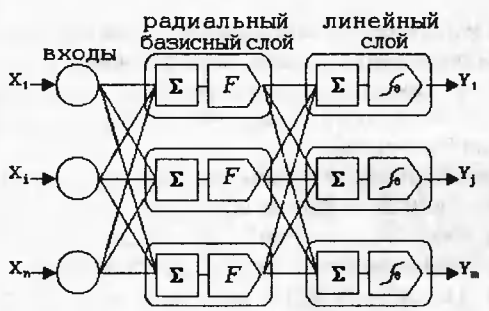
В принципе, для решения этой задачи можно использовать многослойный персептрон, обучаемый методом обратного распространения ошибки, но лучше с ней справляются сети на основе радиально-базисных функций (РБФ-сети).

Заметим, что до сих пор мы рассматривали нейронные сети, состоящие из одинаковых нейронов. Они наиболее просты для понимания и удобны в применении. Действительно, проектировать конструкции, состоящие из одинаковых элементов — дело нехитрое, но утомительное, если этих элементов очень много. Например, можно построить дом только из кирпичей, но удобнее строить его из специальных блоков и панелей. При этом необходимо заранее позаботиться об их совместимости и точности установки отдельных блоков.

В качестве строительного материала для первого слоя мы будем использовать набор функций специального вида, а во втором слое — обычные нейроны. В результате мы придем к новому классу нейронных сетей — так называемым сетям на основе радиально-базисных функций (РБФ) [3].

РБФ-сеть содержит промежуточный (скрытый) слой радиальных базисных нейронов и выходной слой с нейронами, из которых обычно строят персептроны. Каждый нейрон в радиальном базисном слое использует в качестве активационной функции специальную радиально-базисную функцию (отсюда и берет свое название этот тип нейронных сетей). Сигнал с выходов радиальных элементов поступает на выходной слой, где каждый выходной элемент вычисляет линейную комбинацию этих радиальных базисных функций. С точки зрения задачи аппроксимации, скрытые элементы формируют совокупность функций, которые образуют базисную систему для представления входных примеров в построенном на ней пространстве. Структура простейшей РБФ-сети приведена на рис.1.

Рис. 1



Радиальная базисная функция — это непрерывная функция с максимумом или минимумом в некоторой точке, называемой центром, значение которой для произвольной точки монотонно изменяется в зависимости от расстояния (от этой точки до центра).

Такими свойствами обладают следующие функции:

- гаусиан (функция Гаусса) — $F(\varsigma) = \exp(-\frac{\varsigma^2}{2\sigma^2})$;
- мультиквадратичная — $F(\varsigma) = \sqrt{\varsigma^2 + \sigma^2}$;
- обратная мультиквадратичная — $F(\varsigma) = \frac{1}{\sqrt{\varsigma^2 + \sigma^2}}$.

Во всех трех случаях параметр σ определяет радиус влияния каждой базисной функции и быстроту затухания функции при удалении от центра.

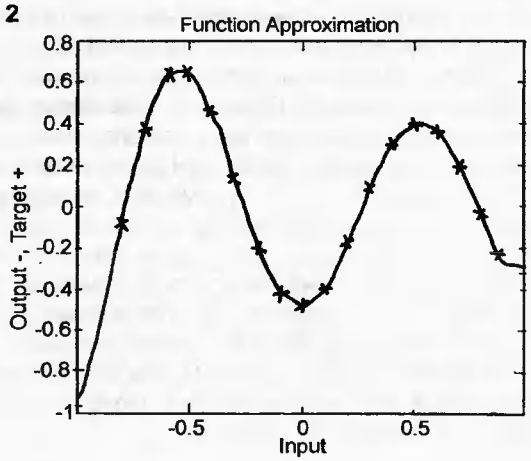
Обучение РБФ-сети

Напомним, что обычно используют два способа обучения: **контролируемое** ("с учителем") и **неконтролируемое** (самообучение "без учителя").

В первом случае нейронная сеть располагает правильными ответами (выходами сети) на каждый входной пример. Веса настраиваются так, чтобы сеть производила ответы, как можно более близкие к известным правильным ответам. Усиленный вариант обучения с учителем предполагает, что известна только критическая оценка правильности выхода нейронной сети, но не сами правильные значения выхода.

Во втором случае не требуется знания правильных ответов на каждый пример обучающей выборки. В этом случае результат обучения определяется внутренней структурой данных или корреляцией между образцами в системе данных, что позволяет распределить образцы по категориям.

Рис. 2



Реже используется **смешанное** обучение, когда часть весов определяется посредством обучения "с учителем", в то время как остальная получается с помощью самообучения. Именно этот вариант часто используется при обучении РБФ-сетей.

Основной алгоритм использует двухшаговую стратегию обучения, или смешанное обучение. Сначала происходит обучение "без учителя". На данном этапе используется информация только о структуре подаваемых на вход данных, т.е. определяются центры и ширина для радиальных элементов. На втором этапе идет уже обучение "с учителем" — оптимизируются параметры линейного выходного слоя, т.е. определяются веса связей между скрытым и выходным слоями. Такой алгоритм обучения РБФ-сети сходится гораздо быстрее, чем алгоритмы обучения многослойных персептронов. Поэтому сеть РБФ обучается очень быстро (на порядок быстрее многослойных персептронов). Однако РБФ-сеть часто содержит слишком большое число скрытых элементов. Это является причиной более медленного функционирования РБФ-сети, чем многослойного персептрона. Эффективность (величина ошибки в зависимости от размера сети) РБФ-сети и многослойного персептрона зависит от решаемой задачи.

Заметим, что РБФ-сети можно рассматривать как частный случай двухслойных сетей с прямым распространением информации — от входа к выходу. Чтобы убедиться в преимуществах данного класса нейронных сетей, воспользуемся демонстрационным примером DEMORB2 из уже знакомого нам пакета MatLab/Neural Network Toolboks [2]. На рис.2 приводится вид аппроксимирующей функции, где крестиками обозначены исходные данные.

В таблице указано время, необходимое для обучения четырех различных конфигураций нейронных сетей — длительность обучения для указанных нейронных сетей составила 1489, 273, 8 и 6 итераций соответственно. Из таблицы видно, что РБФ-сеть обучается в 40 с лишним раз быстрее, чем многослойный персептрон, обучаемый методом Backpropagation! Другими словами, РБФ-сеть по скорости обучения значительно превосходит равноценный по интеллекту персептрон.

Тип нейронной сети	Процедура обучения	Время обучения, с	Метод обучения
Многослойный персептрон	TRAINBP	106,5	Backpropagation
Многослойный персептрон	TRAINBPX	20,3	Faster Backprop
Многослойный персептрон	TRAINLM	5,1	Levenberg-Marquardt
РБФ-сеть	SOLVERB	2,6	Radial Basis Network

В заключение заметим, что пока мы не касались сетей с обратными связями или связями в пределах одного слоя. Речь о них пойдет в следующих статьях.

Литература

1. Лутковский В.М, Лашина Т.А., Назаров П.В. Проектирование нейронной сети: Задачи классификации и распознавания. — Радиолюбитель. Ваш компьютер, 2001, №1, С.15.

2. Лутковский В.М, Назаров П.В. Проектирование нейронной сети. — Радиолюбитель. Ваш компьютер, 2001, №№2-7.

3. Bishop M. Neural Networks for Pattern Recognition. — Oxford: Clarendon Press, 1997.

Рейтинговые поединки в компьютерных шахматах

С.РЮМИК,
г.Чернигов-33,
а/я-1772.

Сущность вещей есть число.
Пифагор

Языку цифр подвластно все. Можно подсчитать количество генов в молекуле ДНК человека, оценить число атомов во Вселенной, определить расстояние от Земли до Альфа Центавра. Шахматы — не исключение, хотя до недавнего времени казалось, что эта игра, которая по определению М.Ботвинника "иногда становится искусством", не сможет быть полностью "оцифрована". Тем не менее, спортивные соревнова-

ния должны проводиться максимально объективно, в связи с чем международная шахматная федерация ФИДЕ ввела систему индивидуальных коэффициентов ЭЛО, которая закрепила статус современных шахмат как полноправного вида спорта.

Первые попытки построить систему оценок шахматного мастерства относятся к началу XX века. В 1939 г. российским шахматистом С.Зефировым была предложена общая концепция, а в конце 50-х годов начались практические испытания нескольких систем, основанных на том, что каждому шахматисту присваивается меняющийся от турнира к турниру индивидуальный коэффициент — рейтинг (от англ. "rating" — оценка).

В 1970 г. ФИДЕ официально утвердила в качестве основной рейтинговую систему профессора физики Арпада Имре Эло, венгра по национальности, эмигрировавшего в США. Рассчитанные 35 лет назад по его системе рейтинги первых семи чемпионов мира приведены в табл.1. Для сравнения, на сегодняшний день планка международного гроссмейстера установлена на уровне 2600 ЭЛО, значит, система рейтинговых оценок выдержала проверку временем и не потеряла свою актуальность.

Табл. 1

Чемпионы мира по шахматам	Рейтинг (макс.)	Год
1 Стейниц В.	2650	1880
2 Ласкер Э.	2720	1906
3 Капабланка Х.	2730	1920
4 Алехин А.	2690	1933
5 Эйве М.	2650	1936
6 Ботвинник М.	2730	1945
7 Смыслов В.	2700	1952

Если приведенные рассуждения справедливы, то логично распространить действие рейтингов и на соревнования компьютерных шахматных программ. Стремительный рост производительности компьютеров и совершенствование алгоритмов расчета привели к тому, что сейчас, имея обычный Pentium-II-450 и 128 Мб ОЗУ, можно практически на равных сражаться с большинством гроссмейстеров мира. Путь на шахматный Олимп для компьютерных программ оказался нелегким. Обозначим основные вехи.

Историческая справка

1950 г. Американский математик Клод Шеннон, основатель теории информации, предложил общий подход к построению шахматных программ. Он утверждал: "Научное исследование проблемы игры в шахматы позволит развить методы построения систем искусственного интеллекта".

1952 г. В Манчестере (Англия) сыграна первая партия между компьютерной программой и человеком. Парадокс в том, что компьютера как такового еще не было. Его роль выполнял сам изобретатель алгоритма — выдающийся математик Алан Тьюринг. "Бумажной машине" Тьюринга требовалось более получаса, чтобы рассчитать очередной ход, и все же она проиграла.

"Бумажная машина" Тьюринга — А.Гленни, Манчестер, 1952 год:

1.e4 e5 2.Кc3 Kf6 3.d4 Cb4 4.Kf3 d6 5.Cd2 Кc6 6.d5 Kd4 7.h4 Cg4 8.a4 К:f3+ 9.g:f3 Ch5 10.Cb5+ c6 11.d:c6 0-0 12.c:b7 Лb8 13.Са6 Фа5 14.Фе2 Kd7 15.Лg1 Кc5 16.Лg5 Cg6 17.Cb5 К:b7 18.0-0-0 Кc5 19.Сc6 Лfc8 20.Cd5 C:c3 21.C:c3 Ф:a4 22.Крd2 Ке6 23.Лg4 Kd4 24.Фd3 Кb5 25.Cb3 Фа6 26.Сc4 Ch5 27.Лg3 Фа4 28.C:b5 Ф:b5 29.Ф:d6 Лd8.

Белые сдались — 0:1.

(***Примечание***: двоеточие обозначает процесс взятия фигуры или пешки, например g:f3 означает взятие пешкой, находящейся на вертикали "g" какой-то фигуры или пешки на поле "f3".)

1956 г. В Лос-Аламосе (США) в ядерном исследовательском центре была создана шахматная программа для компьютера, имевшего характерное название MANIAC-I. Первая партия "человек — машина" проводилась на уменьшенной доске 6x6 полей, длилась 10 часов и закончилась победой "белкового" шахматиста. Год спустя А.Бернштейн (Bernstein) разработал первую шахматную программу на полной доске для компьютера IBM-704.

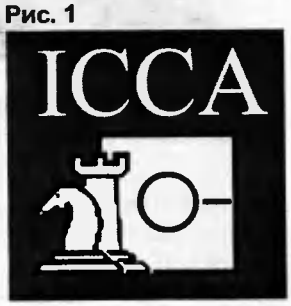
1963 г. Начало работ по созданию первой советской шахматной программы в Институте теоретической физики ИТЭФ под руководством доктора физико-математических наук А.Кронрода.

1967 г. Американская программа "MacHack VI" (PDP-6), разработанная Р.Гринблаттом, впервые сыграла в "человеческом" турнире. В том же году состоялся первый международный матч шахматных программ СССР — США, закончившийся нашей победой со счетом 3:1.

1972 г. Впервые проведен матч между компьютерной программой и читателями газеты "Комсомольская правда". С легкой руки шахматного обозревателя А.Хенкина, программа получила название "Каисса". Читатели, которые участвовали в том матче, помнят, что он длился почти год. "Каисса" одну партию проиграла, а другую свела вничью.

1974 г. В Стокгольме (Швеция) прошел первый компьютерный чемпионат мира, лидером которого стала отечественная шахматная программа "Каисса" [1]. Официально авторами "Каиссы" являются Г.Адельсон-Вельский, В.Арлазаров и М.Донской. Непосредственно над программой работали А.Битман, А.Бараев, А.Усков, А.Леман, М.Розенфельд.

Рис. 1



1977 г. Создание международной организации ICCA (International Computer Chess Association), которая стала координатором работ в области компьютерных шахмат (рис.1).

1978 г. Появление первого специализированного шахматного компьютера "Belle" американского ученого Кена Томпсона, кстати, одного из разработчиков операционной системы Unix. Революционное новшество заключалось в том, что функции передвижения фигур, выписки ходов и оценки позиции были реализованы не программно, а аппаратно, на уровне интегральных схем.

1986 г. Профессор Ганс Берлингер из университета Карнеги-Меллон создал "параллельный" шахматный компьютер, состоящий из 64 одновременно работающих специализированных процессоров.

1988 г. Ученики Г.Берлингера — Фенг-Хсиунг Хсу и Мюррей Кемпбелл — разработали программу "Deep Thought"

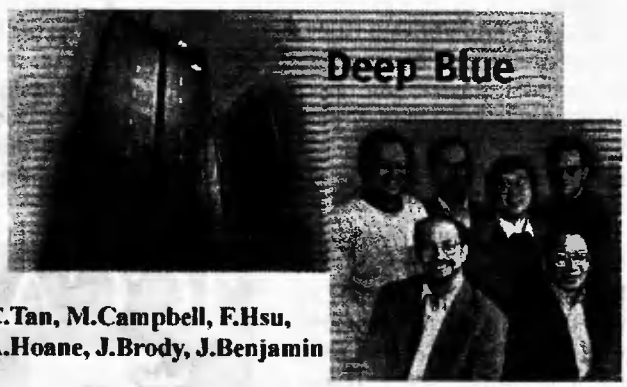
("Глубокая мысль"), которая первой достигла гроссмейстерского рейтинга. В том же году американский гроссмейстер А.Денкер попал в книгу рекордов Гиннеса как первый шахматист столь высокого ранга, который проиграл матч программе "Hitech".

1989 г. Первая встреча двух чемпионов мира — компьютера "Deep Thought" и Гарри Каспарова. Человеческий интеллект оказался сильнее шестипроцессорного "монстра", анализировавшего 750 тысяч позиций в секунду.

1992 г. В корпорации IBM начинаются работы по созданию преемника "Deep Thought", получившего название "Deep Blue". Четыре года спустя, несмотря на стократное увеличение быстродействия, он все же проиграл свой первый матч против Каспарова со счетом 2:4.

1997 г. В Нью-Йорке состоялся матч-реванш между улучшенным "Deep Blue" (на рис.2 —фотография коллектива разработчиков) и действующим чемпионом мира Г.Каспаровым. Человеку противостоял мощный сервер IBM RS/6000 SP, состоящий из 256 параллельно работающих шахматных чипов на основе процессоров PowerPC. Вес оборудования составлял 1,4 тонны. В базе данных компьютера находились 600 тысяч партий, сыгранных всеми гроссмейстерами мира за последние 100 лет. Анализ осуществлялся со скоростью 200 миллионов позиций в секунду при максимальной производительности 60 миллиардов ходов (moves) за 3 минуты. Чемпиону мира пришлось признать поражение со счетом 2,5:3,5. После матча он сказал: "Этот ящик, возможно, находится к интеллекту гораздо ближе, чем нам бы хотелось считать".

Рис. 2



C.Tan, M.Campbell, F.Hsu, A.Hoane, J.Brody, J.Benjamin

Другой точки зрения придерживаются ученые-математики, результат матча для которых не стал неожиданностью. Многие из них укрепились во мнении, что шахматы, как игра детерминированная (теоретически в любой позиции можно предсказать конечный результат), больше не является достойным объектом исследования проблем искусственного интеллекта. К примеру, тот же "Deep Blue" выиграл матч не за счет оригинального алгоритма расчета, а при помощи грубого "накачивания мышц" (увеличения быстродействия процессора).

Компьютер оказался в очередной раз сильнее человека, сумев сымитировать шахматную интуицию, позиционное мышление и комбинационное предвидение точным и далеким расчетом. Водораздел проблем искусственного интеллекта переместился из области шахмат в сферу оценочных экспертных систем. Тем не менее, разработка шахматных программ способствовала появлению новых алгоритмов в теории перебора вариантов и принятия оптимальных решений. Более того, сейчас на смену соперничеству "человек — машина" пришло соревнование "человек — человек". Имеются в виду разработчики компьютерных шахматных программ, число которых, как ни парадоксально, с каждым годом все увеличивается.

Росту популярности этого вида "спорта" способствуют регулярно проводимые турниры и чемпионаты с немалыми призовыми фондами. Играть можно не выходя из дома, через Интернет, причем на разных типах компьютеров. Так что отечественным "Кулибиным" и "Софьям Ковалевским" есть куда приложить силы.

Компьютерные шахматные программы

Компьютерные шахматные программы (КШП) делятся на специализированные и универсальные. Для работы первых из них требуются специальные шахматные компьютеры, в которых на аппаратном уровне реализованы многие шахматные команды. Это значительно ускоряет работу, но привязывает программу к конкретной машине. Чемпионы мира в классе больших ЭВМ (табл.2) в основном работали на специализированных компьютерах.

Табл. 2

Чемпионаты мира среди больших ЭВМ (WCCC)	Год	Программа-победитель, страна	Число участников
1. Стокгольм, Швеция	1974	"Каисса", СССР	13
2. Торонто, Канада	1977	"Chess 4.6", США	16
3. Линц, Австрия	1980	"Belle", США	18
4. Нью-Йорк, США	1983	"Cray Blitz", США	22
5. Кельн, ФРГ	1986	"Cray Blitz", США	22
6. Эдмонтон, Канада	1989	"Deep Thought", США	24
7. Мадрид, Испания	1992	"Chassmachine", Нидерланды	22
8. Гонконг (Сянган)	1995	"Fritz", Германия/ Нидерланды	24
9. Падерборн, Германия	1999	"Shredder", Германия	30

Шахматные гиганты, стоимостью многие миллионы, служат прекрасной рекламой фирмам-изготовителям электронной продукции (IBM, Fujitsu, Siemens), но резко сужают круг лиц, которые бы могли разрабатывать новые алгоритмы и генерировать идеи. Как альтернатива, с 1980 г. стали параллельно проводиться отдельные чемпионаты мира среди микрокомпьютеров WMCCC (табл.3), в которых доминирующая роль принадлежит универсальным КШП. Эти программы пишутся, как правило, на языке Си и его диалектах, что позволяет легко перенастраивать их под любые операционные системы (Unix, Linux, Windows) и под разные типы персональных компьютеров. Так появились упрощенные версии специализированных программ, которые ранее были разработаны только для больших ЭВМ.

В свою очередь, универсальные КШП делятся на коммерческие и любительские. Первые из них обычно спонсируются известными шахматными фирмами-дистрибьюторами — ChessBase, Gambit-Soft, Schroder-BV, Millenium. В разработке коммерческих проектов принима-

Табл. 3

Чемпионаты мира среди микрокомпьютеров (WMCCC)	Год	Программа-победитель, страна	Число участников
1. Лондон, Англия	1980	"Chess Challenger", США	-
2. Травемюнде,ФРГ	1981	"Fidelity", США	-
3. Будапешт, Венгрия	1983	"Elite A/S", США	-
4. Глазго, Шотландия	1984	"Elite X", США	4
5. Амстердам, Нидерланды	1985	"Mephisto", Великобритания	-
6. Даллас, США	1986	"Mephisto", Великобритания	14
7. Рим, Италия	1987	"Mephisto", Великобритания	-
8. Альмерия, Испания	1988	"Mephisto", Великобритания	-
9. Портороз, Югославия	1989	"Mephisto", Великобритания	-
10. Лион, Франция	1990	"Mephisto", Великобритания	12
11. Ванкувер, Канада	1991	"Gideon", Нидерланды	15
12. Мюнхен, Германия	1993	"Hiarcs", Великобритания	28
13. Падерборн, Германия	1995	"Mchess-Pro 5.0", США	34
14. Джакарта, Индонезия	1996	"Shredder", Германия	27
15. Париж, Франция	1997	"Junior", Израиль	34
16. Падерборн, Германия	1999	"Shredder", Германия	30
17. Лондон, Англия	2000	"Shredder", Германия	14

ют участие целые коллективы программистов, математиков и дизайнеров под руководством автора идеи. Выпускаются такие программы на лицензионных CD-ROM стоимостью в пределах $30...100.

Любительские КШП, в отличие от коммерческих, распространяются бесплатно или условно бесплатно, в основном через Интернет. В большинстве случаев их пишут одаренные программисты-одиночки или группы из двух-трех человек. Любительских программ по количеству в несколько раз больше коммерческих, а по силе игры они вплотную "наступают на пятки" своим именитым собратьям.

Табл.4 составлена на основе рейтинг-листа Шведской Ассоциации компьютерных шахмат SSDF [2] (77268 игр, К6-2-450 МГц, 128 Мб ОЗУ, регламент партий — 2 часа на 40 ходов), а табл.5 (любительские программы, май 2001 г., регламент — 30 минут на партию) — по частным оценкам Интернет-сайтов. Интересно, что любительские программы "SOS" и "Crafty" занимают соответственно 12-е и 15-е места в коммерческом списке SSDF, считающемся наиболее объективным и беспристрастным в мире.

Как сопоставить силу игры машин и людей? Напомним, что ФИДЕ присваивает три звания: мастер ФИДЕ — рейтинг 2300, международный мастер — рейтинг 2400 и международный гроссмейстер — рейтинг 2600. Первые 100 сильнейших шахматистов мира имеют рейтинг выше 2550, рейтинг более 2800 имеет только Г.Каспаров. Интересное предложение выдвинул гроссмейстер А.Халифман — для повышения стимула к совершенствованию ввести еще одну ступеньку — "супергроссмейстер".

В национальных шахматных федерациях существуют свои критерии оценки, как правило, несколько отличающиеся от приведенных выше. Например, в США — "старший мастер" (Senior Master) и просто "мастер" (Master). Причем первое звание примерно соответствует нашему званию "мастер спорта", а второе — "кандидат в мастера спорта".

Табл. 5

Название	Авторы	R, ед.	Интернет-адрес
1. Crafty	Dr. Robert Hyatt, США	2528	http://www.tim-mann.org/crafty.html
2. Yace	Dieter Buerssner, Германия	2526	http://amateurschach.in-trier.de/schach/engines/english/yace.htm
3. Amy	Thorsten Greiner, Германия	2524	http://www.geocities.com/hiiamamy/
4. AnMon	Christian Barreteau, Франция	2505	http://amateurschach.in-trier.de/schach/engines/anmon.htm
5. LG2000	Michael Borgstadt, Германия	2493	http://www.borgstaedt.de/
6. SOS	Rudolf Huber, Германия	2443	http://amateurschach.in-trier.de/schach/engines/sos.htm
7. Bionic Impakt	Hans Secelle & Albrecht Heeffer, Бельгия	2436	http://www.Impakt.be/bionic/
8. The Crazy Bishop	Remi Coulom, Франция	2434	http://www-leibniz.imag.fr/~coulom/
9. Comet	Dr. Ulrich Tuerke, Германия	2428	http://amateurschach.in-trier.de/schach/engines/comet.htm
10. ZChess	Franck Zibi, Франция	2419	http://myweb.worldnet.net/~fzibi/zchess.htm

Табл. 4

Название	Авторы	R, ед.
1. Deep Fritz	Frans Morsch & Mathias Feist	2650
2. Fritz 6.0	Frans Morsch & Mathias Feist	2626
3. Junior 6.0	Amir Ban & Shay Bushinsky	2594
4. Chess Tiger 12.0	Christophe Theron	2578
5. Fritz 5.32	Frans Morsch & Mathias Feist	2547
6. Nimzo 7.32	Chrilly Donninger	2547
7. Nimzo 8.0	Chrilly Donninger	2539
8. Gandalf 4.32f	Steen Suurballe & Dan Wulff	2529
9. Junior 5.0	Amir Ban & Shay Bushinsky	2528
10. Hiarcs 7.01	Mark Uniacke	2526

Среди шахматных программ также имеется разделение по силе игры, но официально утвержденных званий не существует. Их заменяют сравнительные оценки в периодически проводимых матчах против ведущих шахматистов мира и в смешанных человеко-машинных турнирах. По расчетам Эрика Холсворта (Eric Hallsworth) — редактора одного из английских шахматных журналов — разница между человеко-машинными рейтингами для программ гроссмейстерского уровня, подобных "Fritz-6", "Junior-6", "Hiarcs 7.32", составляет 30...40 единиц.

