№ 12 (205) март 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Электронные версии газеты “Первое сентября” и приложений http://www.1september.ru

Электронная подписка на газету “Первое сентября” и дайджесты приложений

Дорогие коллеги!

По многочисленным просьбам тех, кто не имеет возможности работать с сервером “Первого сентября” **www.1september.ru** в on-line, а использует только электронную почту, мы организовали рассылку электронных выпусков наших изданий по e-mail. По электронной почте рассылаются электронные выпуски газеты “Первое сентября” и электронные дайджесты приложений. Электронные версии изданий рассылаются в формате HTML, для их просмотра можно использовать обычные браузеры (Internet Explorer, Netscape Navigator и пр.). Электронная газета “Первое сентября” рассылается накануне выхода бумажной, электронные дайджесты приложений – один раз в месяц.

Электронная газета “Первое сентября” включает от 50 до 100 процентов материалов бумажной, содержание электронных дайджестов приложений определяют редакции приложений.

Данная услуга является платной (доступ к серверу “Первого сентября” бесплатный).

- **Стоимость месячной подписки на электронную газету в первом полугодии – 10 рублей.**
- **Стоимость месячной подписки на один (любой) дайджест в первом полугодии – 10 рублей.**
- **Стоимость годовой подписки на электронную газету (1999 г.) – 100 рублей.**
- **Стоимость годовой подписки на один (любой) дайджест (1999 г.) – 100 рублей.**

Пример:

За подписку на электронную газету “Первое сентября” на 1999 г. и дайджесты “Математики”, “Физики” и “Информатики” на один месяц вы заплатите 100+10+10+10=130 р.

Для оформления подписки следует:

1. В любом отделении Сбербанка произвести платеж (реквизиты указаны ниже).

2. ОБЯЗАТЕЛЬНО отправить письмо на адрес **electron@1september.ru.** В письме необходимо указать Ф.И.О. (полностью); электронный адрес, по которому мы будем производить рассылку; издания, на которые вы подписались, с указанием сроков подписки (если подписка оформлена не на год, то необходимо указать конкретные месяцы); сумму платежа; дату платежа.

Пример:

Иванов Иван Иванович; **post@mail.ru**; “Первое сентября” – год, “Физика” – апрель, “Математика” – апрель, “Информатика” – апрель; 130 р.; 20.03.99.

Реквизиты для платежа (форма ПД-4)

Получатель платежа: ООО “Чистые пруды”, ИНН 7730087529. Расчетный счет: 40702810138260100648 в Киевском ОСБ № 5278 МБ АК СБ РФ, к/счет 30101810600000000342, БИК 044525342.

Вид платежа: Электронная подписка.

Интернет для начинающих

А.А. ДУВАНОВ,
duvanov@robotland.users.botik.ru

Продолжение.
См. № 36, 40, 44, 46/98; 2, 5, 7, 9, 11/99

1.10. Что новенького? (Сетевые конференции)

Вася. Что новенького?

Папа. Ты спрашиваешь о моей командировке в Москву на конференцию по школьной информатике?

Вася. Конечно!

Папа. В этой области всегда есть новинки. Выступало много интересных людей. От Роботландии был Первин Юрий Абрамович. Он рассказывал о “Роботландском сетевом университете информационных знаний и технологий”.

Вася. Выступление Юрия Абрамовича было интересным?

Папа. С роботландцами никогда не бывает скучно. Мне очень понравился сетевой исполнитель Жук, о котором он рассказывал.

Вася. С Паучком я знаком, а с Жуком нет. Что умеет делать этот исполнитель?

Папа. Жук — любитель путешествий. Его мир, подобно Интернету, состоит из комнат-узлов, соединенных между собой дорожками-связями. В каждой комнате живет какой-нибудь роботландский чудак. Вот Жук ходит по комнатам и разговаривает с обитателями.

Вася. Жук бродит по своей среде сам по себе?

Папа. В этом-то весь интерес: Жуком можно управлять по электронной почте, а заодно осваивать правила электронной переписки.

Вася. Жаль, что конференции школьников так редки. Нам тоже было бы интересно встретиться, обменяться мнениями, послушать друг друга, поделиться информацией.

Папа. Живое общение, конечно, ничто не заменит, но участвовать в конференциях можно и по Интернету, никуда не выходя из дома.