(Окончание следует)

Г.ТРОЯН,
г.Минск, РИВШ БГУ,
E-mail: G-Trojan@yandex.ru

Эффективный поиск информации в Internet

(Продолжение. Начало в №9/2001)

Каталоги WWW

Рассмотрим особенности систем-каталогов. В каталогах описание источников информации проводится персоналом, то есть людьми, которые составляют краткую аннотацию на каждый ресурс. Затем, как правило, проводится сортировка описанных ресурсов по темам (составление тематического каталога).

Поиск в каталоге очень удобен и проводится посредством последовательного уточнения тем. На начальной (домашней) странице системы подобного рода вы увидите список самых крупных тем (категорий), выделенных персоналом каталога, реализованных в виде гипертекстовых ссылок. Например — Компьютеры, Интернет, Образование, Искусство и т.д. Выбрав ссылку на категорию первого уровня, вы попадете на страницу со списком подкатегорий, и т.д. Таким образом, не углубляясь в сложности составления запросов, вы достаточно легко найдете источники по выбранной вами тематике. Следует заметить, что ресурсы, описанные в каталогах, обычно представляют собой специализированные сайты высокого качества.

Многие каталоги поддерживают возможность быстрого поиска определенной категории или страницы по ключевым словам с помощью локальной поисковой машины.

Отметим, что база данных ссылок (индекс) каталога обычно имеет ограниченный объем. Некоторые каталоги используют программы-роботы для автоматического обновления индекса.

Результат поиска в каталоге представляется в виде списка, состоящего из краткого описания (аннотации) документов с гипертекстовой ссылкой на первоисточник.

Адреса известных каталогов

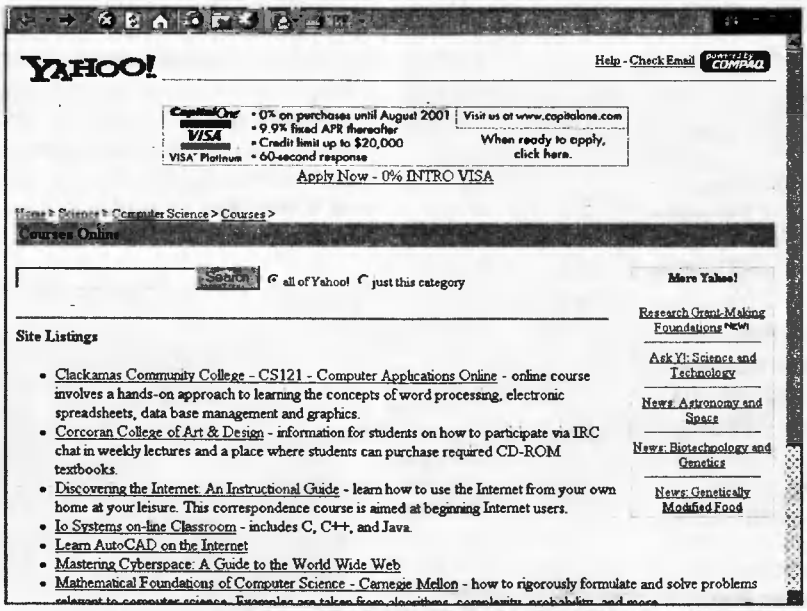
Среди самых популярных зарубежных каталогов следует в первую очередь упомянуть каталог Yahoo. В число каталогов с размером индекса свыше 2 миллионов ссылок входят каталоги Open Directory и LookSmart:

- **Yahoo!** (`http://www.yahoo.com`);
- **Open Directory** (`http://dmoz.org`);
- **LookSmart** (`http://www.looksmart.com`).

Российские популярные каталоги:

- **@Rus** (`http://www.atrus.ru`);
- **List.ru** (`http://www.list.ru`);
- **Weblist** (`http://www.weblist.ru`);
- **Созвездие интернет** (`http://www.stars.ru`).

Рис. 3



Внешний вид каталога Yahoo!

Рассмотрим внешний вид домашней страницы самого популярного англоязычного каталога Yahoo! (http://www.yahoo.com) (рис.3). Как обычно, на начальной странице систем-каталогов вы обязательно увидите список крупных тем (категорий). В каталоге Yahoo! есть возможность проведения быстрого поиска по ключевым словам с использованием бланка для ввода запроса. Наиболее простым способом поиска являются последовательные щелчки мышью по нужной вам категории. Например, наша задача — найти Web-сайты, посвященные on-line-курсам по компьютерным наукам, т.е. курсам, проводящим обучение через Internet. В этом случае можно выбрать следующую схему уточнения по темам: с домашней страницы выбираем ссылку **Science** (Наука), далее **Computer Science** (Компьютерные науки), затем

Рис. 4

Courses (Курсы) и **Courses Online** (Курсы онлайн). В результате переходов получаем список аннотаций с названиями соответствующих страниц. Далее, с помощью щелчка по заинтересовавшей ссылке можно перейти к первоисточнику и изучить его (рис.4).

Поисковые машины

Отличительной чертой поисковых машин является тот факт, что база данных с информацией о Web-страницах формируется и поддерживается в актуальном состоянии программой-роботом и, как следствие, имеет гораздо больший объем по сравнению с системами каталогового типа. Например, поисковая машина Altavista содержит в индексе порядка 550 миллионов ссылок (по состоянию на 06.04.2001) [4].

Поиск в такой системе обычно проводится по запросу, формулируемому пользователем и состоящему в простейшем случае из набора ключевых слов. В последнее время существует тенденция сортировки содержимого индекса по категориям, что позволяет сузить область поиска и использовать возможность поиска с уточнением темы.

Простой поиск. Обобщенные возможности формирования запроса

Как правило, поисковые машины поддерживают два режима — режим простого поиска и режим расширенного поиска. Рассмотрим обобщенные возможности формирования запроса в режиме простого поиска. Можно просто вводить через пробел одно или несколько слов, поиск слов со всевозможными окончаниями моделируется символом "*" в конце слова. Многие системы позволяют искать словосочетания или фразу — для этого искомый фрагмент необходимо заключить в кавычки. Возможно обязательное включение или исключение определенных слов, реализуемое знаками "+" и "-" соответственно, набираемыми вплотную к ключевому слову.

Основная проблема поиска по примитивно составленному запросу (в виде перечисления ключевых слов) заключается в том, что поисковая машина найдет все страницы, на которых указанные слова встречаются в любой части документа. В результате количество найденных страниц будет слишком велико. Для улучшения качества поиска в режиме простого поиска допустимо использование логических операторов и операторов, позволяющих ограничить область поиска, а также выбор определенной категории документов из представленного списка.

Операторы, устанавливающие отношения между ключевыми словами

Большинство поисковых систем используют следующие операторы, устанавливающие отношения между ключевыми словами:

- **AND (И) &** — обязательное присутствие всех ключевых слов;
- **OR (ИЛИ) |** — присутствие хотя бы одного из ключевых слов;
- **NOT (НЕ) !** — отсутствие ключевого слова;
- **NEAR (ОКОЛО) ~** — определенный интервал между ключевыми словами.

В качестве примера приведем запрос, который можно сформулировать в поисковой машине Altavista для нахождения документов, в которых присутствует слово **интернет** со всевозможными окончаниями и словосочетание **поиск работы**, причем расстояние между ними не должно превышать 10 слов:

интернет* NEAR "поиск работы"

Специальные операторы

Многие поисковые системы включают в свой язык составления запросов специальные операторы, позволяющие проводить поиск в определенных зонах документа (например, в его заголовке) или искать документ по известной части его адреса. Полезной возможностью является поиск в сети документов, ссылающихся на страницу с указанным вами адресом (URL). Таким способом можно найти в сети страницы, на которых есть ссылки на ваш Web-сайт. Некоторые системы позволяют ограничить область поиска внутри указанного домена.

В качестве дополнительных специальных операторов можно выделить:

- операторы поиска документов с определенным графическим файлом;
- операторы ограничения по дате;
- операторы уточнения по количеству слов между указанными ключевыми словами;
- операторы учета словоформы;
- операторы сортировки результатов (по релевантности, свежести, старости).

Примечание. К сожалению, на сегодняшний день не существует единого стандарта для различных поисковых систем на количество и синтаксис поддерживаемых операторов. Попытки разработать подобный стандарт предпринимаются, поэтому есть надежда на то, что разработчики поисковых систем позаботятся об удобстве пользователей. На данном этапе развития средств поиска пользователь, обращаясь к определенной поисковой системе, непременно должен в первую очередь ознакомиться с ее правилами по составлению запросов. Обычно на домашней странице присутствует ссылка **Помощь (Help)**, по которой вы сможете перейти к справочной информации.

Сравните, какой вид имеет оператор поиска в заголовке в поисковых системах Altavista, Яndex, Апорт:

- Altavista (`http://www.altavista.com`) — **title:(выражение)**;
- Яndex (`http://www.yandex/ru`) — **$title (выражение)**;
- Апорт (`http://www.aport.ru`) — **title=(выражение)**.

Расширенный (детальный, advanced) поиск

Чтобы написать запрос с уточнением параметров, необходимо знать язык составления запросов для конкретной поисковой машины. Это не очень просто для пользователя, поэтому многие автоматические индексы предлагают воспользоваться возможностями так называемого режима расширенного поиска. Как правило, на начальной странице поисковой системы есть ссылка **Расширенный поиск (Advanced Search)**, реализующая переход к соответствующему режиму составления запросов.

Режим расширенного или детального запроса в разных системах реализован индивидуально, но чаще всего — это бланк, в котором упомянутые выше операторы реализуются установкой соответствующих флажков или выбором параметров из списка. Таким образом, у вас появляется возможность составить качественный запрос, не прибегая к сложному языку и многочисленным операторам.

Представление результатов поиска

Рассмотрим способы представления результатов поиска в поисковых машинах. Обычно количество найденных документов превышает несколько десятков, а в отдельных случаях может достигать сотен тысяч! Поэтому в качестве формы выдачи составляется список документов по 5, 10 или 15 единиц на странице с возможностью перехода к следующей порции внизу страницы. Обязательно указывается заголовок и URL (адрес) найденного документа, иногда система указывает в процентах степень релевантности документа.

В описании документа чаще всего содержится несколько первых предложений или выдержки из текста документа с выделением ключевых слов. Как правило, указана дата обновления (проверки) документа, его размер в килобайтах, некоторые системы определяют язык документа и его кодировку (для русскоязычных документов).

Рис. 5

страницe поисковой системы (рис.5) находится бланк ввода запроса для простого режима поиска. Обратите внимание на ссылку **Расширенный поиск**, реализующую переход к режиму расширенного поиска, и на ссылку **Помощь** для вызова справки о правилах составления запросов.

Одним из компонентов портала является система добровольного рейтинга сайтов/страниц по посещаемости — **Top100**. Эта система дает возможность владельцам серверов определить свою популярность на основе сравнительной оценки с другими серверами. На страницах, владельцы которых желают участвовать в рейтинговой системе, размещается специальный счетчик, фиксирующий сведения о посетителях данной страницы. На основании этих данных

Обработка результатов поиска

Что можно делать с полученными результатами? Если название и описание документа соответствуют вашим требованиям, можно немедленно перейти к его первоисточнику по ссылке. Это удобнее делать в новом окне, чтобы далее иметь возможность анализировать результаты выдачи. Многие поисковые системы позволяют проводить поиск в найденных документах, причем вы можете уточнить ваш запрос введением дополнительных терминов. Если интеллектуальность системы высока, вам могут предложить услугу поиска похожих документов. Для этого вы выбираете особенно понравившийся документ и указываете его системе в качестве образца. Однако автоматизация определения "похожести" — весьма нетривиальная задача, и зачастую эта функция работает неадекватно вашим надеждам [5]. Некоторые поисковики позволяют провести пересортировку результатов. Стандартно результаты поиска сортируются по релевантности, однако затем вы можете выбрать другой способ сортировки (например, по свежести, чтобы вверху списка были показаны самые новые документы, найденные по вашему запросу). Можно сохранить результаты поиска в виде файла на локальном диске для последующего изучения в автономном режиме.

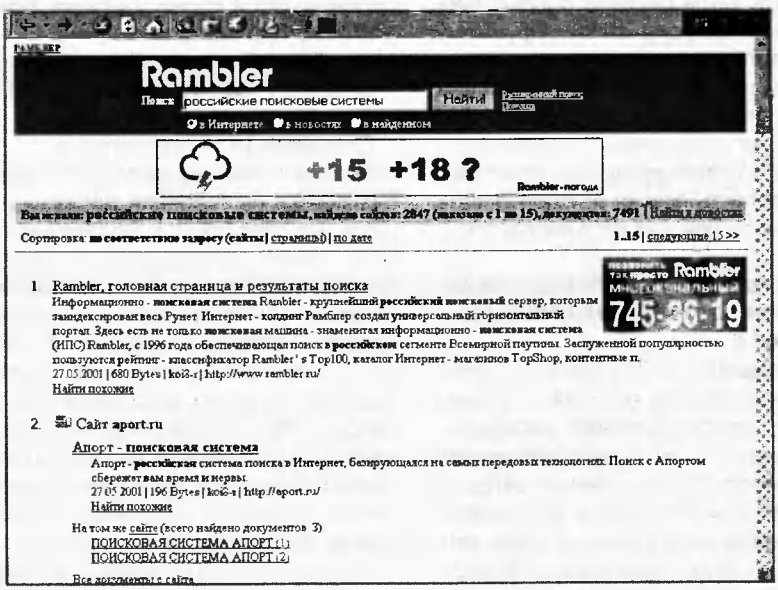
Адреса популярных поисковых машин

Приведем адреса некоторых наиболее популярных поисковых машин за рубежом и в России.

Зарубежные поисковые машины:

- **Altavista** (`http://www.altavista.com`);
- **Go (Infoseek)** (`http://www.go.com`);
- **Google** (`http://www.google.com`);
- **Excite** (`http://www.excite.com`);
- **HotBot** (`http://www.hotbot.com`);

Рис. 6



- **Nothern Light** (`http://www.northernlight.com`).

Российские поисковые машины:

- **Яndex** (`http://www.yandex.ru`);
- **Рамблер** (`http://www.rambler.ru`);
- **Апорт** (`http://www.aport.ru`).

Популярные белорусские поисковые системы:

- система **ALL.BY** (`http://all.by`);
- система ***.BY** (`http://search.promedia.minsk.by`);
- регистр белорусских WWW-ресурсов **Зубр** (`http://www.zubr.com`);
- белорусский интернет-каталог **Акавіта** (`http://akavita.kryvia.net/`);
- каталог **Белорусские ресурсы** (`http://www.belresource.com.by`).

Пример поиска в поисковой машине Рамблер

Рассмотрим поиск по запросу ***российские поисковые системы*** в поисковой машине портала Рамблер (`http://www.rambler.ru`). На домашней странице поисковой системы находятся базовые рейтинги сайтов по категориям, и посетитель Рамблера может с ними ознакомиться [5].

Составим простейший тестовый запрос, состоящий из трех слов — ***российские поисковые системы***. В результате Рамблер нашел 7491 документ на 2647 сайтах. Для каждой найденной страницы мы видим заголовок, начало текста, размещенного на странице, дату последнего обновления, размер файла, кодировку и URL документа (рис.6).

Для уточнения результатов поиска есть возможность установить переключатель **в найденном** в бланке запроса, ввести в поле ввода уточняющие термины и провести поиск среди найденных документов. Кроме того, можно обратиться к поисковой системе с просьбой найти похожие страницы, выбрав ссылку **Найти похожие**, размещенную под описанием избранного вами документа.

(Продолжение следует)

ЛИСТАЯ СТРАНИЦЫ

О голосовом взаимодействии человека и компьютера рассказывает *Егор Поваляев* **(Компьютер Пресс, №6/2001, С.172...177)**

Клавиатура и мышь пока остаются наиболее распространенными устройствами общения с компьютером. Однако с точки зрения пользователя они не идеальны. Людям гораздо привычнее передавать информацию с помощью речи, жестов, мимики. Когда-то считавшаяся непосильной для ПК задача распознавания речи сегодня стала решаемой.

Системы речевого общения делятся на два класса: распознавание речи и синтез речи. Продукты, основанные на технологии распознавания, можно разделить на три большие группы:

- средства речевого управления;
- средства диктовки;
- средства идентификации по образцу речи.

Основная идея систем речевого управления — человек может обратиться к компьютеру с некоторыми словами или командами, а тот его понимает и выполняет необходимые действия. Системы распознавания дискретной речи на сегодня работают достаточно хорошо, но они воспринимают только отдельные слова или фразы. Системы для слитной речи в настоящее время внедряются в основном в области телекоммуникации.

Первыми системами средств диктовки были системы раздельной диктовки — они проще в разработке и менее требовательны к вычислительной мощности ПК, однако вынуждают пользователя произносить слова с короткими промежутками. Следующим шагом развития систем диктовки стали системы распознавания слитной речи, которые требуют более быстрых процессоров и больших объемов памяти. Пионером в этой области стала компания Dragon Systems со своей системой DragonDictate. Эта система позволяет надиктовывать текст в программы Word, Word Perfect, Netscape Navigator, Internet Explorer и другие популярные приложения.

Словарь DragonDictate насчитывает более 30 тыс. слов, к тому же, пользователь может дополнить словарь необходимыми ему терминами.

В настоящее время наиболее популярными программами распознавания речи являются такие программы, как IBM ViaVoice Gold, Philips FreeSpeech 2000, L&H Voice Xpress Pro, Dragon NaturellySpeaking Prefferred и "Горыныч Проф 1.0"

Все пять пакетов рассчитаны в первую очередь на то, чтобы обеспечить ввод в документ текста. Первые четыре позволяют вводить английскую речь, и только последний — русскую.

С задачей ввода текста лучше всех справляется ViaVoice. Он достиг заявленной безошибочности распознавания в 95%, при этом качество распознавания команд осталось высоким. У Dragon NaturellySpeaking Prefferred, Philips FreeSpeech 2000, L&H Voice Xpress Pro точность распознавания — около 90%. Точность распознавания "Горыныч Проф 1.0" — около 60...70%.

Диктовать сплошной текст во всех системах достаточно просто. Однако вносить исправления в текст с помощью голосовых команд гораздо сложнее. Более удобные команды редактирования — у IBM ViaVoice. Кроме того, неплохими возможностями форматирования может похвастаться и Dragon. Однако пользоваться для данных целей клавиатурой и мышью все же удобнее, да и быстрее.

Все пакеты обеспечивают диктовку непосредственно в окне любой программы, работающей с текстами, в частности, Microsoft Word, Excel, и в таких популярных почтовых клиентах как Out Look Express Netscape Messenger. Правда, на компьютерах относительно старых моделей обработка произносимых человеком слов может происходить с задержкой.

Есть возможность не только диктовать программам, но и давать им указания, например, открыть такой-то файл, напечатать такую-то страницу, а также управлять перемещениями по рабочему столу. Соответствующие функции заложены во всех пяти программах, но, к сожалению, не всегда работают. Некоторые команды, например, "click File" (щелкнуть по пункту File) или "click Save" (щелкнуть по пункту Save) в Word, приходится повторять по нескольку раз, прежде чем компьютер распознает команду.

Секретами установки Windows 98 и Windows NT/2000 на один компьютер делится *С. Мовчан* **(Hard'n'Soft, N6, С. 78...84).**

Многие пользователи предпочитают устанавливать на ПК две и более ОС. В самом деле, работать с критически важной информацией можно, скажем, в среде повышенной надежности Windows NT/2000, а играть — в Windows 98/Me. При этом важно обеспечить корректную установку, настройку и функционирование этих операционных систем. Учитывая что Windows 98 не работает с файловой системой NTFS, а многие приложения используют общие библиотеки, эта задача является нетривиальной.

Перед тем как создавать систему с двойной нагрузкой, объединяющую Windows 2000 и Windows 98, оценим технические возможности вашего компьютера и разберемся, можно ли вообще использовать на нем Windows 2000. В таблице указаны требования к аппаратным ресурсам различных версий Windows 2000, которые вы можете сравнить с параметрами вашего ПК.

Не следует экспериментировать с Windows 2000 на маломощных компьютерах с процессором AMD K5 или Intel Pentium с тактовой частотой менее 200 МГц, нервы дороже. Критичным показателем является и объем оперативной памяти: если в компьютере установлено меньше 64 Мб, то

Минимальные требования ОС к процессору, памяти и жесткому диску

Версия операционной системы	Минимальные требования к процессору		Минимальные требования к оперативной памяти, Мб		Минимальные требования к размеру жесткого диска, Гб		Минимальные требования к свободному дисковому пространству, Гб	
	Рекомендации Microsoft	Реальная оценка	Рекомендации Microsoft	Реальная оценка	Рекомендации Microsoft	Реальная оценка	Рекомендации Microsoft	Реальная оценка
Windows 2000 Professional	Pentium 133 МГц	Pentium 200 МГц	64	128	2	3	1	1,5
Windows 2000 Server	Pentium 133 МГц	Pentium 200 МГц	256	384	2	3	1	1,5
Windows 2000 Advanced Server	Pentium 133 МГц	Pentium 200 МГц	256	384	2	3	1	1,5

ни о какой многозадачности или Интернете речи быть не может. Чтобы получать удовольствие от работы с Windows 2000, придется увеличить объем оперативной памяти как минимум до 128, а лучше до 256 Мб.

Для работы с Windows 2000 приготовьтесь принести в жертву Microsoft более 600 Мб дискового пространства под содержимое системных папок ОС. И еще столько же — для файла-подкачки. Естественно, вам потребуется место и для других программ.

Как правило, жесткий диск разбит на несколько логических разделов (дисков). В какой раздел устанавливать каждую из систем — решать вам. При этом нужно уяснить главное — с файловой системой NTFS 5.0 работает только Windows — Windows 98 "понимает" FAT16 и FAT32. Использование дополнительных программ, позволяет "научить" Windows 9.x "видеть" раздел NTFS 5.0, но не более того — все преимущества NTFS проявляются лишь в Windows 2000. Поэтому в одном из разделов жесткого диска обязательно нужно сохранить FAT16 или FAT32 — чаще всего для этих целей выделяют диск C:, который должен быть "виден" обеим операционным системам.

Существует общее правило: устанавливать каждую ОС на отдельный диск, что позволяет избежать множества неудобств и неожиданностей. Но бывают случаи, когда так поступить не позволяет малый объем жесткого диска. Что же будет происходить в данной ситуации? Операционные системы инсталлируются в разные папки — C:/WINDOWS или C:/WINNT и работают независимо друг от друга. Однако прикладные программы могут использовать одни и те же папки (например, Program Files), в которых они размещают свои компоненты и библиотеки. При инсталляции одной ОС поверх другой файлы в общих папках перезаписываются, и программа в некоторых случаях уже не сможет найти необходимые файлы и перестанет работать. Как с этим бороться? К сожалению, универсальных рекомендаций дать нельзя.

Возможны два варианта установки Windows 2000 поверх Windows 98/Me и наоборот.

Кроме загрузки одной из нескольких ОС, установленных в разные разделы диска, можно воспользоваться и другим приемом — заставить их работать одновременно. Для этого существует программа VMWare (www. vmware. com), превращающая ПК в несколько виртуальных. С ее помощью на одном компьютере можно работать с несколькими ОС, переключаясь с одной на другую. При этом нет необходимости пользоваться мультизагрузкой. Недостатком виртуального компьютера является то, что он работает на 30% медленнее и требует очень много ресурсов.

Существует несколько простых правил, которые помогут избежать большинства проблем, с которыми сталкиваются люди, экспериментирующие с двойной загрузкой:

- Перед тем как приступить к экспериментам, проверьте, есть ли у вас в запасе загрузочная дискета или загрузочный диск.
- Если что-то не работает, или установка пошла не так как планировалось, — остановитесь, не впадайте в панику.
- Форматирование жесткого диска или удаление раздела на нем — самый последний вариант решения проблем, и, как правило, к нему прибегают, когда не хотят вникать в суть проблемы, разбираться в ней.
- Для начала попробуйте понять, что неправильно, проверьте ваши предположения и только затем действуйте.

Проблема кражи паролей на доступ в Сеть у пользователей Internet и защита от этого — предмет статьи *А.Орлова* **(Подводная лодка, N6, С. 110...113).**

Для доступа в Internet с помощью городской телефонной сети в общем случае необходимо знать номер телефона провайдера, логин и пароль, а также иметь определенное количество денег на счете в базе данных провайдера, записанных на определенный логин (логин, пароль и место в базе данных провайдера именуются в этом случае аккаунтом). Именно номер телефона провайдера, логин и пароль нужны тем, кто собирается за ваш счет путешествовать по Сети.

Каким образом злоумышленник получает секретную информацию? Обычно он прибегает к одному из следующих способов.

1. Взламывает сервер провайдера и получает доступ к его базе данных с аккаунтами пользователей.

2. Узнает логин с паролем лично у вас. Например, прислав письмо якобы от службы технической поддержки с просьбой сообщить логин и пароль под предлогом аварии базы данных или замены паролей.

3. Открывает сохраненные пароли на вашем компьютере. Если злоумышленник физически получит доступ к вашему компьютеру, то он легко сможет с помощью программы Open Pass в любом окне запроса пароля вместо звездочки увидеть ваш password.

4. Похищение с вашего компьютера файлов, в которых был сохранен пароль. Можно просто подойти и скопировать на дискету, можно воспользоваться локальной сетью или "дырой" в программном обеспечении.

5. Установка на ваш компьютер программы, которая узнает вводимый вами пароль и логин и отсылает их злоумышленнику. Обычно это делается с помощью программ-троянов, попадающих на компьютер жертвы под видом полезных программ.

6. Простой подбор пароля, если вы используете в качестве такового простые осмысленные слова (имена, клички животных и т. д.).

Какие же существуют приемы защиты? Примерный набор действий и мероприятий для этого может быть следующим.

1. Берегитесь вирусов и "троянов". Регулярно обновляйте антивирусные программы и избегайте "случайных связей" — в этом залог вашей безопасности.

2. Не используйте простые пароли.

3. Никогда не сохраняйте пароли на вход в Интернет на жестком диске.

4. Используйте разные пароли для разных ресурсов.

5. Если вы все-таки решили сохранить пароль на вход в Сеть на компьютере — используйте надежные программы для их хранения (шифрования).

6. Никогда не храните на вашем счету у провайдера много денег.

7. Регулярно меняйте свой пароль на вход в Интернет, а если возможно — то и логин.

8. Защитите ваш компьютер от атак через Интернет. Для этого используйте любой файерволл вроде AtGuard, Personal Fire wall и др.

9. Ну и, наконец, ни под каким предлогом не сообщайте никому ваши пароль и логин.

Журналы листал С.Дроздовский.

ООО "НТК ИНФОТЕХ"

М.ДОЛИНСКИЙ,
г.Гомель.

АЛГОРИТМЫ ГЕНЕРАЦИИ КОМБИНАТОРНЫХ ОБЪЕКТОВ

Введение

Существует набор задач, решение которых заключается в генерации всех элементов таких комбинаторных объектов как множество всех подмножеств, перестановки, сочетания, размещения, перестановки с повторениями, сочетания с повторениями.

Для каждого сгенерированного элемента затем проверяются какие-то свойства для конкретной задачи.

В дальнейшем в данной статье предлагается следующий порядок изложения материала для каждого комбинаторного объекта — пример, алгоритм, программа, комментарии к программе.

Все программы для большей наглядности в качестве иллюстрации оперируют с массивом символов от A до Z. Очевидно, что предлагаемые алгоритмы и программы практически не изменяются при работе с массивами элементов любого типа, требуемого по условиям задачи (например, массивами чисел, слов, геометрических фигур и т.д.).

1. Множество всех подмножеств

Пусть мы имеем множество из четырех компонентов — которые мы обозначаем латинскими буквами A, B, C, D соответственно.

И пусть по условиям задачи требуется выбрать подмножество, состоящее из нескольких компонентов и обладающее некоторым свойством. Предлагается такой способ решения задачи — мы генерируем ***все*** возможные подмножества данного множества, и для каждого из сгенерированных подмножеств проверяем, удовлетворяет ли оно заданному свойству. Альтернативный вариант задачи — подсчитать ***все*** подмножества данного множества, обладающие заданным свойством.

Например, для множества из четырех символов A, B, C, D множество всех подмножеств включает в себя следующие множества:

- пустое множество;
- одноэлементные множества — {A}, {B}, {C}, {D};
- двухэлементные множества — {A,B}, {A,C}, {A,D}, {B,C}, {B,D}, {C,D};
- трехэлементные множества — {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D};
- четырехэлементное множество {A,B,C,D}.

В случае, если порядок генерации подмножеств не играет роли (а, например, в случае необходимости подсчитать все подмножества, обладающие заданным свойством, так оно и есть) один из наиболее просто кодируемых алгоритмов генерации множества всех подмножеств выглядит следующим образом.

Определим вектор B, состоящий из четырех чисел, каждое из которых может принимать значение 0 или 1. И будем считать, что значение 1 указывает на то, что соответствующий по номеру компонент исходного множества включается в множество, а значение 0 указывает на то, что элемент не включается.

Рассмотрим теперь последовательность двоичных чисел от 0 до 15 и соответствующие им подмножества:

```
   4321
   DCBA
 - 0000 - пустое множество;
 - 0001 - A;
 - 0010 - B;
 - 0011 - AB;
 - 0100 - C;
 - 0101 - AC;
 - 0110 - BC;
 - 0111 - ABC;
 - 1000 - D;
 - 1001 - AD;
 - 1010 - BD;
 - 1011 - ABD;
 - 1100 - CD;
 - 1101 - ACD;
 - 1110 - BCD;
 - 1111 - ABCD;
```

Таким образом, всего имеется шестнадцать различных подмножеств множества из четырех элементов. В общем случае множество всех подмножеств множества из N элементов содержит 2^N элементов.

Алгоритм, обеспечивающий такую генерацию множества всех подмножеств из N элементов, может быть неформально описан следующим образом.

Формируем массив, состоящий из N нулей, и рассматриваем его как пустое подмножество. Таким образом, начальное значение текущего подмножества — пустое множество.

Для получения следующего подмножества из текущего подмножества обрабатываем текущий массив из чисел 0 или 1 следующим образом. Справа (от первого элемента массива к последнему) ищем первое число, равное нулю. Если такое число не найдено — значит, текущее подмножество является последним — множеством, состоящим из всех элементов, и на этом алгоритм заканчивает свою работу. Если же элемент, равный 0, найден, то он заменяется на 1, а все числа справа от него (если таковые имеются) заменяются на нули.

Более формализованно этот алгоритм может быть записан так:

```
Ввод (N)
Обнуление массива B из N+1 элемента
Вывод (Пустое множество)
Пока B[N+1]=0
  i=1
  Пока B[i]=1 делать B[i]=0, i=i+1
  B[i]=1
  Вывод (множества, определяемого массивом B).
```

Ниже приводится текст программы, которая считывает N — число элементов в множестве, и выводит на экран множество всех подмножеств, обозначая элементы соответствующими по порядку латинскими буквами.

```
const
  alphabet : string[26] = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
var
  b    : array [1..100] of byte;
```

```
  N,i : byte;
begin
  readln(N);
  for i:=1 to N+1 do b[i]:=0;
  writeln ("Пустое множество");
  while (b[N+1]=0) do
    begin
      i:=1;
      while B[i]=1 do
        begin B[i]:=0; inc(i); end;
      B[i]:=1;
      for i:=1 to n do
        if b[i]=1 then write(alphabet[i]);
      writeln;
    end;
  readln;
end.
```

При необходимости обрабатывать (анализировать) построенные подмножества, могут быть также добавлены вызовы процедур обработки, получающие в качестве параметра массив B (указывающий своими единичными элементами номера элементов множества, включенных в текущее подмножество).

2. Перестановки

Пусть мы имеем четыре компонента, обозначенные буквами A, B, C, D соответственно.

Тогда множество всех перестановок из этих компонентов будет включать следующие элементы:

ABCD	BACD	CABD	DABC
ABDC	BADC	CADB	DACB
ACBD	BCAD	CBAD	DBAC
ACDB	BCDA	CBDA	DBCA
ADBC	BDAC	CDAB	DCAB
ADCB	BDCA	CDBA	DCBA

Проиллюстрируем сначала алгоритм построения следующей перестановки на примере перестановок из 9 компонентов, обозначенных соответственно цифрами от 1 до 9.

Первая из таких перестановок — 1 2 3 4 5 6 7 8 9. Пусть текущая перестановка из девяти компонентов — 1 9 5 8 4 7 6 3 2. Каким будет следующее значение перестановки, если мы строим ее в лексикографическом порядке (то есть в порядке возрастания величины числа, составленного из этих цифр)?

Правильный ответ — 1 9 5 8 6 2 3 4 7. Как он получается?

Прежде всего необходимо просматривать исходный массив от конца к началу, чтобы найти первое число, которое ***меньше*** предыдущего. В нашем случае — это 4 (7>6>3>2, а 4<7).

Далее среди просмотренных чисел справа от найденного 4 мы ищем последнее число которое больше 4. Это число 6 (7>4, 6>4, 3<4, 2<4).

Затем меняем эти два найденных числа (4 и 6) местами и получаем — 1 9 5 8 6 7 4 3 2.

Теперь числа (справа от 6), которые составляют убывающую последовательность (7 4 3 2), попарно меняем местами так, чтобы они составили возрастающую последовательность (2 3 4 7) — 1 9 5 8 6 2 3 4 7. Это и есть следующая перестановка. А какая перестановка будет последней для данного примера? Надеюсь, что вдумчивый читалель догадался и сам — 9 8 7 6 5 4 3 2 1.

Несколько неформально алгоритм построения следующей перестановки по текущей может быть записан следующим образом:

1. От конца к началу перестановки ищем первый элемент B[i], такой, что B[i]<B[i+1]. Запоминаем его индекс — i.
2. От элемента i+1 до конца ищем последний элемент, больший чем B[i]. Запоминаем его индекс — k.
3. Меняем местами эти элементы — с номерами i и k.
4. Всю группу элементов от i+1-го элемента до N-го попарно меняем местами (i+1-й элемент с N-м, i+2-й элемент с N-1-м и т.д.)

Формализованный алгоритм генерации всех перестановок из N элементов может быть записан следующим образом:

```
Ввод N
 Прописываем массив B последовательно числами от 1 до N
 Это первая (начальная) перестановка,  выводим ее
 Пока (истина)
   i=N
   Пока (i>0) и (B[i]>=B[i+1]), i=i-1
     Если i=0 то конец работы
     Для j от i+1 до N
       если B[j]>B[i] то K=j
     Обмен значений B[i] и B[k]
     Для j от i+1 до (i+ ((N+1-i) div 2))
       Обмен значений B[j] и B[N+i+1-j]
     Вывод текущей перестановки B
```

Понятно, что цикл попарных перестановок "хвоста" массива B нельзя делать от i+1-го до N-го элемента. Иначе элементы поменяются местами по два раза, и получится, что ничего не изменилось. Цикл нужно выполнить для половины этого "хвоста". Этому и служит несколько сложное для понимания значение конечной переменной цикла:

i+ (N+1-i) div 2.

Ниже приводится программа, генерирующая все перестановки из N компонентов, обозначенных N первыми буквами латинского афавита.

```
const
  alphabet : string[26] = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
var
  b       : array [1..100] of byte;
  N,i,j,k : byte;

  Procedure SwapB(i,k:byte);
    var x : byte;
    begin
      x:=B[i]; B[i]:=B[k]; B[k]:=x;
    end;
  Procedure WriteB;
    begin
      for i:=1 to N do write(alphabet[b[i]]);
      writeln;
    end;

begin
  readln(N);
  for i:=1 to N do b[i]:=i;
  WriteB;
  while (true) do
    begin
      i:=N;
      while (i>0) and (B[i]>=B[i+1]) do i:=i-1;
      if i=0 then exit;
      for j:=i+1 to N do
        if (B[j]>B[i]) then K:=j;
      SwapB(i,k);
      for j:=i+1 to (i+ ((N+1-i) div 2))
        do  SwapB(j,N+i+1-j);
      WriteB;
    end;
end.
```

В программе введены две процедуры —WriteB и SwapB. Процедура WriteB вызывается всякий раз, когда пост-

роена очередная перестановка. В данной программе процедура WriteB просто выводит соответствующую последовательность латинских букв.

Процедура SwapB(i,k) введена для упрощения понимания главной программы. SwapB просто обменивает значениями два элемента массива B — те, которые имеют индексы, соответствующие значениям параметров процедуры i и k.

Процедура SwapB используется в тексте программы два раза: первый — при обмене значениями двух найденных элементов с индексами i и k; второй — при обеспечении попарного обмена элементов "хвоста", в котором текущий элемент с индексом j обменивается местами со своим "партнером", находящимся на позиции N+i+1-j. Таким образом, i+1-й элемент поменяется (при j=i+1) местами с N-м элементом, i+2-й элемент (при j=i+2) с N-1-м и т.д.

Общее число перестановок из N элементов равно N! (читается N факториал). Напомним, что N! = 1*2*3*...*N.

3. Сочетания

Пусть мы имеем пять компонентов, обозначенных латинскими буквами A, B, C, D, E соответственно.

Тогда все сочетания из этих пяти компонентов по три, выписанные в лексикографическом порядке (для букв и цифр от 1 до 5) будут таковы:

```
ABC 123
ABD 124
ABE 125
ACD 134
ACE 135
ADE 145
BCD 234
BCE 235
BDE 245
CDE 345
```

Неформально алгоритм генерации последовательности чисел в лексикографическом порядке можно записать следующим образом.

Выберем наименьшие M из имеющихся N чисел и выпишем их в порядке возрастания (1 2 3 — это начальное сочетание. Очевидно, что наибольшие M чисел из имеющихся, выписанные в порядке возрастания — 3 4 5, составят последнее сочетание. Для того чтобы по текущему сочетанию получать следующее, можно поступать следующим образом.

Находим позицию в текущем сочетании, на которой не стоит последнее из возможных значений, и затем увеличиваем его на 1. А все последующие элементы сочетания получаем увеличением на 1 предыдущего элемента сочетания.

Например, пусть текущее сочетание — 1 3 5. Анализ начинаем с последней позиции сочетания. Число 5 — это последнее возможное значение, потому переходим к предыдущей позиции. Число 3 — это не последнее возможное значение для этой позиции (каковым в данном случае является число 4), поэтому мы его увеличиваем на 1 — получаем число 4. А число в следующей позиции получаем прибавлением 1 к этой 4 — и получаем число 5.

Таким образом, следующее значение будет —1 4 5.

Более формализованно этот алгоритм может быть записан следующим образом:

```
Ввод N, M  (из сколько и по сколько)
Заносим в массив B числа от 1 до M
Это первое сочетание, выводим его
Пока (истина)
  i=M
  Пока (i>0) и (b[i]=N-m+i), i=i-1
  Если i=0 то работа завершена
  B[i]=B[i]+1
  Для j от i+1 до M  B[j]=B[j-1]+1;
  Вывод B — следующей перестановки
```

Ниже приводится программа, которая считывает с клавиатуры значения N и M и выводит в лексикографическом порядке все сочетания из N первых латинских букв по M.

```
uses crt;
const
  alphabet : string[26] = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
var
  b         : array [1..100] of byte;
  N,M,i,j,k : byte;

  Procedure WriteB;
    begin
      for i:=1 to M do write(alphabet[b[i]]);
      writeln;
    end;

begin
  readln(N,M);
  for i:=1 to M do b[i]:=i;
  WriteB;
  while (true) do
    begin
      i:=M;
      while (i>0) and (b[i]=N-m+i)  do Dec(i);
      if i=0 then exit;
      Inc(B[i]);
      for j:=i+1 to M do B[j]:=B[j-1]+1;
      WriteB;
    end;
  readln;
end.
```

Напомним, что общее число сочетаний из N элементов по M может быть вычислено по формуле (где N≥M):

$$C_N^M = \frac{N!}{M!(N-M)!}.$$

(Окончание следует)

Рисунок О.Попова

HI-TECH,
E-mail: serrgio@gorki.unibel.by

НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

(Продолжение. Начало в №9/2001)

1.3. Первое погружение в регистры

#1. Наверняка вы имеете представление о том, что такое переменная. Наиболее продвинутые даже знают, что переменная имеет тип. Кажется вполне естественным, что любой высокоуровневый язык программирования позволяет создавать любое количество переменных того или иного типа...

Так вот, господа — при программировании на языке Ассемблер вас ждет большая неожиданность. Потому что для всех ваших навороченных вычислений разрешается использовать только несколько ***переменных*** с фиксированными "именами собственными" и имеющих фиксированную "длину". Эти "предопределенные переменные" называются ***регистрами***, и каждая из них имеет свою специализацию.

О специализации нам пока что говорить рано, описание наподобие "регистр-указатель базы кадра стека" вам вряд ли о чем-то скажет. Поэтому для начала познакомимся только с так называемыми ***регистрами общего назначения*** (РОН), и то не со всеми, а только с четырьмя основными, которые являются своего рода "рабочими лошадками" микропроцессора.

Вот их "имена собственные" — AX, CX, DX, BX (именно в такой последовательности они "упорядочены" в Intel'овских микропроцессорах).

А сейчас мы поближе посмотрим на этих "рабочих лошадок" микропроцессора.

1. Запустите программу DEBUG.EXE, которая находится по адресу C:\WINDOWS\COMMAND\ или C:\DOS\ (или еще где-нибудь, но если у вас не DOS и не W9x, то тут уж сами ищите).

2. Когда появится приглашение в виде "минусика", введите букву "R" (можно и "r" — регистр символов значения не имеет) и нажмите на <Enter>.

Не правда ли, весьма похоже на то, что показывают в художественных фильмах про хакеров?

```
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=18B2  ES=18B2  SS=18B2  CS=18B2  IP=0100   NV UP EI PL NZ NA PO NC
18B2:0100 6A            DB      6A
```

— Ну и что это такое? — скептически спросите вы.

— А черт его знает! — ответим мы. — Будем разбираться!

#2. То, что у вас должно появиться — это список доступных регистров и текущее значение каждого из них. Как видите, значения регистров AX, BX, CX, DX равны 0. Не правда ли, создается впечатление, что они просто-напросто ***ждут*** того, чтобы в них внесли какое-либо значение?

Природа не терпит пустоты. Писателей приводит в ужас чистый лист бумаги...

Весьма скоро и вы при виде "пустых" регистров будете испытывать непреодолимое наркотическое желание чем-нибудь их заполнить...

Однако прежде чем мы сделаем это ***в первый раз***, давайте уточним тип этих "переменных".

А он очень простой, этот тип — шестнадцатеричное число в диапазоне 0...FFFF. Или, если в BIN, то — от 0 до 1111 1111 1111 1111.

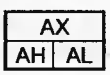
Маловато будет? Нам тоже так кажется. А вот создатели первых "IBM-совместимых" :) компьютеров посчитали, что и этого много!

— Подумать только, целый регистр использовать, если в большинстве случаев — всего-то ничего — двумя тетрадами (0...FF или 0...1111 1111) ограничиться можно! — с иронией сказал один из авторов, и тут же был упрекнут в снобизме :(

— Вы бы еще презрительно вспомнили Пифагора, "мол, с компьютером общаться не умел". Не забудьте, что при тогдашней технологической базе и это было большим прорывом. А экстенсивное расширение, например, разрядности, во-первых, нужно правильно предвидеть (вспомните, сколько в том же MS-DOS закладок на будущее, которые никуда не пошли за ненадобностью), а во-вторых, правильно оценить (в буквальном смысле). Неужели вы думаете, что, например, производители памяти не могут легким мановением руки увеличить ширину шины, соединяющей память с процессором? Могут, но во-первых — это резко повысит стоимость памяти, а во-вторых — не гарантирует повышения производительности.

О чем разговор? Да о том, что и без того жалкая и убогая 16-битная переменная еще и на две части дробится — для совместимости с языками ассемблера для предыдущих моделей процессора Intel, работавших только с 8-битными регистрами; да и просто — ради удобства...

В общем, в умных книжках рисуют вот такую вот "нездоровую" схемку:



А означает она следующее.

Физически существует один регистр — AX, а вот логически он делится на два — на старшую (AH) и младшую (AL) части (от английского — high и low).

Очевидно, что присвоить AX значение, например, 72F9h, мы можем следующими способами:

1. AX = 72F9h (одной командой);
2. AH = 72h; AL = F9h (двумя командами).

Точно так же присвоить значение 78h регистру AH можно двумя способами:

1. AH = 78h;
2. AX = 7800h.

То же самое, но для регистра AL:

1. AL = 78h;
2. AX = 0078h.

Тех, кого смущают числа с буквами, мы со зловредной ухмылкой отсылаем к прошлому номеру журнала [1], дабы вы более внимательно прочитали там главу "1.1. Система счисления".

#3. Если рассматривать регистр "целиком", то каждый из них имеет "длину" 16 бит, которые принято нумеровать справа налево. Так, для числа 2F4Dh, внесенного, например, в регистр AX, мы можем нарисовать такую вот "навороченную" табличку:

AX		2F4D															
AH	AL	2F							4D								
Значение бита		0	0	1	0	1	1	1	1	0	1	0	0	1	1	0	1
Номер бита		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Тетрады		Старшая AH				Младшая AH			Старшая AL					Младшая AL			

Внимательно смотрим на таблицу: одной шестнадцатеричной цифре соответствует тетрада двоичных цифр (4 шт., они же — 4 бита). "Емкость" регистров AH и AL — две тетрады, т. е. 8 бит. Точно такую "длину" имеют: коды символов, скан-коды клавиш, номера функций прерываний и куча всего прочего, чего вы пока не знаете.

Емкость AX (состоящего из двух половинок) — 4 тетрады, т. е. 16 бит; они же (эти 16 бит) иначе еще называются "словом"...

#4. "Принудительно" присвоить регистру значение можно при помощи той же команды "R", только с параметром "имя собственное регистра".

Например, команда

```
- R AX <Enter>
```

выбросит вам на монитор ":".

Введите после двоеточия, например, число 123 и снова нажмите на Enter:

```
:123  <Enter>
```

На дисплее опять появится приглашение "-", на которое мы отвечаем командой "R" без параметров и таким образом вновь просматриваем значения наших регистров:

```
AX=0123  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=18B2  ES=18B2  SS=18B2  CS=18B2  IP=0100   NV UP EI PL NZ NA PO NC
18B2:0100 6A            DB      6A
```

Смотрим внимательно — AX=0123, что и требовалось доказать...

ПРИМЕЧАНИЕ: Мы не рассказали вам о том, что регистры бывают 32-разрядные; умолчали о назначении и специализации отдельных регистров; даже не упомянули о флагах. Все это еще будет рассматриваться при втором и третьем "погружении" в регистры.

1.4. Первое погружение в память

#1. Первым видом памяти, с которым мы войдем (придется!) в тесный физический контакт, будет оперативная, она же — RAM (от английского — Random Access Memory). Оперативная память — это своего рода "рабочая площадка", по которой суетится этакий шустрый многорукий дядька-процессор — чего-то там собирает, от кучи к куче бегает, всех ругает...

Оперативная память — это ряд пронумерованых ячеек размером в байт. Мы можем получить доступ к первому байту памяти, ко второму, к третьему и т.д.

Короче — пришло время испробовать еще одну команду из скромного арсенала DEBUG'a! Запустите debug и введите команду D (от английского — DUMP).

"Картинка", которую вы увидели, называется "дамп памяти" (что в переводе с английского означает "свалка"), и она насыщена не только важной информацией, но и специальной низкоуровневой энергетикой. Да чего уж там греха таить — каждый ассемблерщик знает, что рассматривание дампа памяти поднимает настроение, жизненный тонус и т.д. ;)

Слева — это адрес памяти (почему он как бы из двух частей состоит, мы вам расскажем немного позже). В центре — 16 столбцов из спареных цифр...

А здесь и повториться лишний раз не грех. Каждая пара шестнадцатеричных цифр — это байт. Смотрите внимательно на дамп! Байт по адресу 100 имеет значение 6A, байт по адресу 101 — 00, байт по адресу 102 — 68... Эти "сладкие парочки" — и есть неделимая "единица адресации" оперативной памяти.

Тех, кого смущает наличие буковок в адресе, в очередной раз отсылаем ознакомиться с шестнадцатеричной системой счисления, так как все числа, отображаемые программой debug — именно шестнадцатеричные.

И, наконец, столбец справа — это символы, соответствующие шестнадцатеричным кодам центрального столбца (например, коду 6A соответствует символ J). Большинству кодов не соответствует никакой из "печатных символов" — таким в колонке справа соответствуют точки.

#2. А теперь потренируем наши пальчики дампировать память — пройдемся по некоторым "историческим местам" нашей оперативной памяти... Для этого мы будем вводить команду D с параметром.

Например, команда (параметр L8 означает "вывести 8 байтов"):

```
- D FFFF:5 L8 <Enter>
```

покажет вам системную дату в правом столбце дампа.

Короче, искателям приключений выдаем "простыню" самых интересных адресов (большинство слов в описании вам пока должны быть непонятны, но вы не пугайтесь — понимание придет!).

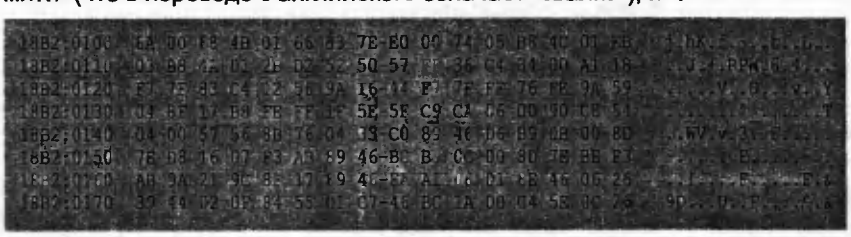
0:417 — два байта разрядов состояния клавиатуры. Они активно используются ROM-BIOS для управления интерпретаций действий клавиатуры. Изменение этих байтов изменяет значение нажатых клавиш (например, верхний или нижний регистр).

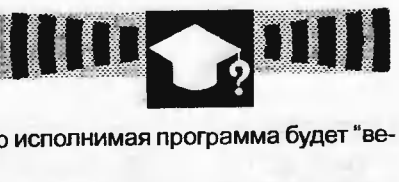
0:41A — слово по этому адресу указывает на начало буфера BIOS для ввода с клавиатуры, расположенного начиная с адреса 41E. В этом буфере хранятся и ждут обработки результаты нажатия на клавиши. Конец буфера — слово по адресу 41C.

0:43E — байт указывает, необходима ли проверка дискеты перед подводом головки на дорожку. Разряды 0...3 соответствуют дисководам 0...3. Если разряд установлен в 0, то необходима проверка дискеты. Как правило, вы можете обнаружить, что разряд установлен в 0, если при предыдущем обращении к дисководу имели место какие-либо проблемы.

Например, разряд проверки будет равен 0, если вы попытаетесь запросить каталог на дисководе, на котором нет дискеты, и затем на запрос, появившийся на экране дисплея: "Not ready reading drive B: Abort, Retry, Ignore?" вы ответите: "A".

0:44C (2 байта) — длина регенерации экрана. Это число байтов, используемых для страницы экрана. Зависит от режима.





`0:44E` (2 байта) — смещение для адреса начала текущей страницы в памяти дисплея. Этот адрес указывает, какая страница в данный момент используется (маленькая, но неприятная подробность — это смещение внутри текущего сегмента видеопамяти, без учета самого сегмента. Например, для нулевой страницы смещение всегда будет равно нулю).

`0:460` (2 байта) — размер курсора, представленный в виде диапазона строк развертки. Первый байт задает конечную, а второй — начальную строку развертки.