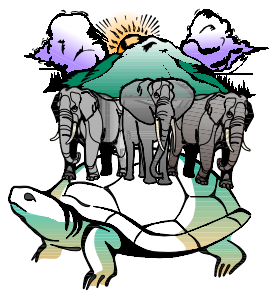
Продолжение на с. 13

НАШИ ДЕТИ БУДУТ ЖИТЬ В XXI ВЕКЕ

12 лекций о том, для чего нужен школьный курс информатики и как его преподавать

Лекции 11, 12

А.Г. КУШНИРЕНКО, Г.В. ЛЕБЕДЕВ



Об основных понятиях, идеях и целях школьного курса информатики “по учебнику” *А.Г. Кушниренко, Г.В. Лебедева, Р.А. Свореня* “Основы информатики и вычислительной техники” (М.: Просвещение, 1990, 1991, 1993, 1996). Дается также ряд практических советов, предлагаются соответствующие методические приемы.

Авторы надеются, что материал окажется полезным для учителей и методистов, использующих указанный учебник, а также для тех, кто желает сравнить разные подходы к преподаванию школьного курса информатики или разработать свой собственный курс.

В этом выпуске представлены лекция “Применения ЭВМ”, а также начало последней лекции (начало “Заключения”).

Продолжение следует

Институт Новых Технологий проводит

Московский открытый конкурс
Школьные web-страницы-99

Цель конкурса: поиск и распространение успешного опыта формирования в школах собственной телекоммуникационной образовательной среды, вхождения в глобальную сеть Internet, зарождения новых моделей учебной деятельности.

Срок подачи заявки: до 30 марта 1999 года.

Форма подачи заявки: письмо в Оргкомитет конкурса (**intpubl@glasnet.ru**) с адресом web-страницы в Internet (URL). В теме письма должно быть помечено: “Конкурс web-страниц”, в тексте указан город и тип вашего учебного учреждения.

Подведение итогов: состоится не позже 15 мая. Жюри конкурса приступит к рассмотрению сайтов участников со 2 апреля. Порядок награждения и точные сроки будут указаны позднее.

Все желающие могут стать членами общественного жюри. Для этого до 15 апреля направьте в Оргкомитет адреса трех наиболее понравившихся сайтов с мотивацией выбора. Мнение общественного жюри будет учитываться при подведении итогов.

Как создать хорошую школьную web-страницу?

Все зависит от вашего воображения, целей, от того, скольких энтузиастов вам удалось привлечь к созданию и поддержанию вашего сайта!

Web-страница школы может быть визитной карточкой, рассказывающей о жизни школы. Она может отражать специфику учебных планов, представлять авторские программы, рассказывать об особенностях используемых образовательных технологий. На ней могут храниться и рассказы об истории школы, и описания школьных поездок, сочинения школьников, биографии учителей и учеников и многое другое.

Web-страница школы может быть и инструментом организации внутришкольной информационно-образовательной среды, помогающей:

- распространять образовательную (материалы к занятиям, тесты, контрольные работы и пр.) и общественную (сообщения о культурных, спортивных и др. мероприятиях) информацию;
- принимать участие во внутришкольных, региональных и международных телеконференциях;
- общаться с учениками и учителями из других школ, городов и стран;
- поддерживать связь между школьными и домашними компьютерами.

Web-страница школы может содержать и информацию для родителей. А может быть, вместе со школьным библиотекарем вы захотите создать электронный каталог школьной библиотеки и, найдя партнера в вашем городе, сможете обмениваться книгами с другими школами.

Хорошая web-страница имеет прозрачную структуру, удобную навигацию, поддерживает кодировки русских шрифтов, содержит контактный телефон, факс, почтовый и электронный адреса школы.

Номинации конкурса: Оргкомитет не определяет сейчас номинации конкурса, это во многом будет зависеть от самих участников и их работ. Номинаций будет несколько. Будут учитываться и информативность, и посещаемость, и дизайн (только не перегружайте сайт графикой и скриптами!).

Мы приветствуем также смелые идеи и остроумные решения!

Успехов вам!

Оргкомитет конкурса

PAINTBRUSH: *изучаем играя*

Ю.Л. ВОРОНЦОВА, О.Ю. САЛОВА

*Окончание. Начало см. в № 10/99**

Уроки № 17, 18, 19. Орнаменты

Цель: научить наклонять и растягивать выделенные объекты, составлять геометрические и растительные орнаменты.

Этот урок мы рекомендуем начать с рассказа об орнаментах, привести соответствующие примеры. Мы рассмотрим геометрический, растительный и смешанный орнаменты.