`0:449` — значение этого байта определяет текущий видеорежим. Для расшифровки требуется целая таблица. Например, 3 — 80-колонный текст, 16 цветов; 13h (19) — 256-цветный графический режим 320x200 и т. д.

Ну и хватит для первого раза. Кому мало — ищите дополнительную документацию (например, можете обратиться за электронными справочниками на сайт http://hi-tech.nsys.by).

ПРИМЕЧАНИЕ: Мы вам не рассказали о "прямой" и "офсетно-сегментной" адресации памяти. Мы вас не познакомили с красивой моделью "сквозной иерархической памяти". Обо всем этом еще будет вестись отдельный разговор.

1.5. Первое погружение в... программу

#1. Любая программа выполняется последовательно (мы ведь пока обсуждаем "простой" IBM PC, а не какой-нибудь крутой векторный параллельный суперкомпьютер). То есть пока не выполнилась текущая "строка" (инструкция) программы, следующая не выполнится. Совсем другой вопрос, какая "строка" будет выполнена после "текущей" (здесь мы имеем дело со всевозможными логическими "ветвлениями", "циклами" и т. д.), или же строчку из какой программы процессор выполнит следующей, а какая — будет ждать своей очереди (так называемая "многозадачность", которую пока трогать не будем — в большинстве случаев мы можем прекрасно прожить и без нее, поскольку все заботы об этом все равно берут на себя операционные системы).

Итак, у нас есть оперативная память, в которую загружается программа перед ее выполнением (сразу же по нажатию на Enter из Norton Commander). Операционная система, которая, собственно, и загружает программу, сообщает процессору, что надо начать обрабатывать команды, которые в памяти начинаются с такого-то адреса. И здесь первый подводный камень, вернее скала, которую трудно не заметить.

Начало программы в памяти процессор различает легко — ему указывает на это командный интерпретатор, а вот конец программы программист должен указывать сам!

Каким образом? А очень легко! Компьютер "распознает" как выход из программы специальную последовательность байтов. Например, для исполнимых файлов типа com (именно с этим типом файлов мы будем работать на начальном этапе) достаточно последовательности CD и 20.

Пробуем-проверяем? Ну конечно же! Только для этого вам понадобится какой-нибудь шестнадцатеричный редактор, выбирайте любой.

Все очень просто — создаем новый файл, единственным содержимым которого является последовательность CD 20, и сохраняем его как, например, myprg_1.com. Если вы позаботились о том, чтобы после CD 20 не было никаких прочих символов, то исполнимая программа будет "весить" только 2 байта.

Запускать это ваше первое творение лучше из Norton или Volcov Commander (все же это пока что DOS'овская программулька). Причем из этих двух оболочек мы настоятельно рекомендуем использовать именно Volcov Commander — в отличие от Norton, он может "на лету" выгружать резиденты из памяти, что нам вскоре очень понадобится.

Что же она делает, эта 2-байтовая малышка? А ничего, просто этот файл обладает двумя важными свойствами:

- это — программа;
- эта программа — программа с корректным выходом.

Последнее и является единственным, что она пока что может делать (корректно выгружаться из памяти)...

Еще к вопросу о выгружаемости — если после CD 20 вы напишете какую-нибудь ерунду, она все равно будет проигнорирована. Дело до ее выполнения просто-напросто не дойдет. Другое дело — если вы напишите ерунду **до**...

#2. Честно говоря, опасно при низкоуровневом программировании ерунду писать. Можно невзначай и винт отформатировать :))). Поэтому ерунду писать не будем, вернее — будем, но не совсем ерунду...

Итак, продолжим наше "извращение". Познакомимся еще с некоторыми "машинными командами" (в нашем случае — последовательностями шестнадцатеричных циферек).

- `B82301` — внести значение 0123h в AX;
- `052500` — прибавить значение 0025h к AX;
- `8BD8` — переслать содержимое AX в BX;
- `03D8` — прибавить содержимое AX к BX;
- `8BCB` — переслать содержимое BX в CX;
- `31C0` — очистка AX;
- `CD20` — конец программы. Передача управления операционной системе.

Вот и давайте создадим еще одну программу типа com со следующим "шестнадцатеричным содержимым":

`B8-23-01-05-25-00-8B-D8-03-D8-8B-CB-31-C0-CD-20`

Если вы все ввели правильно, то прога у вас без проблем запустится, а операционная система не будет ругаться... Правда, визуально (в смысле на мониторе) вы ее работу так и не заметите, но поверьте на слово — она работает! В этом вы еще убедитесь, когда посмотрите на ее работу изнутри — не различающими цветов глазами компьютера... Только сначала еще немного теории...

#3. Теперь поговорим о втором подводном камне :). Один из принципов фон Неймана звучит приблизительно так: ***машине безразлично целевое назначение данных***... Одна и та же цепочка битов может быть и машинными командами, и данными (например, символами, выраженными в виде кодов — есть такая "таблица символов ASCII", наверняка вы ее знаете).

Что из этого следует? А то, что компьютеру нужно указывать, что подразумевается под той или иной "простыней" из битов — данные или код.

На высоком уровне это делает операционная система. Например, она не пытается загрузить в память для выполнения файлы с расширениями, отличными от COM, EXE и BAT (последний вообще не из этой оперы, но принцип сохраняется).

Хотя... вы всегда можете поэкспериментировать! Смените, например, у какого-нибудь текстового файла

тип с TXT на COM и попробуйте его запустить на выполнение (хотя мы это делать настоятельно не рекомендуем!). В большинстве случаев ваш компьютер зависнет! Потому, что:

1. Он пытается интерпретировать данные как код. Соответственно, в процессор "попадает" всякая ерунда.

2. Вряд ли он натолкнется на последовательность CD 20 в вашем тексте :). Даже в том случае, если этот код выполнится "успешно" — ваша программа не возвратит управление операционной системе, а пойдет выполняться хлам, содержащийся в оперативной памяти. Как-то — остатки ранее выполненных программ, куски чьих-то данных, интерпретированные как код... и прочая многочисленная ерунда...

Почти такой же эффект, но с потенциально большей разрушительной силой может получиться, если управление получит ИСПОРЧЕННЫЙ код, который вроде бы "в основном" правильный, но часть его вместо инициализации переменных и прочих подготовительных действий в лучшем случае ничего не делает, а в худшем портит другой код и данные...

Как вам тяжело "въехать" в смысл повествования, состоящего из кусков различных книг, так и компьютеру тяжело понять подобную "мешанину". С той лишь разницей, что любую "неинтересную книгу" вы можете использовать в качестве туалетной бумаги, а вот "компутер" подобного права выбора лишен — он ***должен*** в это "въезжать", его процессор начинает перегреваться, а "мозги кипят и вытекают" через низкоуровневые порты ввода-вывода (командами IN и OUT соответственно).

#4. Еще немного идеологии. О программе, которая выполняется в памяти...

Сколько бы ни было "мозгов" в вашей навороченной тачке, ***любая*** программа для MS-DOS (и совместимых ОС) выполняется в 640 килобайтах "нижней" (или основной) памяти. Если отнять от этой цифры "резидентную часть" операционной системы, многочисленные драйвера и т.д., то оставшееся и есть ***объем памяти***, в котором выполняется ваша программа. А остальные мегабайты — это место для кэширования диска, хранения промежуточных данных и т.п.

Страшно? Медитируйте!

#5. Как уже говорилось в #3, одна и та же последовательность битов в памяти может быть:

- ***кодом*** (т.е. ЧТО компьютеру нужно делать) — последовательностью инструкций;

- ***данными*** (т.е. С ЧЕМ компьютеру нужно выполнять ту или иную работу). Именно данные являются исходной "задачей" и конечным результатом работы процессора;

- ***стеком*** — это область памяти, позволяющая писать реентерабельный/рекурсивный код и служащая для хранения адресов возврата и локальных данных и передачи параметров.

Соответственно, и программа состоит из трех частей (сегментов): сегмента данных (**data**), сегмента кода (**code**) и сегмента стека (**stack**)...

Оставим пока что "гнилой базар" про смысл словосочетаний "реентерабельный/рекурсивный код" и "адрес возврата". Чтобы не затруднять себе понимание происходящего, мы попытаемся абстрагироваться от всех этих ужасающих вещей и для начала заняться только кодом.

#6. Помните, как в конце фильма "Matrix" Нео в конце концов увидел ее — черно-зеленую "матрицу"? Сейчас с вами произойдет нечто подобное! ;)

Посмотрите на машинные коды, и "что они делают" в #2. Немножко дополним эту "простыню". Например, командой "внести значение" 1234 последовательно в каждый из "регистров общего пользования":

```
B83412 - AX=1234
BB3412 - BX=1234
B93412 - CX=1234
BA3412 - DX=1234
```

Наиболее наблюдательные должны для себя отметить, что первый байт — это команда "переместить в регистр", а второй и третий — само число, только байты почему-то "наоборот".

Однако никто не пишет программы в шестнадцатеричных редакторах! Никто! Это большая глупость! Единственное, зачем мы вам про это рассказываем — это чтобы вы поняли, что могут означать загадочные пары шестнадцатеричных цифр в дампе...

Нет необходимости заучивать, что B8 — это "переместить в регистр AX", BB — "переместить в регистр BX" и так далее... Когда-нибудь это может пригодиться тому, кто будет писать компилятор, умеющий генерировать исполняемый код, упаковщики исполняемых файлов, самомодифицирующийся код или, на худой конец, конструкторы полиморфных самошифрующихся вирусов. Но это мы оставим на будущее...

Все намного проще!

В этом вы можете убедиться, загрузив вашу программу myprg_1.com в debug (например, командной строкой "debug myprg_1.com") и введя команду "u".

А вот дальше начинается самое интересное :)))

#7. Вот что вы должны увидеть:

```
11B7:0100 B82301  MOV AX,0123 ; Внести значение 0123h в AX
11B7:0103 052500  ADD AX,0025 ; Прибавить значение 0025h к AX
11B7:0106 8BD8    MOV BX,AX   ; Переслать содержимое AX в BX
11B7:0108 03D8    ADD BX,AX   ; Прибавить содержимое AX к BX
11B7:010A 8BCB    MOV CX,BX   ; Переслать содержимое BX в CX
11B7:010C 31C0    XOR AX,AX   ; Очистка AX
11B7:010E CD20    INT 20      ; Конец программы
```

Возвратившись к #2, перенесем сюда "описание" машинных команд.

Эти mov, add, xor, int — так называемые "мнемонические команды" (более или менее понятные человеку), на основе которых формируется (это debug делает) "машинный код". Не правда ли, так намного легче?

Соответственно, вместо шестнадцатеричных кодов мы легко могли вводить эти команды при помощи команды "A" (однако этим мы займемся позже).

#8. А теперь мы выполним нашу программу ***пошагово*** — произведем так называемую "трассировку" при помощи команды "T".

Итак, вводим "T" и жмем на Enter!

Вот что мы видим:

```
AX=0123 BX=0000 CX=0010 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=11B7 ES=11B7 SS=11B7 CS=11B7 IP=0103 NV UP EI PL NZ NA PO NC
11B7:0103 052500 ADD AX,0025
```

Смотрим на значение AX и вспоминаем предыдущую инструкцию — "внести значение 0123h в AX". Внесли? И правда! А в самом низу — код и мнемоника команды, которая будет выполняться следующей...

Вводим команду "T" снова:.

```
AX=0148 BX=0000 CX=0010 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=11B7 ES=11B7 SS=11B7 CS=11B7 IP=0106 NV UP EI PL NZ NA PE NC
11B7:0106 8BD8 MOV BX,AX
```

AX=0148 — "прибавить значение 0025h к AX". Сделали? Сделали!!

Вводим команду "T" снова:.

```
AX=0148 BX=0148 CX=0010 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=11B7 ES=11B7 SS=11B7 CS=11B7 IP=0108 NV UP EI PL NZ NA PE NC
11B7:0108 03D8 ADD BX,AX
```

AX=0148=BX — "переслать содержимое AX в BX". Сделали? Сделали!!

Вводим команду "T" снова:

```
AX=0148 BX=0290 CX=0010 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=11B7 ES=11B7 SS=11B7 CS=11B7 IP=010A NV UP EI PL NZ AC PE NC
11B7:010A 8BCB MOV CX,BX
```

"Прибавить содержимое AX к BX". Оно? А то!

Вводим команду "T" снова:

```
AX=0148 BX=0290 CX=0290 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=11B7 ES=11B7 SS=11B7 CS=11B7 IP=010C NV UP EI PL NZ AC PE NC
11B7:010C 31C0 XOR AX,AX
```

"Переслать содержимое BX в CX". Сделано!

Вводим команду "T" снова:

```
AX=0000 BX=0290 CX=0290 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=11B7 ES=11B7 SS=11B7 CS=11B7 IP=010E NV UP EI PL ZR NA PE NC
11B7:010E CD20 INT 20
```

"Очистка AX"? И точно AX=0000!

Вводим команду "T" снова... И ГРОМКО РУГАЕМСЯ!!

Потому что, по идее, сейчас наша программа должна была завершиться — у нас же там код выхода прописан, а она куда лезет? NOP'ы какие-то (если продолжать команду "T" вводить), CALL 1085 (да вы продолжайте "трассировку", продолжайте!)

Для тех, кому лень продолжать жать на букву "T", введите для разнообразия команду "G" (от английского — GO). На монитор должна вывалиться надпись "Нормальное завершение работы программы".

"Уф, — должны сказать вы — Работает!"

А то!

#9. Только непонятно вот, почему вдруг между int 20 (CD 20) и надписью "Нормальное завершение работы программы" куча всяких "левых" непонятных команд (в том случае, если вы и дальше производили трассировку, а не воспользовались "халявной" командой "G")?

А потому, дорогие наши, что вы имели счастье нарваться на ***прерывание*** (interrupt)!

Понимаете ли, завершить программу — дело непростое :). Нужно восстановить первоначальное значение регистров, восстановить переменные среды и кучу всего другого! Знаете, как это сложно?

Однако эта процедура насколько сложная, настолько и ***типичная*** для исполняемых программ. А посему, разработчики операционной системы решили избавить программистов от необходимости делать это вручную, и включили эту стандартную процедуру в ядро операционной системы. И сказали: "да будешь ты (процедура обработки прерывания) вызываться как int 20, и будешь ты обеспечивать корректную передачу управления из выполняемой программы — назад в ядро". И стало так...

Ну посудите сами, должна же операционная система ну хоть что-нибудь делать!!

Литература

1. HI-TECH. Низкоуровневое программирование. — Радиомир. Ваш компьютер, 2001, №9, С.24.

(Продолжение следует)

Программирование циклических алгоритмов

А.ШАКИРИН,
г.Минск, БАТУ, каф.ВТ.

Цель этой статьи — освоить простейшие средства отладки модулей проекта и создать приложение, которое использует циклический алгоритм.

1. Отладка модулей проекта

Отладка представляет собой процесс обнаружения, локализации и устранения ошибок в проекте. Она занимает значительную часть рабочего времени программиста, нередко большую, чем разработка проекта.

Практически любой нетривиальный проект перед началом отладки содержит хотя бы одну синтаксическую или логическую ошибку.

1.1. Отладка синтаксичвских ошибок

Синтаксические ошибки заключаются в нарушении формальных правил использования операторов. Эти ошибки появляются в результате недостаточного знания разработчиком языка программирования и невнимательности при наборе операторов на экране дисплея.

Поиск синтаксических ошибок в модулях проекта осуществляется компилятором. Чтобы дать программисту как можно больше информации об ошибках, допущенных в модуле, компилятор отмечает ошибки и продолжает работу до тех пор, пока не будут обработаны все операторы модуля. Следует иметь в виду, что:

- компилятор распознает *не все ошибки*;
- некоторые ошибки могут повлечь за собой то, что правильные операторы будут восприниматься компилятором как ошибочные, и наоборот — ошибочные операторы компилятор воспримет как верные;
- ошибка в одном месте модуля может повлечь за собой серию диагностических сообщений компилятора в других местах модуля;
- из-за некоторых ошибок компиляция модуля может вообще прекращаться, и проверка последующих операторов не производится.

Информация обо всех ошибках, найденных в модуле, выводится в специальное окно, которое появляется в нижней части экрана. Каждая строка этого окна содержит имя файла, номер строки, в которой обнаружена ошибка, и характер ошибки. Если дважды щелкнуть мышью на строке с описанием ошибки, курсор установится в той строке модуля, где обнаружена ошибка. Следует исправлять ошибки последовательно, сверху вниз, и после исправления каждой ошибки компилировать программу заново. С целью сокращения

времени компиляции рекомендуется осуществлять проверку наличия ошибок в режимах Syntax Check и Compile меню Project. Для получения более полной информации о характере ошибки можно обратиться к HELP нажатием клавиши F1.

Отладка синтаксиса считается завершенной, когда после очередной компиляции в режиме Build All меню Project отсутствуют диагностические сообщения об ошибках.

1.2. Отладка логических ошибок

Логические ошибки условно можно разделить на ошибки алгоритма и семантические ошибки. Причинами таких ошибок могут быть: несоответствие алгоритма поставленной задаче, неправильное понимание программистом смысла (семантики) операторов языка программирования, нарушение допустимых пределов и правил представления данных, невнимательность при технической подготовке проекта к обработке на компьютере.

Для выявления ошибок служат *тесты*. Тест — это такой набор исходных данных, который дает результат, не вызывающий сомнений. Промежуточные и конечные результаты теста используются для контроля правильности выполнения приложения.

Составление тестов — непростая задача. Тесты должны быть, с одной стороны, достаточно простыми, чтобы результат легко проверялся, с другой — достаточно сложными, чтобы комплексно проверить алгоритм.

Тесты составляются по схеме алгоритма *до программирования*, так как составление тестов помогает выявить многие ошибки в алгоритмизации.

Количество тестов и их сложность зависят от алгоритма. Комплекс тестов должен быть таким, чтобы все ветви схемы алгоритма были пройдены, по крайней мере, по одному разу.

Несовпадение результатов, выдаваемых приложением, с результатами тестов — признак наличия ошибок. Эти ошибки проявляются в том, что результат расчета оказывается неверным, либо происходит переполнение, деление на 0 и др.

Для локализации места ошибки рекомендуется поступать следующим образом. В окне Редактора Кода установите курсор в строке перед подозрительным участком и нажмите клавишу F4 (выполнить до курсора). Выполнение приложения будет остановлено на той строке модуля, в которой был установлен курсор. Текущее значение любой переменной можно увидеть, если накрыть курсором идентификатор переменной на 1...2 с. Нажимая клавишу F8 (пошаговое выполнение), можно построчно выполнять программу, контролируя содержимое переменных и правильность вычислений.

2. Пример создания приложения

Необходимо создать Windows-приложение, которое выводит таблицу значений функции

$$Y(x) = (1 - \frac{x^2}{2})\cos x - \frac{x}{2}\sin x$$

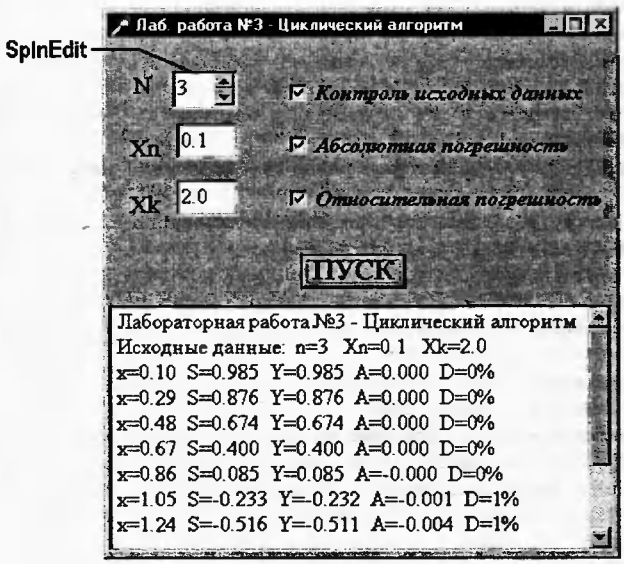
и ее разложение в ряд в виде суммы

$$S(x) = \sum_{n=0}^{n} (-1)^n \frac{2n^2+1}{(2n)!} x^{2n}$$

для значений x от x_n до x_k с шагом $h=(x_k - x_n)/10$.

В панели интерфейса следует предусмотреть возможность управления выводом исходных данных и погрешности вычислений.

Один из возможных вариантов панели интерфейса создаваемого приложения показан на рисунке.



2.1. Размещение компонентов на Форме

Вместо компонента Edit используем компонент SpinEdit, который обеспечивает отображение и редактирование целого числа с возможностью его изменения посредством двойной кнопки.

Компонент SpinEdit находится на странице Samples Палитры Компонентов. В тех случаях, когда объем выводимой информации превышает размер поля компонента Memo, целесообразно снабдить его линейками прокрутки. В свойстве ScrollBars компонента Memo1 установим значение ssVertical — появится вертикальная линейка прокрутки. Присвоим модулю имя UnCiklAlg.

2.2. Текст модуля UnCiklAlg

```
Unit UnCiklAlg;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls, ExtCtrls, Spin;
type
  TForm1 = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    SpinEdit1: TSpinEdit;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
  SpinEdit1.text:='3'; // начальное значение N
  Edit1.text:='0.1';   // начальное значение Xn
  Edit2.text:='2.0';   // начальное значение Xk
  Memo1.Clear;
  Memo1.Lines.Add('Лабораторная работа №3 - Циклический алгоритм');
end;
procedure TForm1.Button1Click(Sender: TObject);
 var xn,xk,x,h,c,s,y,al,del:extended;
     n,k:integer;
begin
 n:=StrToInt(SpinEdit1.Text);
 xn:=StrToFloat(Edit1.Text);
 xk:=StrToFloat(Edit2.Text);
 if CheckBox1.Checked then
   Memo1.Lines.Add('Исходные данные:  n='+IntToStr(n)+
     '   Xn='+FloatToStrF(xn,ffFixed,6,1)+
     '   Xk='+FloatToStrF(xk,ffFixed,6,1));
 h:=(xk-xn)*0.1;          // шаг h
 x:=xn;
 repeat                   // цикл по x
  c:=-x*x*0.5;
  S:=1;
 for k:=1 to n do
  begin
   s:=s+c*(2*k*k+1);
  c:=-c*x*x/((2*k+1)*(2*k+2));
  end;
 y:=(1-x*x*0.5)*cos(x)-0.5*x*sin(x);
 if CheckBox2.Checked then
   if CheckBox3.Checked then
     begin
       al:=s-y;                  // абсолютная погрешность
       del:=abs((s-y)/y)*100;  // относительная погрешность
       Memo1.Lines.Add('x='+FloatToStrF(x,ffFixed,6,2)+
                  '  S='+ FloatToStrF(s,ffFixed,6,3)+
                  '  Y='+ FloatToStrF(y,ffFixed,6,3)+
                  '  A='+ FloatToStrF(al,ffFixed,6,3)+
                  '  D='+ FloatToStrF(del,ffFixed,6,0)+'%');
    end
                                          else
    begin
     al:=s-y;
 Memo1.Lines.Add('x='+FloatToStrF(x,ffFixed,6,2)+
                  '  S='+ FloatToStrF(s,ffFixed,6,3)+
                  '  Y='+ FloatToStrF(y,ffFixed,6,3)+
                  '  A='+ FloatToStrF(al,ffFixed,6,3));
    end
                                        else
   if CheckBox3.Checked then
     begin
     del:=abs((s-y)/y)*100;
    Memo1.Lines.Add('x='+FloatToStrF(x,ffFixed,6,2)+
                  '  S='+ FloatToStrF(s,ffFixed,6,3)+
                  '  Y='+ FloatToStrF(y,ffFixed,6,3)+
                  '  D='+ FloatToStrF(del,ffFixed,6,0)+'%');
    end
                                          else
      Memo1.Lines.Add('x='+FloatToStrF(x,ffFixed,6,2)+
                  '  S='+ FloatToStrF(s,ffFixed,6,3)+
                  '  Y='+ FloatToStrF(y,ffFixed,6,3));
  x:=x+h;
 until x>xk;
end;
end.
```

3. Индивидуальные задания

Необходимо создать Windows-приложение для вычисления соответствующего выражения из таблицы, протестировать его работу и вывести на экран таблицу значений функции $Y(x)$ и ее разложения в ряд $S(x)$ для значений x от x_n до x_k с шагом $h=(x_k - x_n)/10$. Близость значений $S(x)$ и $Y(x)$ во всем диапазоне значений x указывает на правильность вычисления $S(x)$ и $Y(x)$.

№	x_n	x_k	S(x)	n	Y(x)
1	0,1	1	$x-\frac{x^3}{3!}+\ldots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}$	16	$\sin x$
2	0,1	1	$1+\frac{x^2}{2!}+\ldots+\frac{x^{2n}}{(2n)!}$	10	$\frac{e^x+e^{-x}}{2}$
3	0,1	1	$1+\frac{\cos\frac{\pi}{4}}{1!}x+\ldots+\frac{\cos n\frac{\pi}{4}}{n!}x^n$	12	$e^{x\cos\frac{\pi}{4}}\cos(x\sin\frac{\pi}{4})$
4	0,1	1	$1-\frac{x^2}{2!}+\ldots+(-1)^n\frac{x^{2n}}{(2n)!}$	8	$\cos x$
5	0,1	1	$1-\frac{x^2}{1!}+\frac{x^4}{2!}-\frac{x^6}{3!}+\ldots+(-1)^n\frac{x^{2n}}{n!}$	14	e^{-x^2}
6	0,1	1	$x+\frac{x^3}{3!}+\ldots+\frac{x^{2n+1}}{(2n+1)!}$	8	$\frac{e^x-e^{-x}}{2}$
7	0,1	1	$\frac{x^3}{3}-\frac{x^5}{15}+\ldots+(-1)^{n+1}\frac{x^{2n+1}}{4n^2-1}$	12	$\frac{1+x^2}{2}arctgx-\frac{x}{2}$
8	0,1	1	$1+\frac{2x}{1!}+\ldots+\frac{(2x)^n}{n!}$	10	e^{2x}
9	0,1	1	$1+2\frac{x}{2}+\ldots+\frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n$	14	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$
10	0,1	0,5	$x-\frac{x^3}{3}+\ldots+(-1)^n\frac{x^{2n+1}}{2n+1}$	15	$arctgx$
11	0,1	0,8	$\frac{x^2}{2}-\frac{x^4}{12}+\ldots+(-1)^{n+1}\frac{x^{2n}}{2n(2n-1)}$	10	$xarctgx-\ln\sqrt{1+x^2}$
12	0,1	1	$-\frac{(2x)^2}{2}+\frac{(2x)^4}{4}-\ldots+(-1)^n\frac{(2x)^{2n}}{(2n)!}$	8	$1(\cos^2 x-1)$

Литература

1. В.В.Феофанов. Delphi 3. Учебный курс. — М.: Нолидж, 1981.