Сначала предложим ребятам нарисовать геометрический орнамент, например, такой:

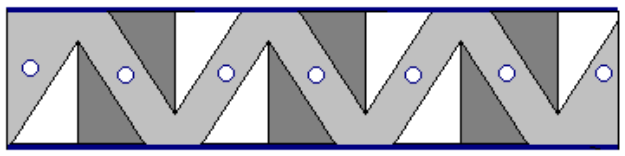


Рис. 20. Геометрический орнамент

Или такой:

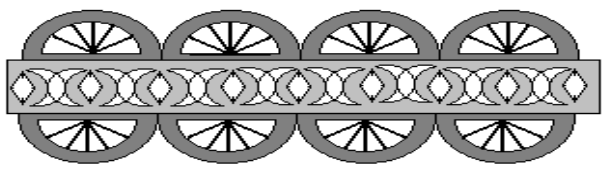


Рис. 21. Геометрический орнамент

Затем на примере растительного орнамента отработаем приемы наклона и растяжки выделенного фрагмента рисунка. Для этого нарисуем фрагмент.

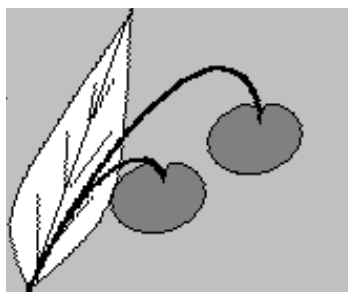


Рис. 22. Фрагмент растительного орнамента

1. С помощью **Линии Безье** нарисуем стебельки, после чего **Наклоним** их на 30 градусов вправо.
2. Нарисуем вишенку, скопируем ее и скомпонуем вишневую веточку.
3. С помощью **Линии Безье, Копирования** и **Отображения** нарисуем листок и **Наклоним** его на 15 градусов. Соберем все детали рисунка вместе.
4. **Скопируем** готовый фрагмент и размножим его для получения орнамента.

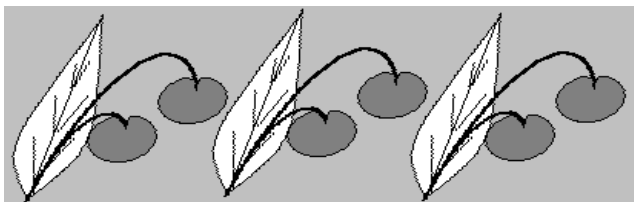


Рис. 23. Растительный орнамент

При наклоне может нарушиться непрерывность линий, поэтому перед **Заливкой** с помощью **Лупы** проверьте линии на разрыв.

Предложите ребятам самостоятельно нарисовать следующий орнамент:

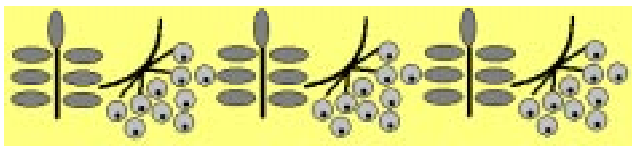


Рис. 24. Растительный орнамент

Чаще мы встречаем сочетание геометрического и растительного орнамента в одном рисунке, например:

Рис. 25. Комбинированный орнамент

Ключ к заданию

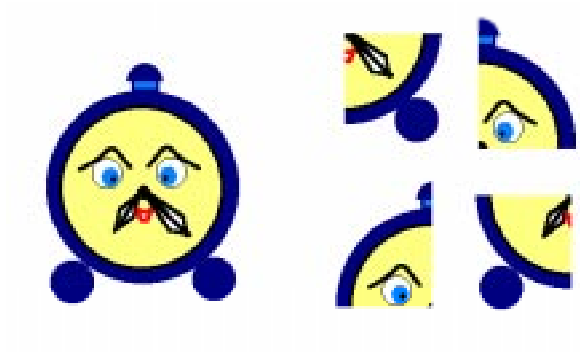
1. Инструментом **Линия** нарисовать угол и треугольник в нем.
2. На свободном поле рисунка при помощи **Дуги** нарисовать половину бутона, **Скопировать** ее, **Отразить** слева направо. Готовый бутон поместить внутри треугольника.
3. **Кистью** нарисовать веточку, **Наклонить** ее, ранее описанным способом получить симметричную и поместить их к бутону в треугольнике.
4. Готовый элемент орнамента **Скопировать**, **Отразить** сверху вниз.
5. **Копируя** и **Перемещая** элементы, составить орнамент между горизонтальными направляющими линиями.
6. Инструментом **Заливка** раскрасить орнамент.

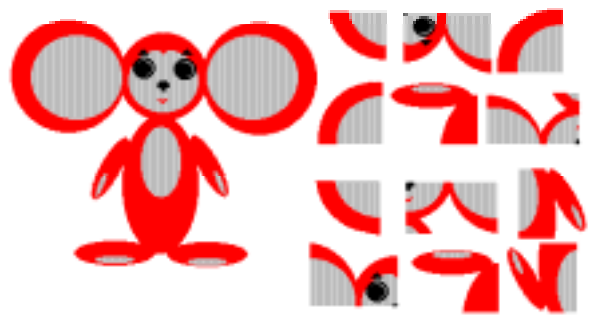
Для закрепления пройденной темы предложите ребятам разработать геометрический и растительный орнаменты и нарисовать их.