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

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

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

ОБ ОДНОМ МЕТОДЕ КРАСИВОГО ВЫВОДА ТЕКСТА

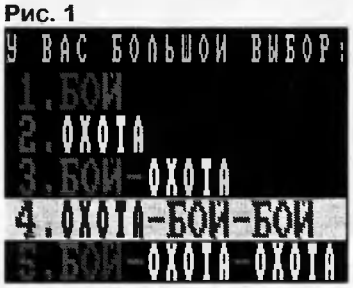
Часть 1

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

С необходимостью выводить текст более красиво я столкнулся при написании своей второй игры — "Sea Fight" (Морской Бой) — для компьютера "ZX-Spectrum". Этот проект, к сожалению, так и не был завершен — есть версия игры, которую даже демонстрационной называть язык не поворачивается; впрочем, это неважно.

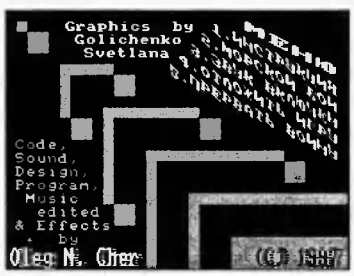
Возникла идея — создать небольшую, но симпатичную процедуру, которая при самых минимальных усилиях позволит хорошенько украсить игрушечку текстами. Изюминка способа — каждый пиксел символа на экране представляется спрайтом произвольного размера, можно обычной точечкой, а можно матрицей M x N точек. Естественно, одного цвета. При этом, в целях уменьшения затрат, пользуемся обычным шрифтом Спека — 8 x 8. Применение спрайтов позволяет использовать в текстах всяческие штриховки, затенения и текстуры. Возможны увеличенные, и даже весьма значительно, "декоративные" символы. Взаимное расположение спрайтиков задается параметрами. Причем, параллельно или перпендикулярно друг другу — каждый своим. И буковки при выводе текста — каждая дислоцируется одна относительно другой также под влиянием параметров. А это уже позволяет размещать тексты не только как обычно — "в линеечку" по горизонтали, но и по вертикали, а кроме того, и вовсе уж экзотично — по диагонали. А если еще задать расстояние между точками буковок отрицательным, то можно отображать их и зеркально. Ну, в общем, при самых что ни есть минимальных усилиях (фонты 8 x 8, несложная и мало требовательная к ресурсам процедурка вывода двухцветных спрайтов и сама NiceType) можно вовсе даже недурственный дизайн сваять для своей программы.

Рис. 1



И в итоге вот что получилось (рис.1).

Рис. 2



А вот пример текста побольше (рис.2).

Это потом уже мне на глаза попалась статья [1], где решается примерно та же задача, но там все более серьезно: программа вся объектно-ориентированная — листинг на полных пяти журнальных страницах. И много из того, что может та замечательная программка, вполне по плечу и нашей! Например, вывод текста под произвольным углом, изменение масштаба, вращение. Ну а в той программе есть дополнительная возможность прорисовки текста по дуге окружности, к тому же, применяются стандартные векторные BGI-фонты фирмы Borland.

А сейчас некоторые шаги к реализации.

"Точка, точка, запятая...". Вывод пикселов, так место или время?

Какой бы восхитительной ни была картинка на экране, она сформирована из точек — пикселов. Насколько более быстро происходит их вывод на экран, настолько более эффективной становится и работа программы. И уже появляется возможность "подвесить" на обработчик IM2 звучную мелодию, эффекты, да и в прерывание уложиться. Естественно, это справедливо лишь в общем, т.к. основной вывод графики обычно производится более комплексными способами — поблочно, с использованием указателя стека SP, или, как минимум, побайтно. Кстати, есть еще очень интересные способы заполнения экрана с помощью команд SET n,(HL) и RST n,(HL).

Но пикселы, наряду с их "усовершенствованием" — чанками и спрайтами — еще долго будут притягивать пытливые умы кодеров.

Поговорим о МАКСИМАЛЬНО эффективных подпрограммах для вывода точки. О быстрой графике можно писать много и очень много, да и вообще, это тема даже не для отдельной статьи. Так что сообщу лишь, что максимально БЫСТРУЮ процедуру вывода точки, которую только смог найти, я видел в первом номере электронного журнала ZX-Guide. О ней поведал Alone Coder (Дмитрий Быстров из Рязани). Тот, без сомнения уникальный, хотя и разовый код ставит по координатам, заданным в регистрах L=Y, E=X, в произвольном месте экрана (и даже за экран) точку за 58 (!) тактов! Подпрограмме, правда, чтобы действовать так быстро, необходимо иметь в памяти аж 4 таблицы.

Если оптимизация идет по размеру кода (а бывает и так), то имеет смысл применить процедуру Plot_sub, находящуюся в ПЗУ Спектрума по адресу #22E5. Перед ее вызовом необходимо поместить содержимое X в C и Y в B.

Однако у этой процедурки, наряду с ее очень невысоким быстродействием, есть один непростительный недостаток — рисование точек возможно не по всему экрану, а лишь в координатной сетке 256 x 176. А начало координат помещено в левом нижнем углу, выше двух строк, зарезервированных для системных сообщений. Отсутствие возможности выводить точки штатными средствами (подпрограммами ПЗУ и BASIC) во всем экранном пространстве — несомненный просчет создателей Спектрума, тем более, что это легко исправить заменой #AF на #BF по адресу #22AB. Но мы попытаемся решить эту задачу программно и минимальной кровью:

```
; Процедура, которая использует Plot_sub
; из ПЗУ для полноценной работы
; в координатной сетке 256 x 192.
Plot
        LD      A,#BF   ; Уст. новый предел для экр. вывода.
        CALL    #22AC   ; Вычисл. с пом. Pixel_Add адрес точки.
        JP      #22EC   ; Нить - "модифицированному" Plot_sub.
```

Восемь байтов! Короче, кажется, уже невозможно.

Такой вывод точки уместен в очень небольших и некритичных по времени выполнения программах — загрузчиках или простеньких интро. В серьезных же демо или играх, да еще и 3D, ее неторопливость совершенно неприемлема. Но сейчас, чтобы не перегружать код, воспользуемся именно этой подпрограммой.

Спрайты, эльфы, духи ночи

Рисовать спрайты можно по-разному. Для Спектрума широко применяются многоцветные посимвольные спрайты с атрибутами, вывод которых не представляет особой трудности, так как они обычно выровнены на границу байта — 32 x 24 (или

хотя бы 32 x 192), и работа происходит без побитовых операций — сдвигов, маскировок и т. п.

Но нам нужно получить на экране с *точностью до пиксела* прозрачные цветные изображения монохромной пиктограммы размером M x N пикселов. Это означает, что все ненулевые биты пиктограммы изображаются на экране пикселами одного цвета, а нулевые биты оставляют соответствующие им экранные пикселы без изменения. Я решил эту задачу, и, как мне кажется, довольно компактно. Код, правда, получился не особенно быстрым. Что? Вы хотите более оптимально? Так действуйте! В мои же скромные задачи входит осветить идею, а не поразить всех крутостью кода. Но кое-что все-таки скажу — как правило, выгоднее код преобразовывать в данные (всяческие таблицы переходов, адресов, ссылок и т.п.). Однако оказывается, что бывают ситуации, когда выгодней данные преобразовывать в код. Это дарит ощущение полета [2].

```
mSprite
; Процедура выводит монохромный спрайт длиной и шириной DE,
; находящийся по адресу HL, пикселами с координатами в BC.
        LD   A,8
        PUSH HL
cX      POP  HL
        PUSH HL
        PUSH DE
        PUSH BC
        PUSH AF
cY      RLC  (HL)
        JR   NC,noPlot
        PUSH BC
        PUSH DE
        PUSH HL
        CALL Plot
        POP  HL
        POP  DE
        POP  BC
NoPlot  INC  HL
        DEC  B
        DEC  D
        JR   NZ,cY
        POP  AF
        POP  BC
        INC  C
        POP  DE
        DEC  A
        JR   NZ,cX
        POP  AF
        DEC  E
        JR   NZ,mSprite
        RET
```

Немножко подробнее о параметрах. С координатами все, вроде, яснее некуда — Y = B (0...191), X = C. При выходе спрайта за пределы экрана будет получена ошибка "Integer out of range" — ее генерирует Plot_sub. D — это высота спрайта в пикселах, E — ширина спрайта в знакоместах. Сам спрайт имеет в памяти такой порядок следования байтов (старшие биты в маске на экране всегда расположены левее):

1	7	13
2	8	14
3	9	15
4	10	16
5	11	17
6	12	18

А не хватит ли огород городить?

Можно с уверенностью сказать, что оформление программ красивенькими текстами самого разнообразного начертания придает им некую вполне ощутимую элегантность. Особенно это уместно в интерактивных программах, всячески использующих текстовый интерфейс, всевозможных меню, диалогах и т.д.

Но немедленно перейдем к нашей генеральной процедуре:

```
nType
; Процедура NiceType - красивый вывод текста.
        POP     HL              ; Извлекаем постпараметры,
        LD      E,(HL)          ; скомпилированные после
                                ; CALL nType:
        INC     HL              ; X, затем
        LD      D,(HL)
        INC     HL              ; Y - пикс. координаты начала текста.
        LD      (XYt+1),DE
        LD      E,(HL)          ; Расстояние между буквами
        INC     HL              ; по горизонтали,
        LD      D,(HL)
        INC     HL              ; по вертикали.
        LD      (HLt+1),DE
        LD      E,(HL)          ; Наклон букв по
        INC     HL              ; горизонтали.
        LD      D,(HL)          ; Расстояние между спрайтами букв
        INC     HL              ; по вертикали.
        LD      (PXt+1),DE
        LD      E,(HL)          ; Расстояние между спрайтами букв
        INC     HL              ; по горизонтали.
        LD      D,(HL)          ; Наклон букв
        INC     HL              ; по вертикали.
        LD      (CRt+1),DE
        LD      A,(HL)
        LD      (HGt+2),A       ; Размер спрайта в байтах.
        INC     HL
        LD      (ADt+1),HL      ; Адрес начала спрайта.
        LD      C,A
        LD      B,0
        ADD     HL,BC           ; Адрес начала текста.
ChrO    LD      A,(HL)          ; Начинаем вывод текста
        OR      A               ; посимвольно.
        PUSH    HL              ; Если код символа равен 0,
        RET     Z               ; завершить процедуру.
        LD      L,A
        LD      H,0             ; Вычислим адрес
        ADD     HL,HL           ; матрицы
        ADD     HL,HL           ; нужного нам
        ADD     HL,HL           ; символа
        LD      DE,(#5C36)      ; по формуле:
        ADD     HL,DE           ; AddrMtx = 8 * SymC + (CHARS).
XYt     LD      DE,0
        LD      C,8             ; Цикл построкового вывода.
bgn     PUSH    HL
        PUSH    DE
        LD      A,(HL)
        LD      B,8
do8     PUSH    BC              ; Влож. цикл постолбцового вывода.
        RLCA                    ; Циклически прокручиваем
        PUSH    AF              ; байт матрицы фонта.
        JR      NC,CRt          ; Если бит матрицы включен,
        PUSH    DE
        LD      C,E
        LD      B,D
ADt     LD      HL,#3C00
HGt     LD      DE,#101
        CALL    mSprite         ; выводим спрайт!
        POP     DE
CRt     LD      HL,8            ; Модифицируем локальные
                                ; координаты
        ADD     HL,DE           ; в соответствии с CRt.
        EX      DE,HL
        POP     AF
        POP     BC
        DJNZ    do8             ; Конец цикла постолбцового вывода.
        POP     DE
PXt     LD      HL,8            ; Модифицируем глобальные координаты
        ADD     HL,DE           ; в соответствии с PXt.
        EX      DE,HL
        POP     HL
        INC     HL
        DEC     C
        JR      NZ,bgn          ; Конец цикла построкового вывода.
HLt     LD      HL,85           ; Вычислим новые экр. координаты
        ADD     HL,DE           ; следующей буковки для вывода.
        LD      (XYt+1),HL
        POP     HL              ; !!! ВНИМАНИЕ !!!
        INC     HL              ; В парных параметрах при мл. байте<0
        JR      ChrO            ; ВСЕГДА получаем перенос в старший!
```

В компьютерах, которые могут похвалиться мегабайтами, для придания программному интерфейсу большей выразительности применяют не один десяток шрифтов. Там это оправдано. Ну а на старом добром Спекки мы добьемся потрясающих результатов и с помощью преобразований. И стилизуем где нужно, и укрупним, и "*курсив*" будет, и "полужирный", и "**жирный**". И все они образованы из одного фонта.

Этот вариант процедуры, как и все в этом мире, можно улучшить. Помимо значительного ускорения ее работы, можно увеличить и функциональность NiceType путем введения управляющих кодов, чтобы, например, прямо в цепочке символов задавать новые координаты текста, цвет paper и ink, вкл/выкл inverse и over (иногда это очень полезно), табуляцию, да и, наконец, возможность тут же начать новый текст с другим спрайтом и аргументами.

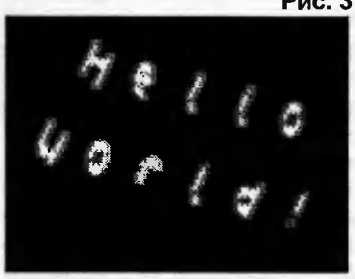
Надо все же сказать об одном ограничении, которое я сознательно ввел в NiceType, чтобы упростить ее. Это фиксированная — в одно знакоместо (8 пикселов) — ширина спрайта. Тогда это казалось вполне достаточным. Но не беда! Очень легко обходимо. Доверяю исправить это тебе, читатель... :)

Как же все это "кушать"?

Ничего более сложного, чем "Hello World!", и не предвидится! Вот достойный пример использования NiceType:

```
HELLO
        CALL    nType
        DEFB    64,180,53,22
        DEFB    -2,-5,2,-2
        DEFB    8,#5A,#A5,#5A,#A5,#5A,#A5,#5A,#A5
        DEFM    "Hello World!"
        DEFB    0
        RET
```

Рис. 3



А вот и результат работы примера (рис.3).

При "перетаскивании" программки на Паскаль я кое-что несколько улучшил. Добавилась возможность выбирать размер BIOS-шрифта для текста. Далее, некоторые параметры я сделал вещественными. Это позволяет рисовать текстики, к примеру, наклоненными не поступенчато, а более плавно. И уже можно варьировать аргументы так, что каждая последующая буква будет выше или ниже не именно на пиксел, а более сглаженно — пиксел на два, десять или сколько угодно символов. В спектрумовском варианте процедуры это не реализовано, но здесь ведь она занимает меньше двух сотен байтов, а какие зато возможности!

Но о килобайтах, мегагерцах и Турбо Паскале мы поговорим во второй части статьи.

Хочу также поблагодарить В.С.Медноногова (CopperFeet) за боевые заслуги перед отечеством :=) , а также за отличный кросс-ассемблер для процессора Z80.

От редакции. На сайте журнала (http://radio-mir.com) выложены: nicetype.trd — готовый к запуску пример Hello World! в формате TR-DOS disk для распространенных эмуляторов "ZX-Spectrum", а кроме того, иллюстрации к статье в формате экрана Spectrum; nicetype.a80 — исходный текст описываемой процедуры и небольшой пример ее использования; asm80.exe — Ассемблер 128 для Z80 V1.7 (c) 1995,96,97,99 Медноногов В.С. (CopperFeet) (для трансляции nicetype.a80 в объектный код).

Литература

1. А. Тихонов. И надпись написал... — Монитор, 1993, №3.

2. М. Эйбраш. Быстродействующие адаптеры VGA и трехмерная анимация. — Журнал д-ра Добба, 1993, №1.

3. И. Рощин. Секреты текстового вывода. — Радиолюбитель. Ваш компьютер, 2001, №№2, 3.

Как "завалить" баг 2000 года по имени Windows

Д.ЯМАЙКИН,
E-mail: thydmirty@usa.net

Про баг — это я, конечно, загнул. Скорее, это вирус :-). Впрочем, шутки шутками, а задача весьма интересная — что нужно сделать для того, чтобы система "упала"? И если с Win98 (а скорее всего, и с Win95) задача решалась тривиально — int 10h, и Виндоуз, кажется, даже мышкой не шевелила, то к 2000 г. Майкрософт справилась-таки с этой сложной задачей, и Винда "падать" перестала. Пора придумывать что-то новенькое.

Мое новенькое вылилось в "отстрел" сообщений. Как известно, вся система на них держится, и если вдруг (ни с того, ни с сего) сообщения перестанут доходить до цели, то и программы (да и сама система) начнут работать некорректно:

```
#include "stdafx.h"
#include "windows.h"

long __stdcall MHook(int code,
WPARAM wParam,
LPARAM lParam)
{

        do
        {
        }
        while(true);

        return 0;
}

int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine,
                     int       nCmdShow)
{
    int gm_result;
      MSG msg;

      SetWindowsHookEx(WH_GETMESSAGE,&MHook,hInstance,0);

    do
        {
      gm_result = GetMessage(&msg, NULL, 0, 0);
      }

      while(true);

      return 0;
}
```

Это простенькое API-приложение Винду не "убивает", но "сажает в мешок" так, что система ничего не видит и не слышит (кроме, естественно, "пинков" в виде окошка Windows Security — Alt+Ctrl+Del). Task Manager можно запустить, но он погоды не делает, так как сам работает по той же схеме, что и остальные приложения, то есть красиво "висит", даже не утруждая себя перерисовкой — это ведь тоже сообщение.

Механизм работы очень прост. API-функция

HHOOK SetWindowsHookEx(int *idHook*, **HOOKPROC** *lpfn*, **HINSTANCE** *hMod*, **DWORD** *dwThreadId*);

ставит функцию-ловушку *lpfn* по поводу *idHook* . Поводами являются различные системные происшествия — мышка "дернулась", на кнопку кто-то (подсказываю — юзер) "наступил", сообщение прошло и т.д. Параметр *dwThreadId* указывает на поток, для которого ставится ловушка, если его значение — ноль, то для всей системы.

Что происходит дальше? Ловушка стоит. По системе проходит сообщение, вызывается функция, и... Дальше сообщение не идет, потому что сначала оно должно быть обработано функцией, а она его будет обрабатывать до Reset.

Вот, собственно, и все..., Win2000 даже перегрузиться не в состоянии.

Интересен еще такой факт — Win98 этой программой не "убьешь", даже не "поцарапаешь". Запустили задачу, сняли..., работаем дальше — вот такая она "крутая".

Вывод — раньше и нарзан был крепче, и Виндоуз лучше.

Литература

1. MSDN Library от Visual Studio 6.

Автоматическое определение кодировки текста

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

При написании программы для работы с текстовыми файлами мне надо было учесть, что они могут быть в одной из двух кодировок — ALT (так называемая альтернативная кодировка) или WIN (кодировка русской версии Windows). Указывать кодировку каждого текстового файла вручную не хотелось, и я решил реализовать в программе автоматическое определение кодировки. Итак, спешу поделиться результатами своих исследований!

Коды русских букв, как известно, расположены во второй половине кодовой таблицы. Посмотрим, как различаются кодировки ALT (рис.1) и WIN (рис.2), для удобства показаны только русские буквы.

Рис. 1

	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. 2

	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	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Что мы видим? Только коды E0h...EFh в обеих кодировках соответствуют русским буквам, но разным: для ALT это буквы "р"..."я", а для WIN — буквы "а"..."п".

Теперь вспомним, что в тексте на русском языке различные буквы встречаются с разной частотой. Этим мы и воспользуемся для определения кодировки текста.

В табл.1 приведены данные о том, как часто встречается в текстах каждая из букв русского языка. Буквы упорядочены по убыванию встречаемости. Данные были получены в результате анализа текстового файла длиной 126 Кб, в котором буквы "а"..."я" занимали 69% объема.

Пусть c1, c2 — два байта из диапазона E0h...EFh. В кодировке ALT им соответствуют буквы a1 и a2, в кодировке WIN — w1 и w2. Частоты встречаемости этих букв в некотором "среднем" тексте — f(a1), f(a2), f(w1) и f(w2).

Выберем c1 и c2 так, чтобы в кодировке ALT буква, соответствующая c1, встречалась чаще, чем буква, соответствующая c2 (т.е. f(a1) > f(a2)), а в кодировке WIN, напротив, буква, соответствующая c1, встречалась реже, чем буква, соответствующая c2 (т.е. f(w1) < f(w2)).

Тогда для определения кодировки достаточно подсчитать, сколько раз в тексте встретится байт, содержащий c1 (обозначим это число n(c1)), и сколько раз — c2 (обозначим это число n(c2)). Если n(c1) > n(c2), будем считать, что текст — в кодировке ALT, иначе — в кодировке WIN. При равенстве n(c1) и n(c2) будем выбирать кодировку случайным образом, с равной вероятностью ALT и WIN.

Очевидно, что существует не одна пара (c1, c2), обладающая указанным выше свойством. Какую же из них выбрать, чтобы вероятность правильного определения кодировки была наибольшей?

При определении кодировки возможны четыре различных варианта:

1. Текст в кодировке ALT правильно распознан как ALT.
2. Текст в кодировке WIN правильно распознан как WIN.
3. Текст в кодировке ALT ошибочно распознан как WIN.
4. Текст в кодировке WIN ошибочно распознан как ALT.

Уменьшив вероятность вариантов 3 и 4, мы тем самым увеличим вероятности вариантов 1 и 2, а значит, повысим надежность определения кодировки.

Как мы можем уменьшить вероятность варианта 3 (текст в кодировке ALT ошибочно распознается как WIN)? Очевидно, что для этого разница между f(a1) и f(a2) должна быть как можно большей. Ведь в реальном тексте, кодировка которого распознается, частоты встречаемости букв a1 и a2 не будут точно совпадать с f(a1) и f(a2). И если f(a1) и f(a2) достаточно близки, то легко может случиться так, что в тексте окажется больше букв a2, чем a1, в результате кодировка будет определена неправильно.

Совершенно аналогично, для уменьшения вероятности варианта 4 разница между f(w1) и f(w2) также должна быть как можно большей.

Итак, мы должны максимизировать величины |f(a1)-f(a2)| и |f(w1)-f(w2)|, или, что то же самое, минимизировать обратные величины. Выберем для

Табл. 1

Буква	частота встречаемости, %
о	10,92
а	8,89
е	8,10
н	6,43
и	6,39
л	5,87
т	5,76
с	5,11
к	4,57
р	4,16
в	3,65
м	3,08
д	3,06
у	3,03
п	2,71
ь	2,32
ы	2,12
з	2,00
я	1,88
г	1,69
ч	1,67
б	1,55
й	1,19
ш	1,01
ж	0,92
х	0,84
ю	0,33
ц	0,29
щ	0,26
э	0,14
ф	0,06
ъ	0,03

Табл. 2

c1, c2	a1, a2	w1, w2	$\frac{1}{\lvert f(a1)-f(a2)\rvert} + \frac{1}{\lvert f(w1)-f(w2)\rvert}$
E2h, EEh	т, ю	в, о	529
E1h, EEh	с, ю	б, о	553
E1h, E5h	с, х	б, е	782
E1h, EDh	с, э	б, н	826
E2h, E5h	т, х	в, е	919
E1h, E8h	с, ш	б, и	1023
E1h, EAh	с, ъ	б, к	1486
E3h, EEh	у, ю	г, о	1493
E2h, EDh	т, э	в, н	1612
E3h, EDh	у, э	г, н	1647

этого следующий критерий — сумма квадратов обратных величин должна быть наименьшей.

наоборот — в кодировке WIN.

На рис.3 показана зависимость вероятности правильного определения

Как видим, при уменьшении длины текста вероятность стремится к 50% — такой же точности мы могли бы добить-

Рис. 3

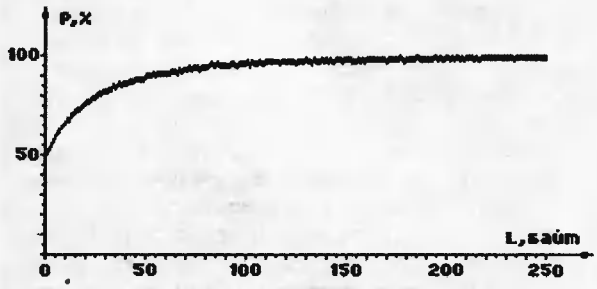
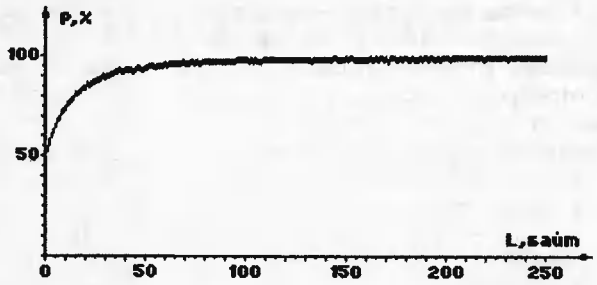


Рис. 4



Переберем (конечно, не вручную, а с помощью специально написанной программы) все возможные пары (c1, c2), удовлетворяющие условиям f(a1) > f(a2) и f(w1) < f(w2), чтобы выбрать из них лучшие в смысле вышеописанного критерия . В табл.2. приведены десять лучших пар (c1, c2): в первой строке — самая лучшая, в последующих — чем ниже, тем хуже.