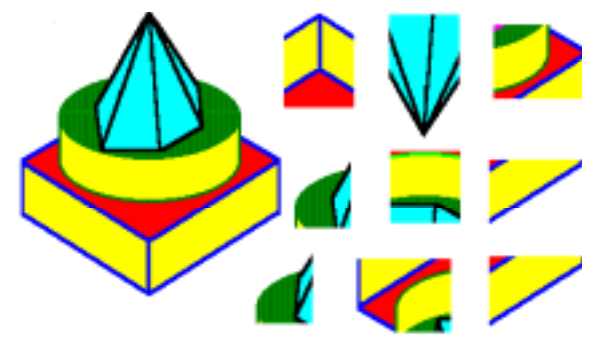
Уроки № 20, 21. Мозаика

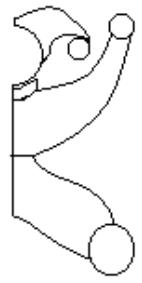
Цель: закрепить навыки по отражению фрагмента, научить поворачивать фрагмент на заданный угол.

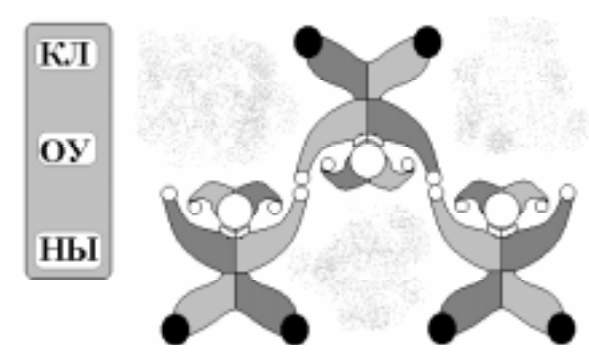
Мы в детстве не раз играли в мозаику, составляя рисунки из кубиков, пуговичек, кусочков картона. Вот и сейчас мы предлагаем поиграть в мозаику, только компьютерную. При сборе мозаики потребуется выделять и перемещать фрагмент, отображать и поворачивать его. Если фрагменты мозаики расположены очень близко, то выделять их можно с помощью инструмента **Лассо**. Имейте в виду, что фрагмент, выделенный **Лассо**, можно только перемещать, при повороте будут появляться искажения рисунка.

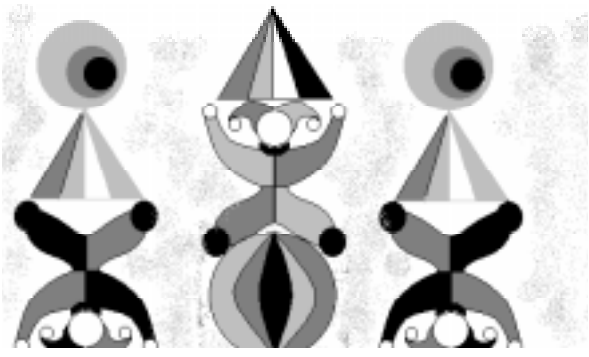












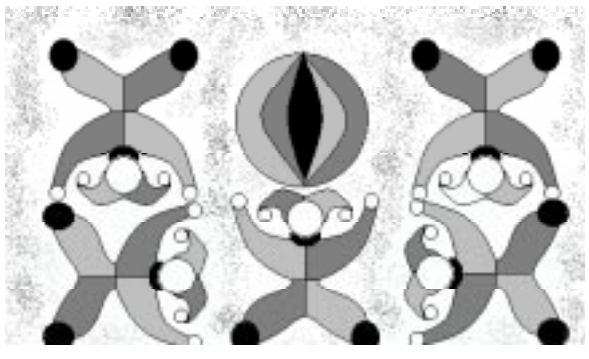


Рис. 27. “Цирковые клоунады”

* Перечень уроков, опубликованных в № 10/99, приведен в конце статьи.

Уроки № 22, 23, 24. Одевалки

Цель: научить экспортировать фрагмент рисунка на новый графический лист.

Для проведения занятий по теме вам потребуются заготовки платьев, костюмов и т.д.

Рекомендуем начать урок с рассмотрения следующих вопросов:

- история одежды;
- элементы одежды;
- модные цвета и цветовые гаммы;
- украшения;
- мода.

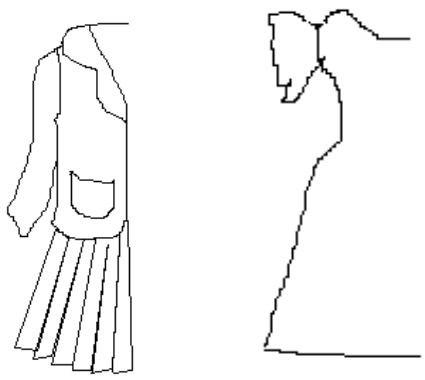


Рис. 28. Фрагменты костюмов

Скопируйте и отразите предложенные фрагменты, поработайте с цветом, а также с элементами украшений.

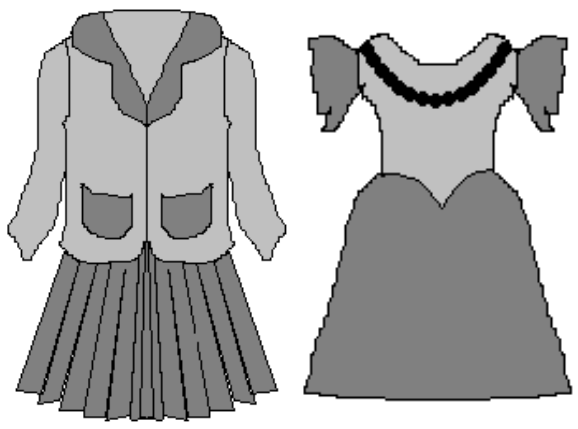


Рис. 29. Костюмы

Значительно интереснее не абстрактно разрабатывать одежду, а примерять ее на манекен. Манекеном у нас будет кукла, хранящаяся в отдельном файле. Ее можно одевать в уже готовые одежки (для этого надо экспортировать куклу на лист, где размещена одежда).

Рис. 30. Кукла-манекен

УРОКИ

Уроки № 25, 26, 27, 28. Чертилки

Цель: научить рисовать план, эскиз, чертеж.

Конечно, за несколько уроков нельзя научить ребят черчению. Но такая цель и не ставится. Достаточно познакомить ребят с некоторыми элементами черчения, пусть они самостоятельно начертят план школы, сделают чертеж простой детали.

Нарисуем план этажа школы. При выполнении плана мы будем менять толщину линий, но не будем выдерживать строгие размеры, станем только соблюдать пропорции.

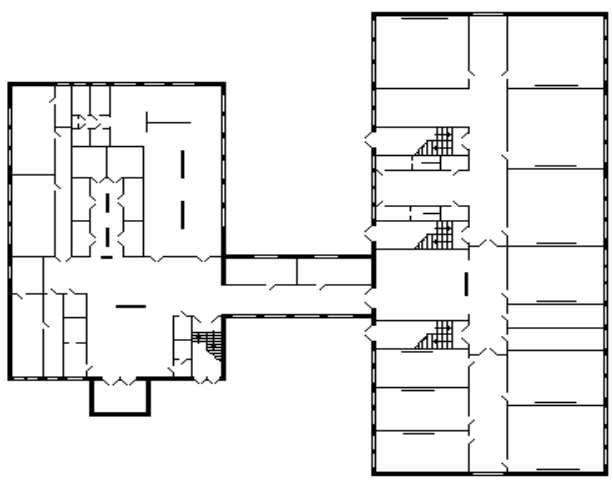


Рис. 31. Схема этажа школы

План класса.

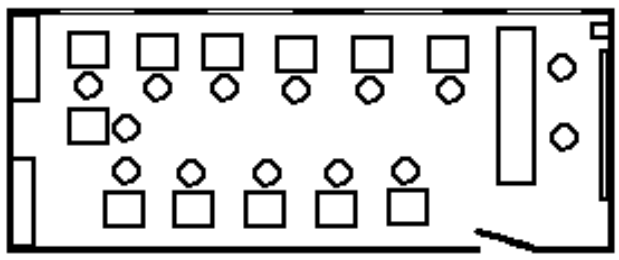


Рис. 32. План класса

Чертеж простой детали.

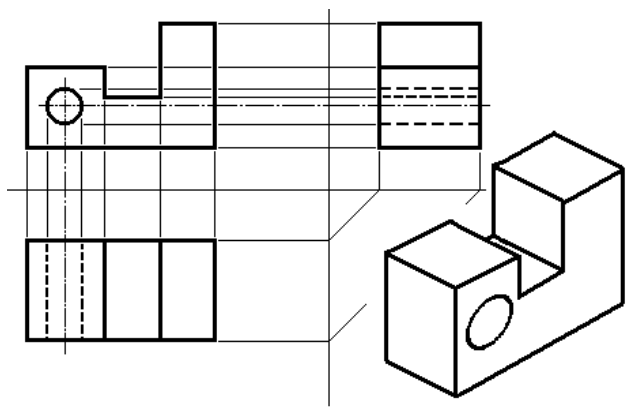
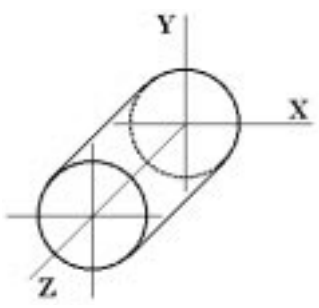


Рис. 33. Чертеж детали

Построим диметрические и изометрические проекции фигур.