Наилучшей парой оказывается (E2h, EEh). Таким образом, если в тексте больше байтов E2h, чем EEh, то считаем, что этот текст в кодировке ALT; если

кодировки (P) от длины текста (L). В анализируемых текстах буквы "а"..."я" составляли в среднем 70%, тексты в кодировках ALT и WIN встречались с равной вероятностью, определение кодировки происходило по паре байтов (E2h, EEh). Очевидно, что чем меньше анализируемый текст, тем больше (в среднем) частоты встречаемости букв a1 и a2 в нем будут отличаться от f(a1) и f(a2) (а частоты встречаемости букв w1 и w2 — от f(w1) и f(w2)), и тем меньше вероятность правильного определения кодировки (рис.3).

ся, просто определяя кодировку подбрасыванием монетки.

Также понятно, что при уменьшении доли букв "а"..."я" в тексте (скажем, если текст содержит и русские, и английские слова или насыщен псевдографикой) вероятность правильного определения кодировки снижается.

Чтобы ее увеличить, можно последовательно определять кодировку по каждой из трех наилучших пар: (E2h, EEh), (E1h, EEh) и (E1h, E5h), а окончательный результат устанавливать методом голосования (рис.4).

ПИНГВИН-СПАСАТЕЛЬ

Никто не застрахован от катастроф на жестком диске, когда для восстановления информации потребуется загрузочная дискета. Windows при установке каждый раз предлагает создать такую дискету, но даже если вы ее когда-то и создали, попробуйте теперь вспомнить, где она находится, от какой именно версии, и все ли драйвера вновь установленных устройств на нее перенесены. Впрочем, в подобной ситуации может помочь универсальная загрузочная дискета с системой Linux.

Среди советов по восстановлению информации, приведенных Джином Уилбурном [1], особого внимания заслуживает описание использования одного из вариантов такой дискеты — Tom's Root Boot diskette. Дистрибутив tomsrtbt можно скачать с сайта www.toms.net/rb/ где приведена также инструкция по инсталляции из среды Linux или DOS (необходима "чистая" сессия DOS после перезагрузки, а не окно DOS под Windows9x). Загрузочная дискета форматируется на 1,7 Мб и, кроме ядра Linux, содержит почти две сотни различных драйверов и утилит. Если этого не достаточно, на упомянутом выше сайте выложено для загрузки еще большое количество полезных программ.

Рассмотрим действия пользователя при некоторых обычных неприятностях. Например, у вас была система с двойной загрузкой Linux/Windows, и вы решили обновить версию Windows. Бац! — и загрузочного сектора как не бывало (Windows всегда переписывает MBR— загрузочный сектор диска). Раздел Linux не поврежден, но он недоступен. Нужно запустить lilo, чтобы восстановить двойной загрузчик в MBR. С "родным" boot-диском это бы не составило проблемы. А вот пример использования для этой цели tomsrtbt (предполагается, что корневая система Linux — /dev/hda4):

```
# mount -t ext2 /dev/hda4 /mnt
# chroot /mnt
# /sbin/lilo
Added linux *
Added dos
# exit
# cd/
# umount mnt
```

Другая ситуация: вы сделали опечатку в /etc/fstab, когда реконфигурировали систему, и в качестве загрузочного устройства вместо /dev/hda4 написали /dev/hda44. Система больше не грузится. Вставьте дискету tomsrtbt, перезагрузитесь, войдите в систему как root, достаньте дискету из дисковода и запустите редактор:

```
# mount -t ext2 /dev/hda4 /mnt
# vi /mnt/etc/fstab
# umount /mnt
```

При некорректном выключении компьютера, например, при пропадании сетевого питания, диск может быть поврежден так, что разделы станут недоступными. Для восстановления поврежденного раздела (например, /dev/hda2) можно использовать утилиту e2fsk с диска tomsrtbt:

```
# e2fsk -f /dev/hda2
```

Эту команду можно повторять несколько раз.

Если раздел полностью починить не удается, иногда все же можно спасти несколько важных файлов, переписав их на дискету. После загрузки tomsrtbt загружает себя полностью в память, так что дисковод остается свободен. В следующем примере с помощью tomsrtbt дискета форматируется (в формате Linux EXT2), монтируется в систему, далее монтируется раздел жесткого диска, и на дискету копируется несколько файлов:

```
# fdformat /dev/fd0H1440
# mke2fs /dev/fd0
# mkdir /mnt/image
# mkdir /mnt/floppy
# mount -t ext2 /dev/hda4/mnt/image
# mount -t ext2 /dev/fd0/mnt/floppy
# cd /mnt/floppy
# cp /mnt/image/etc/*.conf .
```

ООО "НТК ИНФОТЕХ"

```
# cp /mnt/image/etc/conf.modules .
# cp /mnt/image/var/named/* .
# cd /mnt
# umount floppy
# umount image
```

А теперь пример сохранения файлов с помощью tomsrtbt из системы Windows, которая больше не грузится (потребуются дискеты, отформатированные под DOS). Можно также отредактировать файл autoexec.bat:

```
# mkdir /mnt/win
# mkdir /mnt/floppy
# mount -t vfat /dev/hda1/mnt/win
# mount -t vfat /dev/fd0/mnt/floppy
# vi /mnt/win/autoexec.bat
# cd /mnt/floppy
# cp /mnt/win/autoexec.bat .
# cp /mnt/win/config.sys .
# cp "/mnt/win/program files/netscape/bookmarks.htm" .
# cd /mnt
# umount floppy
# umount win
```

Для переноса информации между Linux и Windows может использоваться набор mtools, который в среде Linux оперирует дискетами и информацией формата DOS/Windows. В частности, его программы позволяют преобразовывать LF-окончания строк в Linux в типичные для Windows CR/LF. В следующем примере форматируется дискета для DOS, на нее копируются файлы с Linux-окончаниями и файл с Windows-окончаниями:

```
$ mformat a:
$ cd /etc
$ mcopy *conf* a:
$ cd /home/author/docs
$ mcopy -t myrecipes.txt a:
```

Обратный процесс (копирование файла из DOS в Linux с преобразованием концов строк) выглядит так:

```
$ cd /home/author/docs
$ mcopy -t a:myoldrecipes.txt
```

Обратите внимание на значок $, показывающий, что форматирование можно было выполнять обычному пользователю (не root). Чтобы позволить всем пользователям обращаться к дискетам посредством mtools, следует сначала включить их в группу floppy и задать группе разрешение read-write-execute на /dev/fd0.

Надеемся, что данный материал окажется также интересен тем пользователям Windows, кто хотел бы "попробовать" Linux, но не решается пожертвовать местом на жестком диске для начального знакомства.

Литература

Gene Wilburn. Disaster recovery tips. — The Computer Paper, June 2001.

Подготовил В.Ермоленко.

СЛОВО — НЕ ВОРОБЕЙ

Одна из возможностей, заложенных фирмой Microsoft в текстовый редактор Word97 и последующие версии, может крупно подвести тех, кто невнимательно им пользуется. Речь идет о сохранении в файле документа предыдущих вариантов в процессе внесения исправлений (Revision Marks) несколькими пользователями. По умолчанию предыдущие варианты не видны при просмотре, так что не всем даже известно об их существовании.

Фирма Alcatel, производитель модемов для ADSL (службы скоростной передачи данных по выделенным телефонным линиям), распространила заявление по поводу обнаруженной в одной из моделей "дыры" в безопасности, что, в принципе, позволяет хакерам дистанционно нарушать соединение и отключать такой модем. Однако в разосланном документе Word остались комментарии и исправления, внесенные сотрудниками фирмы в процессе обсуждения этого заявления [1]. Приблизительный перевод фрагментов данного текста очень поучителен:

"...Alcatel ~~инициировала сертификацию программы firewall, которая даст пользователям максимальную степень защиты.~~

~~(Когда и где появится такая программа? Центр сертификации сообщил, что они не верят, будто firewall — это ответ. Что вы делаете для того, чтобы предоставить настоящее исправление ошибки?)~~

Поэтому Alcatel настоятельно рекомендует постоянное использование firewall, в особенности для постоянно включенных кабельных и DSL-модемов.

Чтобы увеличить безопасность своей продукции, ~~в частности для более информационно чувствительного рынка малых и домашних офисов,~~ Alcatel уже ранее вводил дополнительные меры безопасности для предотвращения прямого вмешательства в модемы удаленными пользователями. Эта аппаратная защита — firmware — имеется в некоторых моделях. ~~Firmware включается по умолчанию при выпуске этих модемов.~~

~~(Этот абзац беспокоит. Можно понять так, что мы оказываем предпочтение одному сегменту наших клиентов — SOHO. Почему мы не даем этот уровень уверенности всем нашим клиентам? Почему мы не включаем firewall по умолчанию для всех наших клиентов?).~~

Alcatel поставляет модемы со включенной защитой firmware."

Попасть в подобную ситуацию легко могут те пользователи Word, кто вместо использования чистого бланка предпочитает открыть какой-нибудь ранее созданный документ с нужным логотипом, "удалить" ненужное содержимое и написать новый текст. Убедитесь, что удаленного текста в файле действительно нет, перед тем как отправить его адресату. Для показа изменений в Word97, выберите Сервис | Исправления | Выделить исправления | Отображать исправления на экране (Tools | Track Changes | Highlight Changes | Highlight Changes on Screen). В Word2000 этому соответствует Tools | Track Changes | Changes with Highlighting, в Word2002 — Tools | Track Changes). Кроме Revision Marks, подобную же опасность представляют режимы Fast Save и Save Versions.

Литература

1. http://morons.org/articles/1/188.

Подготовил В.Ермоленко

Рисунок О.Попова

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

Передача параметров при вызове ассемблерных процедур из программы на Бейсике

Введение

После того как программа на Бейсике написана, часто выясняется, что она работает непозволительно медленно. Есть несколько способов решения этой проблемы. Можно полностью переписать программу на ассемблере, добившись при этом максимального быстродействия — но на это придется затратить много времени и сил. Можно использовать компилятор Бейсика — это не потребует сколько-нибудь существенных усилий, но программа не будет столь же быстрой, как написанная на ассемблере, к тому же, могут возникнуть проблемы, вызванные недостаточным объемом памяти при компиляции или при попытке запуска откомпилированной программы. Можно, наконец, переписать на ассемблере только те участки программы, на выполнение которых тратится большая часть времени. При этом быстродействие будет лишь немногим ниже чем при полном переписывании программы на ассемблере, а объем работы будет значительно меньше. Именно этот способ оптимизации в ряде случаев оказывается наиболее предпочтительным.

При использовании ассемблерных процедур встает вопрос передачи параметров. В стандартном Бейсике средства для этого весьма скромные — предусмотрено лишь, что вызываемая процедура может вернуть в регистровой паре BC одно значение, представляющее собой целое число в диапазоне 0...65535. Например, при выполнении команды

```
....
1000 LET A = USR 40000
....
```

будет вызвана ассемблерная процедура, располагающаяся по адресу 40000, после чего переменной A будет присвоено значение регистровой пары BC.

Можно выделить для передачи параметров определенную область памяти, записывать туда перед вызовом процедуры значения передаваемых переменных с помощью оператора POKE, а после вызова — считывать возвращаемые значения с помощью оператора PEEK. Пусть, скажем, требуется передать процедуре масштабирования координат, находящейся по адресу 40004, переменные X и Y (целые, в диапазоне 0...65535) через область памяти 40000...40003 (с адреса 40000 располагается значение X в формате младший байт/старший байт; с 40002 — значение Y в таком же формате). Процедура изменит эти значения определенным образом, оставив результат в тех же ячейках памяти. После этого надо прочесть возвращенные значения и присвоить их переменным X1 и Y1.

```
.....
1000 POKE 40000,X-INT(X/256)*256: POKE 40001,INT(X/256)
1010 POKE 40002,Y-INT(Y/256)*256: POKE 40003,INT(Y/256)
1020 RANDOMIZE USR 40004
1030 LET X1=PEEK(40000)+PEEK(40001)*256
1040 LET Y1=PEEK(40002)+PEEK(40003)*256
.....
```

Как видим, при таком способе передачи параметров увеличивается объем Бейсик-программы, тратится время на передачу значений (иногда довольно значительное — например, при передаче массивов), приходится запоминать, по какому адресу какая переменная находится, да и, наконец, просто может не хватить памяти.

Более перспективным способом представляется тот, когда ассемблерная программа непосредственно работает с переменными Бейсика. Рассмотрим этот способ более подробно.

Переменные Бейсика

Рассмотрим сначала, какие бывают в Бейсике переменные, где и в каком формате они хранятся.

Переменные Бейсик-программы находятся в области памяти, начало которой адресуется системной переменной VARS, расположенной по адресу #5C4B=23627. Переменные идут строго друг за другом, а после последней из них находится байт #80 — признак конца области переменных.

Хотя в именах переменных допускаются и прописные, и строчные латинские буквы, а также пробелы и цифры (цифра не может быть первым символом), в области хранения переменных имена преобразуются к строчным буквам, а пробелы не сохраняются (таким образом, для интерпретатора переменные Size X и sizex — одно и то же), знак "$" для символьных переменных также не сохранится.

Ниже перечислены различные типы переменных и приведен формат их хранения (данные о формате, приведенные в [1], прямо скажу, не отличаются полнотой и точностью, поэтому многое пришлось изучать в ходе практических экспериментов).

Числовая переменная с именем из одной буквы:
- имя (1 байт);
- значение (5 байтов).

Числовая переменная с именем из нескольких символов:
- имя (код первого символа увеличен на #40, последнего — на #80);
- значение (5 байтов).

Переменная цикла for-next:
- имя (1 байт), код символа увеличен на #80;
- значение (5 байтов);
- ограничение (5 байтов);
- приращение (5 байтов);
- номер строки, в которой расположен соответствующий оператор FOR (2 байтв);
- номер следующего после FOR оператора в этой строке (1 байт).

(*Примечание*: переменная с именем из одной буквы, определенная вначале как обычная, впоследствии, если она используется в качестве переменной цикла for-next, изменяет свой формат хранения.)

Числовой массив:
- имя (1 байт), код символа увеличен на #20;
- длина массива в байтах (2 байта);
- количество размерностей (1 байт);
- первая размерность (2 байта);
- .....
- последняя размерность (2 байта);
- элементы (5 байтов каждый), хранящиеся в следующем порядке — сначала возрастает последний индекс, затем

предпоследний и т.д. Например: (1,1), (1,2), (1,3), (2,1), (2,2), ...

Символьная переменная:
- имя (1 байт), код символа уменьшен на #20;
- количество символов (2 байта);
- содержимое.

Символьный массив:
- имя (1 байт), код символа увеличен на #60;
- длина массива в байтах (2 байта);
- количество размерностей (1 байт);
- первая размерность (2 байта);
- .....
- последняя размерность (2 байта);
- элементы (1 байт каждый). Порядок хранения такой же, как у числовых массивов.

Как видим, по значению первого байта имени переменной сразу же можно определить ее тип (см табл.):

Диапазон значений первого байта	Тип переменной
#41...#5A	Символьная
#61...#7A	Число (имя — одна буква)
#81...#9A	Числовой массив
#A1...#BA	Число (имя — несколько символов)
#C1...#DA	Символьный массив
#E1...#FA	Переменная цикла

Приведу некоторые сведения о пятибайтовой форме представления чисел, используемой при хранении переменных. Целые числа в диапазоне 0...65535 представляются следующим образом:

0	0	Младший байт числа	Старший байт числа	0

А целые числа в диапазоне -65535...-1 — так:

0	#FF	Младший байт числа	Старший байт числа	0
		(число представляется в дополнительном коде)		

Вещественные числа хранятся в более сложной форме. Сведения о формате их хранения вы можете почерпнуть, например, из [3]. Здесь я их не привожу, так как работа с такими числами на ассемблере оказывается необходимой лишь в очень редких случаях. Часто можно преобразовать решаемую задачу так, чтобы работать только с целыми числами, что намного ускоряет вычисления.

Попробуем теперь, вооружившись полученными знаниями, "расшифровать" содержимое реальной области переменных, дамп которой приведен на рис.1.

Рис. 1

Первый байт области переменных — #98. Он принадлежит диапазону #81...#9A, следовательно, первая переменная — числовой массив. Как мы помним, для числового массива код символа имени хранится увеличенным на #20. Определим имя — #98-#20=#78, это код символа "x". В следующих двух байтах находится длина массива (не считая байта имени и двух байтов самой длины). Прочитав ее (#23=35) и перейдя на соответствующее число байтов вперед, мы можем сразу перейти к следующей переменной. Но пока делать этого не будем, а рассмотрим структуру массива. В следующем байте хранится число 2 — значит, массив двумерный. По первому измерению его размер равен 2, по второму — 3 (а всего, значит, в нем 6 элементов). Дальше располагаются сами элементы. X(1,1)=4, X(1,2)=0, последующие элементы также равны нулю.

Первый байт следующей переменной — #C1, следовательно, это символьный массив с именем A$... Ну и так далее. Надеюсь, с оставшейся частью этой области переменных вы сможете разобраться самостоятельно.

Процедура определения адреса переменной

Для определения местоположения переменной Бейсика удобно воспользоваться нижеприведенной процедурой. Вот ее входные и выходные параметры.

Вход: DE указывает на имя переменной, преобразованное в соответствии с ее типом (т.е. имя должно быть именно в таком виде, как оно хранится в области переменных) и заканчивающееся байтом 0.

Выход: если переменная найдена — флаг Z сброшен, HL указывает в области переменных на первый байт после имени найденной переменной, DE указывает на следующий байт после 0 (а значит, если хранить имена искомых переменных друг за другом, то, вначале загрузив в DE адрес первого имени и затем последовательно вызывая процедуру, мы будем последовательно получать адреса этих переменных). Если переменная не найдена, флаг Z установлен, DE остается без изменений.

Процедура при своей работе изменяет содержимое регистра A.

```
VARS        EQU   #5C4B

FIND_VAR    LD    HL,(VARS)

            PUSH  BC

FIND_M0     PUSH  DE
            PUSH  HL

;Сравнение имени текущей переменной с именем искомой:

FIND_M1     LD    A,(DE)
            INC   DE

            AND   A
            JR    Z,FIND_YES  ;Имена совпали.

            CP    (HL)
            INC   HL
            JR    Z,FIND_M1

;Имена не совпали — пропускаем текущую переменную:

            POP   HL
            POP   DE
            LD    A,(HL)
            INC   HL

            CP    #80         ;Дошли до конца области переменных?
            JR    Z,FIND_NO   ;Если да — выходим, флаг Z
                              ;установлен.

            CP    #60
            JR    C,SKIP_M    ;Пропускаем символьную переменную

            CP    #80
FIND_M4     LD    BC,5
            JR    C,SKIP_BC   ;Пропускаем числовую переменную
                              ;с именем из одной буквы.
            CP    #A0
```

```
          JR    C,SKIP_M    ;Пропускаем числовой массив.

          CP    #C0
          JR    NC,FIND_M3

;Пропускаем числовую переменную с именем из нескольких
;символов:

FIND_M2   LD    A,(HL)
          INC   HL
          RLA
          JR    NC,FIND_M2
          JR    FIND_M4

FIND_M3   CP    #E0
          LD    C,18        ;B=0!
          JR    NC,SKIP_BC  ;Пропускаем переменную цикла.

;Пропуск символьной переменной или массива:

SKIP_M    LD    C,(HL)
          INC   HL
          LD    B,(HL)
          INC   HL

SKIP_BC   ADD   HL,BC
          JR    FIND_M0

FIND_YES  POP   BC          ;Снимаем два теперь не нужных
                            ;числа со стека.
          POP   BC
          INC   A           ;A станет равным 1, флаг Z
                            ;сбросится.

FIND_NO   POP   BC
          RET               ;Выход
```

Интерпретатор Бейсика производит подобный поиск при каждом обращении к какой-либо переменной — вот вам и одна из причин низкой скорости работы Бейсик-программ.

Пример оптимизации

Попробуем оптимизировать по быстродействию какую-либо простую Бейсик-программу — например, приведенную ниже.

```
10 LET N=0: INPUT A$
20 IF A$="" THEN GO TO 60
30 FOR I=1 TO LEN (A$)
40 IF A$(I)="X" THEN LET N=N+1
50 NEXT I
60 PRINT N
```

Эта программа запрашивает у пользователя строку и подсчитывает, сколько раз в ней встретится символ "X". Критичным по быстродействию участком, очевидно, является цикл в строках 30...50. Перепишем его на ассемблере. Пусть он размещается, скажем, с адреса 40000:

```
          ORG   40000

          LD    DE,NAME_A   ;Ищем переменную A$.
          CALL  FIND_VAR

;Теперь HL указывает на 2-байтное значение, определяющее
;длину строки A$.

          LD    E,(HL)
          INC   HL
          LD    D,(HL)
          INC   HL

;DE - длина строки, HL указывает на первый символ строки.

          LD    BC,0        ;Счетчик символов "X".

M1        LD    A,(HL)
          CP    "X"
          JR    NZ,M2
          INC   BC

M2        INC   HL
          DEC   DE
          LD    A,D
          OR    E
          JR    NZ,M1

;BC - количество символов "X" в строке.

          LD    DE,NAME_N   ;Ищем переменную N.
          CALL  FIND_VAR

;Теперь HL указывает на значение переменной N. Помещаем туда
;содержимое BC, представленное в 5-байтовой форме:

          LD    (HL),0
          INC   HL
          LD    (HL),0
          INC   HL
          LD    (HL),C
          INC   HL
          LD    (HL),B
          INC   HL
          LD    (HL),0

          RET               ;Выход.

NAME_A    DB    "a"~#20     ;Имя переменной A$.
          DB    0
NAME_N    DB    "n"         ;Имя переменной N.
          DB    0

<далее следует текст процедуры FIND_VAR>
```

Результат компиляции запишем в файл "file1". После этого остается лишь переписать исходную программу:

```
5  CLEAR 39999: RANDOMIZE USR 15619: REM: LOAD "file1" CODE
10 LET N=0: INPUT A$
20 IF A$="" THEN GO TO 60
30 RANDOMIZE USR 40000
60 PRINT N
```

Напомню условия, которым должна удовлетворять вызываемая из Бейсика ассемблерная процедура:

- маскируемые прерывания при работе вашей процедуры могут запрещаться, но перед выходом должны быть разрешены. Учтите, что при запрещенных прерываниях не увеличивается значение системного таймера — это может быть важным, если вы используете его, скажем, для подсчета времени выполнения программы;
- значение регистровой пары HL' при выходе из процедуры должно быть таким же, как и при входе — равным #2758. Так что, если в процедуре вы использовали HL' для каких-то своих целей, перед выходом не забудьте вставить команды LD HL,#2758: EXX;
- значение регистровой пары IY не должно меняться при включенных прерываниях, так как оно используется системной процедурой обработки прерываний. Поэтому, если вам нужно использовать IY, вначале запретите прерывания, а перед их разрешением не забудьте восстановить содержимое IY (там должно быть число #5C3A).

Несколько полезных советов

Значения числовых переменных лучше преобразовать для последующей обработки из 5-байтной формы в одно- или двухбайтную, в зависимости от диапазона — так их можно будет быстрее обрабатывать. Если нужно работать с массивами, и в памяти достаточно свободного места, стоит скопировать их по адресам, кратным 256 — это позволит ускорить

доступ к элементам такого массива (подробнее о том, как быстро работать с массивами, вы можете прочитать в моей статье "Оптимизация на примере intro "Start" [4]).

Создание новых переменных

Возможна ситуация, когда в переписываемом фрагменте программы выполняется присвоение значения ранее еще не использовавшейся переменной. В этом случае информация о ней отсутствует в области переменных, и, следовательно, надо ее туда добавить. Для этого необходимо выделить в конце области переменных (между концом последней переменной и байтом #80) область памяти необходимой длины и занести в нее имя и значение новой переменной.

Выделение области памяти выполняется с помощью процедуры MAKE_ROOM, расположенной в ПЗУ по адресу #1655. При ее вызове в регистровой паре HL должен находиться адрес, с которого нужно выделить место, а в BC — длина выделяемой области.

Адрес выделяемой области памяти нетрудно определить, если учесть, что непосредственно за областью переменных располагается область редактируемых строк программы, адрес начала которой хранится в системной переменной E_LINE. Следовательно, для получения искомого адреса достаточно взять значение E_LINE и уменьшить его на 1.

Рассмотрим пример. Пусть имеется такая программа:

```
10 INPUT A: INPUT B
20 LET C=A+B
30 PRINT C
```

и нам нужно переписать на ассемблере строку 20. Известно, что переменные A, B и C — целые, и их значения принадлежат диапазону 0...65535. В этом случае ассемблерный текст будет таким:

```
              ORG     40000

MAKE_ROOM     EQU     #1655
E_LINE        EQU     23641

              LD      DE,NAME_A
              CALL    FIND_VAR

;Теперь HL указывает на значение переменной A,
;DE указывает на NAME_B.

              INC     HL
              INC     HL
              LD      C,(HL)
              INC     HL
              LD      B,(HL)

;BC - значение переменной A.

              CALL    FIND_VAR

              INC     HL
              INC     HL
              LD      E,(HL)
              INC     HL
              LD      D,(HL)

;DE - значение переменной B.

              EX      DE,HL
              ADD     HL,BC           ;Вычислили C.
              PUSH    HL              ;Запомнили в стеке.

;Выделяем место для переменной C.

              LD      HL,(E_LINE)
              DEC     HL              ;Где выделяем.
              PUSH    HL
              LD      BC,6            ;Сколько выделяем.
              CALL    MAKE_ROOM
              POP     HL

              LD      (HL),"c"        ;Имя переменной.
              INC     HL

              POP     DE              ;Сняли со стека
              LD      (HL),0          ;значение переменной
              INC     HL              ;и записываем его
              LD      (HL),0          ;в 5-байтной
              INC     HL              ;форме.
              LD      (HL),E
              INC     HL
              LD      (HL),D
              INC     HL
              LD      (HL),0

              RET                     ;Выход.

NAME_A        DB      "a"             ;Имя переменной A.
              DB      0
NAME_B        DB      "b"             ;Имя переменной B.
              DB      0

<далее следует текст процедуры FIND_VAR>
```

Литература

1. Бейсик ZX Spectrum. — М.: VA PRINT, 1993.
2. Секреты ПЗУ. — ZX-Ревю, 1991, №10.
3. Персональный компьютер ZX Spectrum. Программирование в машинных кодах и на языке ассемблера. — М.: Инфорком, 1993.
4. И.Рощин. Оптимизация на примере intro "Start". — Радиомир.Ваш компьютер, 2001, №№7-10.

КОМПЬЮТЕРНЫЕ ХРОНИКИ

NUO'tm virtual compo

Подведены итоги еще одного виртуального конкурса на платформе "ZX-Spectrum" — Nuotrauka'third millennium. Двадцать музыкальных модулей, тринадцать картинок и четыре ASCII-работы (для тех, кто не знает, ASCII — это когда картинку создают в текстовом режиме с помощью различных символов) были оценены как всеми желающими, так и специальным жюри. Соответственно, имеются две версии результатов. Впрочем, лично для меня разница между ними оказалась невелика: по результатам общего голосования мой музыкальный модуль занял 15-е место из 20, а по оценке элитных сценеров — 11 из 14. :)

Итак, назовем тройку лидеров в каждом compo по результатам общего голосования (полные результаты, так же как и сами работы, ищите на спектрумовских сайтах).

Музыка: 1 — "first trip" (sergant//fbn), 2 — "it's timing" (vox//alc), 3 — "imagination" (nik-o//tl).

Графика: 1 — "u r dun" (gas13), 2 — "conan" (drawer//entire), 3 — "kin ship (key la!)" (survivor).

ASCII: 1 — "zilogXXI" (falcon//kingdom dreams), 2 — "ctl" (kristoph//ctl i.a.), 3 — "am i god?" (screamer//jmb).

По словам организаторов, подробный отчет о NUO'tm будет опубликован во втором номере электронного журнала "Scream".

Хроники вел И.Рощин.

С.АФАНАСЬЕВ,
г.Ульяновск.
E-mail: 21745asv@mail.ru

Глюкодром

Где эта сволочь?
Михалыч, к/ф "Особенности национальной охоты"

Вот сижу я за клавиатурой и думаю, как же все-таки плохо, когда программы пишут люди, которым "строго параллельно" удобство пользователя! Мышь на Спеке уже как минимум четыре года, а до сих пор мало кто научился корректно определять ее присутствие. Существуют компьютеры, на которых "при наличии отсутствия факта присутствия данного девайса" из портов, соответствующих ему, читается всякий бред, и если опрос мыши не отключить, то юзер получит абсолютно нерабочую программу, а автор — массу абсолютно нецензурных выражений в свой адрес (от этого юзера). Вот и сегодня я попытался запустить некую программу под названием "BRED_512". Не подумайте чего плохого, таким загадочным именем автор окрестил редактор уровней для "ЧЕРНОГО ВОРОНА". Соответственно, количество нецензурных выражений в адрес этого BRED'а (:-) превысило все разумные пределы. А проблема заключалась в вышеописанном глюке, которого, кстати, избежать не просто, а очень просто. Это делается примерно так:

```
        LD    B,100   ; можно и меньше
LOOP    PUSH  BC
        LD    BC,#FADF
        IN    A,(C)
        CPL
        AND   %00000111
        POP   BC
        JR    NZ,M_OFF
        DJNZ  LOOP
        JR    CONTINUE
        ...
M_OFF               ; здесь процедура, отключающая
                    ; опрос мыши
CONTINUE...
```

Итого 17 байтов — и проблемы как не бывало! Нечто подобное использовалось в оболочке электронной газеты "DIGITAL MADNESS", которая прекратила свое существование незадолго до окончания работы над ее вторым номером :-(. Ставится данная "мулька" в самом начале программы, отрабатывает перед запуском основного блока. А еще лучше проставить ее уже в LOADER, тогда процедура M_OFF должна дать понять загрузчику, что после распаковки нужно отключить опрос мыши.

Теперь кратко объясню суть вышеизложенной программы. Все просто — она 100 раз опрашивает порт кнопок мыши, и если хотя бы один раз считается число, отличное от #FF, это свидетельствует о нестабильности портов мыши, и опрос ее будет жестоко (безжалостно, варварски, ну прямо-таки зверски) отключен. Предполагается, что "юзверь" не будет давить на кнопки во время загрузки (ну а если будет — в награду получит отключенную мышь — нечего на кнопки давить, когда не просят). Кстати, число 100 (#64, или %01100100) взято (как вы думаете, откуда?), естественно, с потолка. При желании, число повторов цикла можно установить любым, хоть до 10^{64} (если терпения хватит организовать такое количество циклов). Один раз опрашивать не рекомендуется — может получиться так, что как раз в этот момент из порта #FF и будут "проистекать глюки". Если у кого-то не хватает фантазии представить, как выглядит данный "глюк", то могу кратко его описать — курсор как безумный скачет по всему экрану, включая все попадающие под него иконки, и абсолютно не реагирует на попытки им управлять.

Кто-то может заявить — "повесил бы себе мышу, и решил бы все проблемы...". Отвечу — дело не в том, есть у меня мышь или нет, я-то ее во всех имеющихся патологических программах уже "зарезал". Дело в том, что я не единственный с таким извращенным компьютером и без мыши. Наверняка есть люди с аналогичными проблемами, а сломать чужую программу они не могут в силу своего благородного воспитания (;)). Добавить несколько лишних байтов в программу не так уж сложно, зачем же портить нервы тем немногим пользователям, которые еще остались (я надеюсь...) на Spectrum.

Приведенная выше процедура испытана на всех доступных мне компьютерах с мышью и без нее, и имеет 100% гарантию работоспособности. Может, это и не шедевр, но она работает, причем надежно.

Disk error.
Trk 0 sec 9.
Retry, Abort, Ignore?
(c) ПЗУ ТР-ДОС

Речь здесь пойдет о таком распространенном ныне явлении как турболоадер (в просторечии ТРУБОГРУЗ). Но сначала небольшое лирическое отступление.

Видел пи кто-нибудь из вас, уважаемые читатели, компьютер с одним дисководом? Думаю, что многие. А кто из вас видел, чтобы этот единственный дисковод был подключен как диск "B" без всякой возможности включить его как "A"? Можете меня поздравить, с недавнего времени я являюсь счастливым обладателем такого вот чуда. Причина такого крутого поворота событий пока не выяснена, но я надеюсь, скоро будет найдена и пофиксена.

Так вот, пару дней назад притащил я себе пару дисков с тремя версиями игры "Страна дураков" (ooops! Sorry! "Поле чудес", of coz!) и решил, понимаешь, слегка поиграть. Щщщщаааззззззз! Размечтался! Все они оказались снабжены вышеупомянутыми турбозагрузчиками, которые пытались выполнить загрузку с диска "A", не обращая внимания на то, что такого диска на моем компьютере нет ВООБЩЕ, со всеми вытекающими отсюда последствиями. Будучи до глубины души оскорблен таким хамским поведением безобидных на вид игрушек, я решил провести вскрытие ("Доктор, что со мной? — Не знаю, вскрытие покажет...") с целью выяснения причин данного безобразия. Приготовив STS 6.2, я приступил к делу...

На этом хочу прервать это несколько затянувшееся вступление. Скажу только, что операция прошла успешно, пациент умер :). Причина была выяснена, и далее я постараюсь объяснить, как не допустить подобных "глюков" при написании своих программ. Я не собираюсь подробно описывать команды ВГ93 и способы их применения. Просто хочу кое-что прояснить.

Для начала — кусок стандартного трубогруза:

```
        LD    D,TRACK ;номер дороги
        ... ... ...

        LD    A,#3C   ;<--- !!!!

        BIT   0,D
        JR    Z,L1

        LD    A,#2C   ;<--- !!!!

L1      LD    C,#FF
        CALL  OUTC
        ... ... ...
OUTC    LD    IX,#2A53
        PUSH  IX
        JP    #3D2F
```

Процедура OUTC выполняет запись аккумулятора в произвольный порт контроллера. Номер порта задан в регистре C. Соответственно, данный фрагмент выполняет запись в порт #FF одного из двух чисел — #2C или #3C, в зависимости от номера дорожки, которую надо прочитать.

Теперь поясню назначение порта #FF:

- bit 6 — плотность записи (как правило, игнорируется контроллером);
- bit 4 — сторона диска (1 — нижняя, 0 — верхняя);
- bit 3 — STOP — остановка диска;
- bit 2 — BREAK — немедленное прерывание текущей операции и остановка мотора;
- bit 0, 1 — номер дисковода.

А сейчас я побитно распишу те самые числа, которые в данном примере пишутся в порт #FF:

```
#2C=%00101100
  ^^
#3C=%00111100
  ^^
```

Как говорится, комментарии излишни, но все-таки — биты 0 и 1 порта #FF равны 00, соответственно, будет выбран дисковод A независимо от того, с какого дисковода стартовала программа.

Как же бороться с таким злобным "глюком"? Да очень просто. Есть такая системная переменная, проживающая по адресу #5D19 (23833, если по-русски). В ней обычно лежит номер текущего дисковода (если только ваша программа еще не испортила этот адрес). А вот каким концом ее приспособить к нашему турбогрузу, я думаю, вы и сами поймете.

P.S. Недавно мне на глаза попался электронный журнал ZX-HARD, в котором метод, описанный в первой части данной статьи, яростно клеймился как абсолютно неправильный, на том основании, что обычно, если мыши нет, то из всех портов, которые ей соответствуют, читается #FF. То есть при использовании данного метода будет определено, что мышь подключена. Поясняю — этим способом действительно нельзя определить ФИЗИЧЕСКОЕ наличие мыши. Он (способ) всего-навсего проверяет, не приведет ли опрос мыши к летальному исходу (для программы, естественно)... В заключение хочу принести извинения PAWEL'y/REALSOFT за неумышленное использование в своем nickname слова REAL (статья [1]). К REALSOFT я отношения не имею.

Литература

1. С.Афанасьев. В поисках вечной жизни. — Радиолюбитель. Ваш компьютер. 2001, №4, С.38.

Автоматическая генерация имен и создание файла

И.ГОДУНОВ,
г. Минск.

Ваша долгожданная программа, наконец, начала оформляться. Результаты ее работы вам приятно наблюдать. Осталось нанести последний штрих — сохранить результаты работы программы. Но вам нужно не просто выполнить команду SAVE для диска, а создать автоматизированную систему записи на диск множества файлов с разными именами, причем неважно: за один раз или за несколько. Важно, чтобы программа сама определяла, какие имена повторяются, а какие нет. Кроме того, должна производиться проверка свободного пространства на диске, и количество файлов не должно превышать 128. Приведенная ниже программа выполняет как раз такие функции. На диске создается серия файлов типа: PROG-001 <C>.

На диске будет создаваться серия имен с индексами от 1 до 128, но если позволит каталог, то и 999. Пропущенные номера (например, после стирания) будут восполняться.

```
NOM_I   DEFM "PROG-"      ; маска имени
NOM_N   DEFM "0"          ; числовой индекс
NOM_N1  DEFM "0"
NOM_N2  DEFM "1"
NOM_N3  DEFM "C       "   ; тип файла и 7 пробелов
TSO     DEFB 0            ; переменная количества совпадает
BUF     EQU  nnnnn        ; начальный адрес буфера 0 трека
ST_AD   EQU  nnnnn        ; начальный адрес блока памяти,
                          ; подлежащий сохранению
EKV     LD   DE,256       ; подпрограмма вычисления
                          ; количества секторов по длине
        LD   BC,0         ; длина в HL перед запуском подпрограммы
VVO     XOR  A
        SBC  HL,DE
        INC  BC
        JR   Z,IEXIT2
        JR   C,IEXIT
        JR   VVO
IEXIT   LD   A,C          ; A - количество секторов
        LD   B,0          ; сектор занят не полностью
        RET
IEXIT2  LD   A,C
        LD   B,1          ; сектор занят полностью
        RET
        ; ЗАПУСК ПРОГРАММЫ
RUN     LD   A,6          ; сообщение PRESS ENTER FOR SAVE
        LD   B,0
        LD   C,0
        CALL ZAP_E        ; подпрограмма печати сообщений на
                          ; экране
        LD   A,7
        LD   B,0
        LD   C,11
        CALL ZAP_E
KV5     CALL INKEY        ; опрос клавиатуры
        CP   13           ; контроль нажатия ENTER
        JR   Z,SAVE2
        JR   KV5
        LD   A,(D1)       ; переменная <D1> хранит номер
                          ; текущего дисковода
        LD   (23833),A
SAVE2   LD   C,#18        ; настройка на дискету
        CALL 15635
        LD   HL,BUF       ; начальный адрес буфера, куда будут
                          ; считываться 9 секторов
        LD   D,0          ; 0 трек
        LD   E,0          ; 0 сектор
        LD   B,9          ; 9 секторов
        LD   C,5
        CALL 15635
        LD   HL,BUF+2048  ; начальный адрес 9 сектора в буфере
        LD   DE,228
        ADD  HL,DE
        LD   A,(HL)
        CP   128          ; контроль количества файлов
        JP   NZ,SAVE3     ; если меньше, то переход к
                          ; следующ. проверке
        LD   A,5          ; сообщение: Files>128. Insert
                          ; another disk
        LD   B,0
        LD   C,0
        CALL ZAP_E
        LD   A,6          ; сообщение: Press ENTER
        LD   B,1
        LD   C,0
        CALL ZAP_E
KV3     CALL INKEY
        CP   13
```

```
        JP    Z,SAVE2
        JP    KV3
SAVE3   LD    HL,BUF+2048 ; начальный адрес 9 сектора в буфере
        LD    DE,229      ; контроль количества свободного
                          ; места на диске
        ADD   HL,DE
        LD    E,(HL)
        INC   HL
        LD    D,(HL)
        PUSH  DE
        LD    HL,(T_AD)   ; текущий адрес
        LD    DE,ST_AD    ; начальный адрес сохраняемого файла
        XOR   A
        SBC   HL,DE
        CALL  EKV
        POP   DE
        LD    C,A
        LD    A,D
        CP    0
        JP    NZ,SAVE4    ; переход, если места достаточно
        LD    A,E
        CP    C
        JP    NC,SAVE4
        LD    A,9         ; сообщение: No space on disk!
        LD    B,0
        LD    C,0
        CALL  ZAP_E
        LD    A,10        ; сообщение: Insert another disk
        LD    B,1
        LD    C,0
        CALL  ZAP_E
        LD    A,6         ; сообщение: Press ENTER
        LD    B,1
        LD    C,20
        CALL  ZAP_E
KV4     CALL  INKEY
        CP    13
        JP    Z,SAVE2
        JP    KV4
SAVE4   LD    HL,BUF      ; *поиск неповторяющегося имени файла*
        LD    (T_BUF),HL
NASR    LD    HL,(T_BUF)  ; начало сравнения
        LD    A,(HL)
        CP    0
        JP    Z,KCAT      ; контроль метки конца каталога
        CP    80          ; <P>
        JP    NZ,NEELK    ; переход к следующему элементу
                          ; каталога
        INC   HL
        LD    A,(HL)
        CP    82          ; <R>
        JP    NZ,NEELK
        INC   HL
        LD    A,(HL)
        CP    79          ; <O>
        JP    NZ,NEELK
        INC   HL
        LD    A,(HL)
        CP    71          ; <G>
        JP    NZ,NEELK
        INC   HL
        LD    A,(HL)
        CP    45          ; <->
        JP    NZ,NEELK
        INC   HL
        LD    B,(HL)
        LD    A,(NOM_N)
        CP    B
        CALL  Z,S_VPD     ; переход при совпадении
        INC   HL
        LD    B,(HL)
        LD    A,(NOM_N1)
        CP    B
        CALL  Z,S_VPD
        INC   HL
        LD    B,(HL)
        LD    A,(NOM_N2)
        CP    B
        CALL  Z,S_VPD
        LD    HL,TSO      ; если 3 цифры номера совпали, то
                          ; сменить номер
        LD    A,3
        CP    (HL)
        LD    (HL),0      ; восстановление переменной TSO
        JP    NZ,NEELK
        LD    A,(NOM_N2)  ; смена номера (увеличение)
        INC   A
        LD    (NOM_N2),A
        CP    58
        JP    NZ,SAVE4
        LD    A,48
        LD    (NOM_N2),A
        LD    A,(NOM_N1)
        INC   A
        LD    (NOM_N1),A
        CP    58
        JP    NZ,SAVE4
        LD    A,48
        LD    (NOM_N1),A
        LD    A,(NOM_N)
        INC   A
        LD    (NOM_N),A
        JP    SAVE4       ; начать проверку заново
S_VPD   PUSH  HL          ; при совпадении - счетчик увеличить
        LD    HL,TSO
        INC   (HL)
        POP   HL
        RET
NEELK   LD    HL,(T_BUF)  ; установка на следующий элемент
                          ; каталога
        LD    DE,16
        ADD   HL,DE
        LD    (T_BUF),HL  ; проверка на конец каталога
        LD    DE,BUF+1192
        XOR   A
        SBC   HL,DE
        JP    Z,KCAT
        JP    NASR
KCAT    LD    HL,NOM_I    ; просмотр каталога закончен
        LD    C,#13       ; копирование элемента каталога
                          ; в дескриптор файла
        CALL  15635
        LD    LH,(T_AD)
        LD    DE,ST_AD
        XOR   A
        SBC   HL,DE
        EX    DE,HL
        LD    HL,ST_AD
        LD    C,#0B       ; создание файла типа <C>
        CALL  15635
        LD    A,9
        LD    (23814),A
        LD    C,#0A       ; поиск элемента каталога с целью
                          ; выяснения номера
        CALL  15635
        PUSH  BC
        LD    HL,NOM_I
        LD    C,#14       ; копирование дескриптора в память
        CALL  15635
        LD    A,87        ; изменение типа на <W>, но можно и
                          ; другой
        LD    (NOM_N3),A
        LD    HL,NOM_I
        LD    C,#13       ; копирование в дескриптор
        CALL  15635
        POP   BC
        LD    A,C
        LD    C,9         ; запись дескриптора файла на диск
        CALL  15635
        LD    A,8         ; сообщение: SAVE O.K.
        LD    B,3
        LD    C,0
        CALL  ZAP_E
        JP    QUIT2       ; завершение работы
; * подпрограмма вывода сообщений на экран
; * рег. A - номер сообщения в таблице [0-255]
; * рег. HL - номер сообщения [0-32767], но запуск с ZAP_E2
; * рег. B - номер строки, рег. C - номер столбца
```

```
ZAP_E  LD   L,A
       LD   H,0
ZAP_E2 ADD  HL,HL
       LD   DE,st_a      ; начальный адрес таблицы адресов
                         ; сообщений
       ADD  HL,DE
       LD   E,(HL)
       INC  HL
       LD   D,(HL)
       PUSH DE
       LD   HL,#5C3C
       RES  0,(HL)
       LD   HL,#2762
       LD   A,#16        ; установка позиции печати
       RST  #10
       LD   A,B
       RST  #10
       LD   A,C
       RST  #10
       DI
       POP  DE
VOZV2  LD   A,(DE)
       CP   13
       JR   Z,exit2
       RST  #10
       INC  DE
       JR   VOZV2
exit2  RST  #10          ; перевод строки
       RET
; * ожидание любой нажатой клавиши (пауза)
PAUSE  LD   (SAY),IY
       LD   IY,#5C3A
       EI
       RES  5,(IY+1)
LOOP   BIT  5,(IY+1)
       JR   Z,LOOP
       LD   IY,(SAY)
       RET
; * определение нажатой клавиши
; * результат в рег. A
INKEY  LD   (SAY),IY
       LD   IY,#5C3A
       EI
       RES  5,(IY+1)
LOOP2  BIT  5,(IY+1)
       JR   Z,LOOP2
       LD   A,(IY-50)
       LD   IY,(SAY)
       RET
SAY    DEFS 2            ; сохранение IY
D1     DEFS 2            ; номер текущего дисковода
T. AD  DEFS 2            ; адрес, установленный на последний
                         ; адрес блока памяти + 1
T. BUF DEFS 2            ; текущий адрес буфера 0 трека
st. a  DEFW S0           ; таблица адресов сообщений
       DEFW S1
       DEFW S2
       DEFW S3
       DEFW S4
       DEFW S5
       DEFW S6
       DEFW S7
       DEFW S8
       DEFW S9
       DEFW S10
S1     DEFM "ERROR" ; *таблица сообщений*
       DEFB 13
S2     DEFM "No disk. Insert. Press enter."
       DEFB 13
S3     DEFM " "
       DEFB 13
S4     DEFM "Break"
       DEFB 13
S5     DEFM "Files>128. Insert another disk."
       DEFB 13
S6     DEFM "Press ENTER"
       DEFB 13
S7     DEFM " for save"
       DEFB 13
S8     DEFM "Save O.K."
       DEFB 13
S9     DEFM "No space on disk!"
       DEFB 13
S10    DEFM "Insert another disk."
       DEFB 13
```

Алгоритм непосредственной записи на диск может показаться несколько усложненным. Так, например, запись "напрямую" в дескриптор файла не производится, а используется стандартная подпрограмма TR-DOS. Создание (запись) файла осуществляется с помощью функции #0B, которая рассчитана на работу с стандартными типами файлов. И если нам потребуется использовать нечто "другое", то могут возникнуть проблемы — все это сделано с целью свести к минимуму конфликт с операционной системой.

Наиболее полным решением было бы включить обработку ошибок, возникающих при работе системы (отсутствие дискеты, защита от записи, ошибка записи и др.), так как в этом случае программа не "вываливалась" бы с сообщением и стиранием экрана.

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

Оптимизация
на примере intro "Start"

(Окончание. Начало в №№7-9/2001)

Построение таблиц

Если в программе используются какие-либо таблицы, то во многих случаях ее длину удается сократить за счет следующего простого приема — не хранить таблицы в теле программы, а формировать их перед использованием. Выигрыш получается за счет того, что фрагмент программы, строящий таблицу, оказывается короче чем сама таблица.

Разумеется, надо учитывать, что на построение таблицы требуется определенное время, иногда довольно значительное. Впрочем, обычно формирование таблиц происходит лишь один раз, в самом начале работы программы, так что это имеет не такое уж большое значение.

Вот как этот прием используется в intro.

Таблица синусов (точнее, таблица значений функции *y=[(1+sin (2p*x/255))*127]*) длиной 256 байтов создается с помощью 32-байтного фрагмента программы (строки 145...180). Получаем выигрыш в 224 байта — правда, на построение тратится около 12 с. Существуют, впрочем, и более быстрые (но и более длинные) способы построения таблицы синусов — если интересно, рекомендую ознакомиться с моей статьей "Еще о программировании арифметических операций" (раздел "Тригонометрия") [2].

Таблица PALETTE, в которой находятся значения атрибутов, также занимает 256 байтов. А вот фрагмент программы, ее строящий (строки 182...197), занимает всего 14 байтов, плюс 12 байтов на данные (строки 274...285) — итого 26 байтов. Как нетрудно подсчитать, выигрыш составляет 230 байтов.

Таблица со сведениями об используемом при проигрывании музыки сэмпле... Тут ситуация более любопытная. Если вы посмотрите на структуру сэмпла (рис.2), то увидите, что его можно разделить на две части — начало и зацикленный участок. Так вот, начало (первые четыре элемента) так и хра-

нится в теле программы (строки 826...829). А продолжение сэмпла достраивается в строках 75...116 (на это тратится 25 байтов). В итоге общие затраты памяти составляют 29 байтов, а получаемая в итоге таблица SAMPLE занимает 77 байтов. Как видим, 48 байтов при этом удается сэкономить.

Использование недокументированных возможностей процессора

В строках 221, 223, 230 и 245 вы можете увидеть недокументированные команды для работы с младшим и старшим байтами 16-битного регистра IX как с отдельными 8-битными регистрами XL и XH. Их удобно использовать, когда для хранения переменных не хватает обычных регистров. Точно так же можно работать и с половинками регистра IY, правда, в intro этому не удалось найти применения. Хотя команды, работающие с половинками индексных регистров, на байт длиннее и на 4 такта медленнее по сравнению с командами, работающими с обычными 8-битными регистрами (за счет байта-префикса), их использование все же более выгодно, чем хранение переменных в памяти. И, между прочим, в более поздних модификациях процессора Z80 (Z380 и т.д.) эти команды уже не являются недокументированными.

В строках 666...677 (запись значений в регистры AY) для проверки условия выхода из цикла используется недокументированная особенность команды OUTD — после ее выполнения флаг переноса сбрасывается, если значение регистра H не изменилось, и устанавливается, если значение регистра H изменилось и при этом выводимое в порт значение не равно нулю. Собственно, в процессе написания intro я и обнаружил эту особенность — интересующиеся могут прочитать об этом в моей статье "Влияние команды OUTD на флаг переноса" [3].