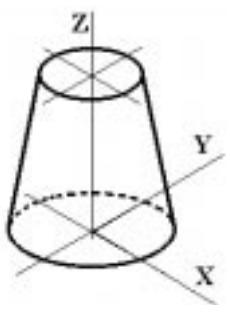


Рис. 34. Проекции фигур

Начертим электрическую схему.

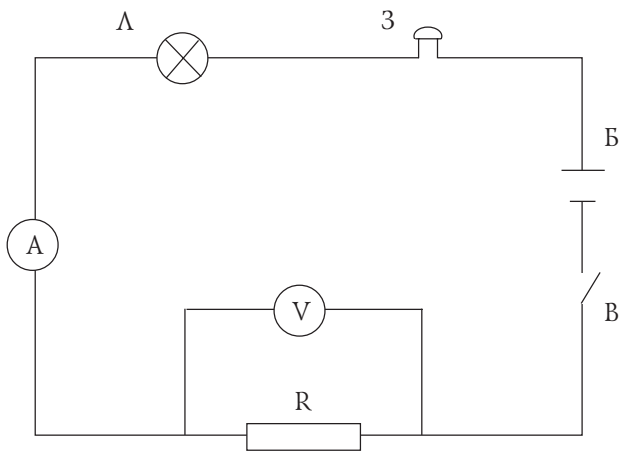


Рис. 35. Электрическая схема

Дорогие коллеги! Чтобы ваши уроки были интересными, не замыкайтесь только на обучении техническим приемам работы с компьютером. Делайте уроки познавательными, используйте материалы из различных областей знаний. Иллюстрируйте свои рассказы рисунками и репродукциями, образцами и моделями. При подготовке материала уроков обращайтесь за помощью и советом к учителям соответствующих дисциплин. Планируйте и проводите с ними совместные работы.

Цель настоящих занятий — показать, что компьютер — это всего лишь инструмент для решения многих жизненных задач. Как ручка в руках писателя, как кисточка в руках художника, так и компьютер в ваших руках позволит окунуться в мир разных профессий и увлечений, в мир фантазий и настоящей работы...

Литература

1. *Ратбон Э.* Windows 95 для "чайников". Учебный курс. К.: Диалектика, 1996, 272 с., ил.
2. *Ю.Шафрин.* Основы компьютерной технологии. Москва: ABF, 1996, 560 с., ил.
3. *Ю.Шафрин, О.Ефимова.* Практикум по компьютерным технологиям. Москва: ABF, 1997, 560 с., ил.

Заключение

Перечень уроков, опубликованных в первой части статьи (№ 10/99)

Уроки № 1—2. Разукрашки
Урок № 3. Рисовалки
Уроки № 4—5. Распылялки
Уроки № 6, 7, 8, 9. Рисовалки фигурами
Уроки № 10, 11, 12. Копирование
Уроки № 13, 14, 15, 16. Ввод текста

Решение задачи X международной олимпиады по информатике “Контакт”

Е.В. АНДРЕЕВА

Доктор Astro Insky работает в радиотелескопическом центре. Недавно она заметила очень странное пульсирующее микроволновое излучение, исходящее непосредственно из центра Галактики. Посланы ли эти сигналы какой-нибудь внеземной формой разумной жизни? Или это простое сердцебиение звезд?

Задача. **Вы должны помочь доктору Insky, создав инструмент для анализа битовых последовательностей в файлах, которые она записала. Доктор Insky хочет найти битовые подпоследовательности длиной от *A* до *B* включительно, которые встречаются в записанной последовательности наиболее часто. Подпоследовательности могут перекрываться. Определяются *N* наибольших различных частот (т.е. количество вхождений). Учитываются подпоследовательности, которые встречаются хотя бы один раз.**

Входные данные

Файл CONTACT.IN содержит данные в следующем формате:

Первая строка содержит целое *A* — минимальную длину подпоследовательности.

Вторая строка содержит целое *B* — максимальную длину подпоследовательности.

Третья строка содержит целое *N* — количество различных частот вхождения.