Цикл построен таким образом, что запись в регистры AY происходит в обратном порядке — от R10 к R0. После записи в R0 значение HL уменьшается с #8000 до #7FFF. Чтобы при этом флаг переноса оказался установлен (а это и служит условием выхода из цикла), необходимо, чтобы выводимое в R0 значение не было равно нулю. Это значение представляет собой младший байт делителя частоты в канале A. На протяжении всей мелодии этот байт никогда не становится равным нулю — это получилось, в общем-то, случайно. Если бы вдруг оказалось, что в какие-то моменты этот байт обнуляется, то пришлось бы изменить мелодию — скажем, выбрав другую тональность.

При использовании недокументированных возможностей возникает вопрос — как это все будет работать при выполнении программы под эмулятором. Команды, работающие с половинками индексных регистров, поддерживаются во всех известных мне эмуляторах, а вот насчет правильного выставления флагов при выполнении команды OUTD такой уверенности у меня нет. Впрочем, как показывает практика, авторы эмуляторов постоянно дорабатывают свои творения, и обнаруживающаяся неработоспособность какой-либо программы лишь служит дополнительным стимулом для повышения качества эмуляции. :-)

Еще несколько хитростей

Изображение в эффекте "плазма" движется по траектории, описываемой уравнениями вида $x=sin(t)$, $y=sin(t+f_i)$. Такая траектория называется фигурой Лиссажу, и ее вид в данном случае зависит от значения f_i — разности фаз. Если $f_i=0$ или p, фигура вырождается в отрезок прямой; если $f_i= p/2$ — превращается в окружность, а при других значениях представляет собой эллипс (что как раз и нужно). В программе значениям f_i от 0 до $2p$ соответствуют числа от 0 до #100. Таким образом, понятно, что разность фаз может быть любой, только не кратной #40 (т.е. $p/2$). Воспользовавшись этой "свободой", я выбрал значение разности фаз равным #AA — это позволило в строке 227 вместо команды ADD A,N (2 байта/7 тактов) использовать команду ADD A,H (1 байт/4 такта). Дело в том, что в регистре H к этому моменту уже содержится число #AA — старший байт адреса таблицы TABLE_SIN. Вот так и удалось сократить программу еще на байт.

Начальное значение t (находящееся в регистре XL) оказывает влияние только на точку, с которой начнется движение, а вид траектории от него не зависит. Поэтому я не задаю это значение и тем самым не трачу лишние байты. Так что, если вы будете запускать intro при различных значениях XL, то всякий раз увидите его немного другим. :-)

При проигрывании музыки используется тот факт, что количество нот в паттерне (8) является степенью двойки. Это позволяет оптимизировать фрагмент программы, где производится увеличение номера ноты и проверка на достижение конца паттерна (строки 465...471). После того как номер ноты увеличен, вместо команды сравнения (CP 8) используется команда AND 7. И та, и другая команда установит флаг Z, если после увеличения номер стал равным 8, так что пока выигрыша вроде бы не видно. Но после команды AND при этом произойдет еще и обнуление аккумулятора, а это нам как раз и надо, ведь после 7-й ноты одного паттерна идет 0-я нота следующего. Вот и выигрыш — не потребовалось специально обнулять номер ноты.

Посмотрите, как происходит формирование 256-байтной таблицы PALETTE. Она должна состоять из 12 равных частей, заполненных значениями, взятыми из 12-байтной таблицы COLORS. Но 256 на 12 нацело не делится, и части получаются такими — 11 частей по 22 байта и одна длиной 14 байтов.

Процедура заполнения (строки 191...197) устроена очень просто. В DE помещен адрес начала таблицы PALETTE. Из (HL) берется очередное значение и 22 раза записывается в (DE) (после записи каждого байта значение E увеличивается). Если после записи очередных 22 байтов достигается конец таблицы PALETTE (т.е E=0), заполнение заканчивается. Если нет, то HL увеличивается, и в таблицу заносятся очередные 22 байта.

Если вы протрассируете эту процедуру, то обнаружите нечто далеко не очевидное с первого взгляда. После того как таблица PALETTE заполнится в первый раз, работа процедуры не закончится. Таблица будет заполняться "по кругу" еще и еще — до тех пор, пока последний байт очередного 22-байтного участка не совпадет с последним байтом таблицы. К этому моменту, как нетрудно подсчитать, в таблицу будет записано 2816 байтов (это наименьшее общее кратное чисел 256 и 22), а HL увеличится на 2816/22=128. Поэтому адрес, устанавливаемый в HL перед началом заполнения (строка 184), меньше чем адрес начала таблицы COLORS на 128-12=#74 байта, так что после того как заполнение завершится, таблица PALETTE будет как раз заполнена данными из таблицы COLORS.

Казалось бы, к чему такие сложности — ведь можно просто добавить проверку достижения конца таблицы во внутреннем цикле. Да, можно — но это займет еще два байта памяти.

Между прочим, подобные таблицы применяются при реализации многих графических эффектов, так что, надеюсь, изложенный принцип их оптимального формирования кому-то пригодится.

Вместо заключения

Ну что, не ожидали, что нужно так много знать, чтобы написать такую маленькую программу? :-) Попробуйте свои силы в оптимизации — сократите intro хотя бы на байт без потери функциональности! Сразу скажу, что это возможно, и один из способов мне известен...

Литература

1. BV_Creator. Z80: оптимизация загрузки констант в регистры. — Радиолюбитель. Ваш компьютер, 2000, № 9, 2001, №2.

2. И.Рощин. Еще о программировании арифметических операций. — Радиолюбитель. Ваш компьютер, 2000, № 12; 2001, №№1-3.

3. BV_Creator. Влияние команды OUTD на флаг переноса. — Радиолюбитель. Ваш компьютер, 2001, №5, С.40.

ООО "НТК ИНФОТЕХ"

А.ВЕНДИЛОВСКИЙ,
г.Минск.

ШТОРМ

Звёзды, как далеки вы, и как жестоки, или как жестоки люди, стремящиеся к вам?

2117 г. — галактическая империя Земли задыхается под тяжестью своих размеров. То и дело вспыхивают восстания и "парады суверенитетов". Император твердой рукой подавляет волнения, стирая в пыль целые планеты, превращая их в безжизненные пустыни, но даже эти драконовские меры уже ничего не могут остановить.

2135 г. — ядерные бомбардировки не прошли даром, планет земного класса осталось очень мало, за контроль над ними стали сражаться все, кто только мог. Гражданская война поднимала голову...

2150...2200 г.г. — выжившие остатки галактической империи создают челноки-колонизаторы и отправляются в неизведанные районы галактики, в надежде найти там пригодные для жизни планеты. Некоторым это удалось...

2200...2300 г.г. — на месте рухнувшей империи создается независимая Федерация Свободных Мирров (ФСМ). Колонии налаживают торговые отношения с Изначальными планетами, к концу 2286 г. ФСМ достигает промышленного уровня довоенной империи.

2320 г. — происходит встреча с потомками одной из земных колоний, которые именуют себя веллианами. Веллианны и ФСМ подписывают соглашение о мире и сотрудничестве. Веллианы передают землянам технологию ноль-порталов и генетических экспериментов. Совместно создается галактическая сеть ноль-порталов.

2351 г. — станции контроля ноль-порталов оказались "троянскими конями". Веллиане перехватывают управление над ними, и военная армада веллиан атакуют беззащитные планеты Федерации. Начало Второй Галактической Войны...

История Земли. Учебник для военных учебных заведений

Глава первая. Введение

Яркое солнце слепило глаза, показания сходящих с ума приборов, отчаянный вопль аварийных систем и настойчивый голос бортового компьютера смешались в шуме надрывно работающего движка. Он должен вернуться, он должен вернуться домой...

... Они вылетели ранним утром — четыре самолета, звеном. Уже тогда их предупредили, что это просто смертельная миссия. В последних боях они потеряли много людей, веллиане знали это, поэтому к базе приближалась большая танковая колонна противника под прикрытием искажающего поля и с воздушной поддержкой. Последняя надежда была на его звено — они должны были прорваться к генератору поля и уничтожить его, позволив остальным эскадрильям вплотную заняться танками.

Кровь заливает глаза, дым проникает под кислородную маску. Вместо левого крыла лишь металлический обрубок с искрящими проводами и дымящимися частями обшивки. Правый движок работает за двоих, но надолго ли его хватит?...

...Как описать бой внутри искажающего поля? Когда плазменное оружие прекращает работать, а приборы становятся "слепы", то надеяться можно только на примитивные системы, вроде малокалиберных пушек и неуправляемых ракет? И яркий солнечный день внезапно становится мрачным, сумрачным, полным коричневых оттенков и предчувствия бури. Зенитных комплексов было понатыкано как сорняков в огороде. "Дельта 3" получил здесь свое, яркая вспышка от разорвавшегося боекомплекта и дымное облачко, из которого на землю посыпался град исковерканных обломков — и нет друга... Только времени на переживания не оставалось, кроме зениток, в воздухе кружило еще пяток истребителей, сорвавшихся с охраны колонны для "теплой" встречи землян. "Черный принц Дельта 2" и "Фокстрот Дельта 4" затеяли смертельную карусель. Свист рассекаемого воздуха, вопли ярости и крики умирающих в эфире, отрывистый стук МКПов и скомканное сознанием время, пока он разворачивал свой самолет для окончательного удара по неприметной на этой местности машине.

Энергия падает с каждой секундой, освещение в кабине самолета уже погасло, сейчас он — легкая мишень. Нос аппарата опасно наклоняется в сторону земли, но каким-то чудом его удается выправить. От потери крови холодеют руки, и перед глазами появляются темные пятна. Он остался один...

...Он слышал, как кричал "Черный принц", что он подбит и теряет управление — в следующую секунду метка его самолета исчезла с радара. Палец надавил на кнопку пуска ракеты... Тогда еще ему не было известно, что у него на хвосте сидел один из вражеских истребителей, и "Фокстрот", уже расстрелявший к этому времени практически весь боезапас, бросился защищать друга единственным доступным для него методом — пошел на таран...

Вдали показались первые земные укрепления, по дороге быстро двигались чьи-то танки, теперь, когда система наведения "сдохла", с такой высоты и не определишь — свои или чужие...

...Ярость и боль потери превратили его в берсерка. После уничтожения генератора заработали плазменные пушки и самонаводящиеся ракеты, которые он щедро выпускал в любую летящую вражескую цель. МКП с трудом справлялся с перегревом. Неуязвимый в своем бешенстве, он настигал их в воздухе и превращал в кровавые ошметки. И его испугались — оставшиеся в живых поторопились отступить. Последнего он достал в лобовой атаке, но тот взорвался слишком близко — левое крыло было срезано словно гигантским ножом, а левую ногу окатило волной боли. По лбу стали стекать маленькие ручейки крови.

Смерть, которая так долго шагала рядом, теперь неслышно подошла к нему. Темный туман сгущался и манил покоем и отсутствием боли, он начал медленно уплывать, чувствуя, как рука выпускает беснующийся штурвал. В тот миг перед ним появилось лицо из далекого прошлого, и старые воспоминания ожили...

...Они, молодые юнцы, стояли перед этим старым асом, командором первой степени, и ждали, когда будут допущены к экзаменам. Генерал, еще раз оглядел строй и поморщился как от зубной боли: "Слушайте меня внимательно, ...! Вы, маменькины сынки, отходившие от ее юбки лишь затем, чтобы выпить пива в баре за углом, возомнили себя пилотами? Это прекрасная и гордая форма, которую вы носите, лучше сидит на чучеле в моем огороде, чем на вас, но вам доверят летать на этой сложной и прекрасной технике, которая так нужна на фронте. Я до сих пор не понимаю, зачем вы здесь, лучше бы сразу прислали стадо бабуинов — их хоть чему-то можно научить. А вас...— генерал еще раз скривился и бросил на строй такой взгляд, что все невольно вытянулись по стойке смирно. — Да от коровьего дерьма, прилипшего к моему ботинку, и то больше пользы. И тот, кто не пройдет тест, или, упаси Бог, разобьет машину, — тут генерал улыбнулся так, что все в строю почувствовали холодные мурашки на спине, — может заранее приготовиться. Я обещаю, что сгною его на самой дальней и вшивой военной базе, где он станет пожизненным чистильщиком запасной третьей взлетной полосы, а в перерывах между чистками и нарядами будет драить зубной щеткой все сортиры в радиусе ста километров. Тому, кто сможет пройти экзамен, быть может, я доверю развозить тележку с супом. По машинам!"

Он засмеялся, потянулся, и боль, так долго мучившая, стала его якорем в

ООО "НТК ИНФОТЕХ"

мире живых. Закусив губу, что-то крича, он сражался с управлением погибающей машины. Онемевшие пальцы набрали код вызова посадки, и невидимые электромагнитные поля бережно подхватили разбитый самолет и мягко направили его к ангару. "Мы еще оплачем павших, отомстим за друзей и порадуемся жизни..."— и лишь когда медики уложили его на носилки, он позволил себе потерять сознание.

Глава вторая. Ода движку

Что сразу вызывает восхищение, так это картинка. Можно только тихо сказать: "Изумительно!", и понять, почему столь уважаемая фирма Matrox выбрала эту игру как лучшую в области графики и дизайна. А уж поверьте, "фуфло" они никогда на премию не выдвигали. Посмотрите на эти небеса — легкая перемещающаяся утренняя дымка, через которую мы быстро пролетим, и медленно плывущие облака. И солнце, настоящее, яркое, СЛЕПЯЩЕЕ (запомните, при атаке на противника нужно заходить со стороны солнца!), вызывающее ряд световых линз и бликов на оптике самолета. За все время, которое я нахожусь за компьютером, это самое реальное отображение солнечных лучей. Когда первый восторг уляжется, подлетите к воде — и вы испытаете легкий шок. Вода действительно полупрозрачна, и очень реально отражает происходящее над ней. Но если действительно хотите понять, чем занимались программисты все эти годы, посмотрите на свой аппарат со стороны — "вылизан" до идеала! В кабине виден пилот; видно, как вращаются пушки при стрельбе и падают на землю гильзы; под крыльями вырывается пламя из сопел двигателей — и все это в реальном динамическом освещении! Представьте, все это можно увидеть, используя всего лишь вторую Вуду! Ну а если вы счастливый обладатель GeForse, то лучше сразу возьмите валидол — игра "выжмет" из карточки все ее хваленые возможности и особенности. Со стороны это напоминает очень крутой американский фильм-блокбастер с многомиллионными спецэффектами. Повреждения вашей машины или врага будут очень четко видны — трудно не заметить оторванное крыло или разбитый коклит.

Но и управлять подбитой машиной очень трудно. Управление этим летающим агрегатом — это отдельный разговор. Набор управляющих клавиш не смутит того, кто прошел Descent FS, но вот мышкой "крутить" намучаешься — очень интересно просчитывается физическая модель и поведение в воздухе, существует понятие "максимальный угол поворота и атаки", различный для каждого аппарата. И будем откровенны, игра изначально "затачивалась" под джойстик (иначе зачем его рекламируют на обложке с диском?).

Все действо происходит на огромном континенте — разработчики утверждают, что потребуется семь часов, чтобы преодолеть его из конца в конец. Все живет своей самостоятельной жизнью — вылетают на боевое дежурство самолеты, иногда вам присылается помощь, и очень часто приказы эскадрильям меняются при изменении боевой ситуации. Кстати, при всей свободе "лети — не хочу" забудьте о самостоятельности — вы находитесь на военной службе, где исполняют приказы. И если приказ недвусмысленен — об атаке конкретной цели, то можете мне поверить, если вы решите изобразить из себя самого умного, то можете быть уверены, миссия будет провалена. Что приятно удивило, так это AI компьютера — враг совсем не собирается изображать из себя легкую мишень, и сбить его — большая проблема. Легких побед здесь не бывает — трудно не только уничтожить врага, но и самому выжить. Противник очень метко стреляет...

Всего в наличии две компании по двадцать пять миссий каждая. Миссии нелинейны — и провал задания не обязательно заставит вас начинать все заново (за что был так ругаем DFS), а просто поведет сюжет по новому руслу.

Что поднимает геймплей такой игры до невиданных высот (ну наконец-то, дождались)? В игре присутствует более пятидесяти летающих, ездящих и плывущих объектов, с которыми вы так или иначе будете "общаться". Так что скучать не придется. Ну а когда все одиночные миссии будут пройдены, добро пожаловать в мультиплеер. Одновременно может сражаться до сорока восьми игроков. Присутствуют режимы capture the flag, teamplay, deathmatch. Битва за флаг такой толпой на самолетах выглядит весьма оригинально...

Глава третья. То, что осталось за кадром

Очень бы хотелось закончить статью восхвалением наших программистов, если бы не одно "но". Игра рассчитана по системным требованиям для первого Пентиума но... Владельцы первых "пней", а также обладатели процессоров AMD K6x, приготовьтесь, что игра даже не запустит "автоменю" с диска, и вам придется копировать полностью весь компакт на винчестер, а после этого из специального каталога на диске заменять инсталляционные библиотеки и запускать инсталляцию игры с "винта". Позже — специальной утилитой указывать место расположения CD-ROM... Этого нудного и загадочного процесса можно было легко избежать путем выбора варианта инсталляции. Мои знакомые программисты были очень удивлены такой ситуацией, и клялись, что могут изготовить нормальный вариант установки игры в течение суток.

Но на этом проблемы не заканчиваются. Очень долго я пытался заставить игру понять, что у меня есть вторая Вуда, но, несмотря на все попытки, сделать это мне не удалось. В Интернете, на сайте производителя, мне удалось найти информацию, что игра просто не находит эту карточку (это было трудно указать в мануале?), и для исправления этой ошибки нужно скачать и установить пятимегабайтный патч (он, кстати, добавляет еще новую миссию). А после установки патча игра отказалась запускаться на моем процессоре К6-2-333. Что я говорил в адрес разработчиков в течение следующего часа, когда производил мичуринское скрещивание двух версий (выборочная замена пропатченных библиотек на те, что должны были запускать игру на моем процессоре) — лучше опустить. Мне удалось заставить ее работать и находить Вуду 2, но я до сих пор не уверен, что в будущем это не приведет к какому-либо сбою во время игры. Допустить такие "ляпы" при таком уровне игры — это просто удивительно, и очень огорчает.

Системные требования

Минимальная конфигурация

Процессор — Pentium II 266, Celeron 300, Duron.

Оперативная память — 64 Мб.

Свободное место на диске — 650 Мб.

Операционная система — Windows 95/98/ME, Windows 2000 Professional, Windows 2000 Server.

Видеокарта — ускоритель второго поколения, поддерживающая DirectX 7.0.

CD-ROM.

Мышь.

Звуковая карта.

В добавление к перечисленным выше основным требованиям необходимо, чтобы на машине был установлен DirectX версии 7.0 или старше и Internet Explorer версии 4.0 или старше. Для игры по сети необходимо иметь TCP/IP network-протокол.

Оптимальная конфигурация

Процессор — Pentium III, Athlon.

Оперативная память — 128 Мб.

Свободное место на диске — 750 Мб.

Операционная система — Windows 95/98/ME, Windows 2000 Professional, Windows 2000 Server

Видеокарта — ускоритель последнего поколения

Полная установка с CD-ROM.

Трехкнопочная мышь.

Трехосный, с 16 кнопками джойстик.

Direct Sound-совместимая звуковая карта.

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:*

141406, г.Москва, Химки-6, ул.Библиотечная, 18-84 или

220095, г.Минск-95, а/я 199,** передавать по тел. **+(37517) 221-01-10,** либо через E-mail: **rm@radio-mir.com

Инвалид 2 группы с детства **купит** по сходной цене или примет в дар компьютер IBM AMD 486DX4-100; комплектующие к 486-му; дискеты с различными программами 5,25" и 3,5" и CD-ROMы.
682905, Россия, Хабаровский край,
р-н им.Лазо, п.Дурмин,
ул.Комсомольская, 1,
Ковалевич Я.В.

Со времен конструирования "ZX-Spectrum" осталось много деталей: ПЗУ (2764, РФ4), Z80 кварцы и др. Отдам за символическую плату или поменяю на другие детали.
620039, Екатеринбург, а/я 172,
Сергей. E-mail: banan@pisem.net.

Куплю недорого к IBM Электроникв МС1502: контроллер винчестера; винчестер; руководство на дискете 5,25" 720К; дискеты.
682905, Россия, Хабаровский край,
р-н им.Лазо, п.Дурмин,
ул.Комсомольская, 1,
Ковалевич Я.В.

Ищу схему Atari65xe.
Костя.
E-mail: kocty@rambler.ru

Куплю дисковод, контроллер, системный диск, ПЗУ "супер+", ROM-диск и т.д, для компьютера "Вектор 06Ц".
423950, Татарстан, р.п. Уруссу,
ул.Тукая, 19 — 10,
Сибен А.К.,
тел.(85510) 2-43-24.

Куплю контроллер для ПК "Байт".
Тел. в г.Горки Могилевской обл.
(8-02233) 38-7-73,
Владимир.

Предлагаю огромное количество программ для ZX-Spectrum на 5.25" и 3.5" FD. Есть все! Также есть в наличии много 5.25" (720 к) дисков, литература и многое другое.
Адрес: 213200, Беларусь,
Могилевская обл., г. Чаусы,
ул. Фрунзе, 28
Зарецкий Евгений Георгиевич,
тел. (02242) 21136,
E-mail: zeg@tut.by

Уважаемые читатели!

Те, у кого возникли проблемы с подпиской на наши журналы, 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	4, 8	1 — 5, 7 — 12	1, 4, 5, 6, 7, 11, 12	25	800	4
1999	3, 9, 10, 11	1 — 6, 10, 11, 12	3, 5, 6	30	1000	5
2000	4	1 — 12	4, 7, 10, 11, 12	35	1200	6
2001	3 — 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

Наши платежные реквизиты:
- 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;
- для жителей России, Украины и др. стран СНГ —
получатель : ООО "НТК ИНФОТЕХ", ИНН 7703155561,
р/с 40702810100022120172 в АКБ "Межтопэнергобанк"
корр. счет 30101810900000000237 БИК 044585237
Адрес банка: 107078, г.Москва, ул.Садовая-Черногрязская, 6.

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@radio-mir.com

Приобретение отдельных номеров журналов

В магазинах радиодеталей **"ЧИП и ДИП"** по адресам:
- 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.Ярославль, ул.Нахимсона, 12, тел. (0852) 27-57-15.

В **"Бермосе"** по адресу: г.Москва, ул.Садовая-Спасская, 19/1
(ст. метро "Красные ворота").

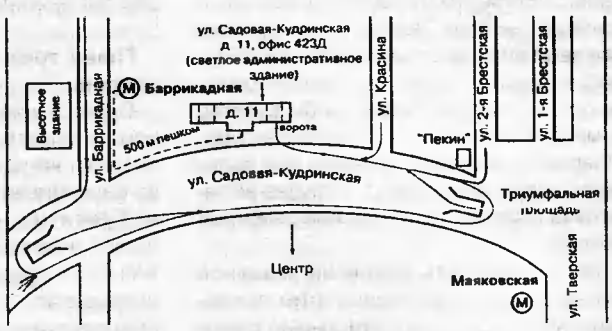
На радиорынках в Москве: Митинском (места R4, S8, K52, E50, G56), Царицынском (место 121).

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

На Украине: г.Киев, фирма **"Торм"**, твл. +(38044) 227-42-13.

В Беларуси: г.Минск, пр.Ф.Скорины, д.92, магазин **"Книга XXI век"**, тел. (017) 264-27-97 (ст.метро "Московская").

В издательстве **"Солон-Р"**, по адресу: г.Москва, ул.Садовая-Кудринская, д.11, офис 423Д (ст. метро Баррикадная, ст. метро Маяковская).
Время работы:
с 09.00 до 18.00
кроме субботы,
воскресенья.
Твлефоны:
(095) 254-44-10,
252-36-96,
факс: (095) 252-72-03.
Радиорынки: Царицынский (место 13/А), Митинский (ряд 8, контейнер 12, место Z25).
E-mail: Sokolov-Solon@coba.ru и Solon-R@coba.ru



Открылся новый радиорынок!

Его официальный адрес: МО, Одинцовский р-н, 19-й км магистрали М1.
Журналы можно приобрести в павильонах "DMK-Press" — места 8-22, 8-23.

А.Вендиловский.

Шторм


ФАРГУС
НА РУССКОМ ЯЗЫКЕ
MYST III
EXILE
2CD
РУССКАЯ ВЕРСИЯ
ЗОЛОТАЯ КОЛЛЕКЦИЯ
2: STEEL SOLDIERS
1CD
РУССКАЯ ВЕРСИЯ
STEEL
ВСЕ ДАЛЬНОБОЙЩИКИ
РУССКИЕ ВЕРСИИ
ДАЛЬНОБОЙЩИКИ 1
ДАЛЬНОБОЙЩИКИ 2
СБОРНИК ИГР
ФАРГУС
НА РУССКОМ ЯЗЫКЕ
СПОРТ ТАЙМ 2001
NHL 2001
FIFA 2001
NBA 2001
3В1
ГЕРОИ III
МЕЧА И МАГИИ
ДЫХАНИЕ СМЕРТИ
ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
ФАРГУС
ИГРЫ НА ДВИЖКЕ QUAKE
СОЛДАТЫ ФОРТУНЫ
HALF LIFE
QUAKE III Arena
3В1
ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
ДОБРО и ЗЛО
BLACK & WHITE
РУССКАЯ + АНГЛИЙСКАЯ ВЕРСИИ
2В1
ARCANUM
OF STEAMWORKS & MAGICK OBSCURA
КОЛЛЕКЦИЯ ИГРУШЕК
ТРОПИКО
ШТОРМ
КОЛЛЕКЦИЯ ИГРУШЕК
ФЛАНКЕР 2
ЛИЦЕНЗИОННАЯ КОПИЯ ИГРЫ ОТ "1С"