Четвертая строка содержит последовательность символов 0 и 1, ограниченную символом 2.

Пример входного файла:

```
2
4
10
0101001001000100011110110000101
0011001111000010010011110010000
0002
```

В примере требуется отыскать десять (третья строка) наибольших частот вхождения подпоследовательностей длиной от двух (первая строка) до четырех (вторая строка) в последовательности (четвертая строка файла)

```
0101001001000100011110110000101
0011001111000010010011110010000
000
```

Обратите внимание, что четвертая строка входного файла разделена на несколько строк для того, чтобы поместиться на бумаге. В этом примере подпоследовательность 100 встречается 12 раз, а подпоследовательность 1000 встречается 5 раз. Наиболее часто встречается подпоследовательность 00.

Выходные данные

Файл CONTACT.OUT содержит не более *N* строк, каждая из которых содержит одну из *N* наибольших частот вхождения и соответствующие ей подпоследовательности. Строки должны располагаться в порядке убывания частот вхождения и иметь следующий вид:

```
частота подпосл. подпосл. … подпосл.
```

— где `частота` — это количество вхождений указанных далее подпоследовательностей. Подпоследовательности в каждой строке должны располагаться в порядке убывания длины. Подпоследовательности одинаковой длины должны располагаться в порядке убывания их числового значения. Если в последовательности встречаются менее *N* различных частот, выходной файл будет содержать меньше *N* строк.

Пример выходного файла:

Для указанного примера выходной файл имеет следующий вид:

```
23 00
15 10 01
12 100
11 001 000 11
10 010
8 0100
7 1001 0010
6 0000 111
5 1000 110 011
4 1100 0011 0001
```

Ограничения

Размер входного файла не превышает 2-х мегабайт.

Параметры *A*, *B* и *N* имеют такие ограничения:

$0 < N \le 20$

$0 < A \le B \le 12$

Решение

В силу того что файл входных данных в данной задаче может быть достаточно большим, решать ее нужно наиболее эффективным способом, читая входной файл всего один раз. Пусть мы уже считали из файла k символов битовой последовательности. Так как нас интересуют различные подпоследовательности длины от 1 до 12, то в массиве `pos:array[1..12] of word` будем хранить подпоследовательности, заканчивающиеся k-м битом, длина которых соответствует индексам элементов массива `pos`. Сами же элементы массива есть целые числа в диапазоне от 0 до $2^{12}-1=4095$, так как любой битовой последовательности, длина которой не превосходит 12, можно поставить в соответствие десятичное число, равное результату перевода двоичного числа, состоящего из не более чем 12 цифр, в десятичную систему. Например, битовая последовательность 011011 соответствует десятичному числу 27. Тогда при считывании $k+1$-го элемента входной битовой последовательности c_{k+1} мы можем пересчитать текущую подпоследовательность длины i так: $\mathtt{pos}_{k+1}\mathtt{[i]}=(\mathtt{pos}_{k}\mathtt{[i]}*2+c_{k+1}) \bmod (2^i)$. Здесь умножение на 2 соответствует сдвигу битовой последовательности влево на один разряд (в Паскале это можно реализовать непосредственно с помощью битовой операции сдвига влево: `shl 1`), прибавление c_{k+1} — внесению значения c_{k+1} в последний разряд, а остаток от деления результата сложения на 2^i означает выделение последних i разрядов полученного числа, то есть отбрасывание ставшего ненужным $i+1$-го справа разряда. Например, если подпоследовательность длины 6, заканчивающаяся на k-м символе $\mathtt{pos}_{k}\mathtt{[6]}=011011$, а $c_{k+1}=0$, то $\mathtt{pos}_{k+1}\mathtt{[6]}=110110$.

Общее количество сформированных таким образом подпоследовательностей даже одной длины может иметь порядок $2 \cdot 10^6$, то есть хранить их все одновременно даже в динамической памяти не представляется возможным. Однако все полученные подпоследовательности длины i соответствуют всего 2^i различным десятичным числам от 0 до 2^i-1. Поэтому для анализа всех подпоследовательностей длины i достаточно завести массив длинных (четырехбайтовых) целых чисел размером 2^i, то есть максимум размером 4096 при $i=12$. В каждом элементе такого массива мы будем подсчитывать количество соответствующих значению индекса элемента подпоследовательностей. Так, при формировании подпоследовательности 011011 на единицу будет увеличено значение 27-го элемента массива, учитывающего подпоследовательности длины 6. Таким образом, анализируя все подпоследовательности по мере их формирования, к окончанию считывания входного файла мы уже будем знать, сколько и каких подпоследовательностей он содержал. Остается только, проанализировав эту информацию, найти n максимальных частот вхождения во входную строку любых подпоследовательностей (полученный массив как раз и содержит частоты вхождений различных подпоследовательностей). Когда n максимальных частот найдены, мы еще раз просмотрим весь массив частот и распечатаем двоичное представление индексов тех элементов, частота вхождения которых в исходную последовательность вошла в n лучших.

Для печати i правых цифр двоичного представления некоторого целого десятичного числа можно воспользоваться битовыми операциями. А именно: m-я справа двоичная цифра десятичного числа p равна нулю, если истинно логическое выражение `((1 shl (m-1))and p=0)`, и равна единице в противном случае. Таким образом мы избегаем использования алгоритма перевода десятичного числа в двоичное.

Приведем теперь полный текст программы решения задачи “Контакт”. Заметим только, что для хранения частот вхождения всех подпоследовательностей статической памяти недостаточно. В программе для этого используются 12 массивов размером 2^{12}, созданных в динамической памяти. Такая структура данных при описании имеет следующий вид:

```
p:array[1..12]of ^array[0..1 shl 12-1]of longint.
```

Тогда, для того чтобы в программе обратиться к минимальному элементу этой структуры, соответствующему, например, подпоследовательности 011011 длины 6, нужно использовать конструкцию `p[6]^[27]`. В остальном же использование такой структуры данных не отличается от статических массивов. Реально в программе будут созданы не все 12 динамических массивов, а лишь с a-го по b-й, согласно считанным из входного файла параметрам задачи.

```
const maxd=1 shl 12-1;{maxd=2^12-1}
type pp=array[0..maxd] of longint;
var
  {массив для подсчета количества различных
   подпоследовательностей:}
  p:  array[1..12] of ^pp;
   {массив для формирования подпоследовательностей
   длины от 1 до 12:}
  pos:array[1..12] of word;
  nn: array[1..21] of longint;
                {массив максимальных частот}
  a,b,i,j,n,m,k,d:longint;
  c:char;
  flag:boolean;
procedure ReadData;
begin
  assign(input,'contact.in');
  reset(input);
  readln(a,b,n);
  for i:=a to b do
    begin
      new(p[i]);{создание массива p[i]^
                  в динамической памяти}
      Fillchar(p[i]^,sizeof(p[i]^),0)
  {заполнение массива p[i]^ длины maxd+1 нулями}
    end;
  for i:=a to b do pos[i]:=0;
  k:=0;
  while not eof do
  begin
    while not eoln do
    {считываем очередную строку}
    begin
       read(c); {считываем очередной символ
                 входного файла}
      k:=k+1;
      if k>b then k:=b;{длина, большая b,
                        нас не интересует}
      d:=ord(c)-ord('0');{перевод символа
                          цифры в число}
      if d in [0,1] then
        for i:=a to b do
          begin
            {изменяем i-ю подпоследовательность}
           pos[i]:=(pos[i] shl 1+d) mod (1 shl i);
           {учитываем полученную
            подпоследовательность}
            if k>=i then inc(p[i]^[pos[i]])
          end
    end
   readln {переходим к следующей строке}
  end
end;
procedure Solve;
begin
  for k:=1 to n+1 do nn[k]:=0;
  k:=0;
  for i:=a to b do
    for j:=0 to 1 shl i-1 do
    begin
      flag:=true;
      for m:=1 to k do
        if p[i]^[j]=nn[m] then
          begin
            flag:=false;
            break
             {число p[i]^[j] в массиве nn уже есть}
          end;
      if not flag then continue;
      {вставка в массив nn текущей длины k
       числа p[i]^[j]}
      m:=k;
      while (m>0)and(p[i]^[j]>nn[m]) do
      {ищем место для вставки}
      begin
        nn[m+1]:=nn[m];
        m:=m-1
      end;
      if (m=0)or(p[i]^[j]>nn[m+1])and(p[i]^[j]<nn[m]) then
      begin
        m:=m+1;
        nn[m]:=p[i]^[j];
        k:=k+1;
       {увеличиваем текущую длину массива}
        if k>n then k:=n {нас интересуют
                  только первые n элементов}
      end
    end;
    assign(output,'contact.out');
    rewrite(output);
    for k:=1 to k do
    begin
      write(nn[k]);{печать частоты}
      for i:=b downto a do
        for j:=1 shl i - 1 downto 0 do
          if p[i]^[j]=nn[k] then
            begin
              write(' ');{разделитель между
                          подпоследовательностями}
              for m:=i-1 downto 0 do
          {печать двоичного представления числа}
                if (1 shl m)and j<>0 then
                  write(1) else write(0)
            end;
      writeln {окончание печати
    подпоследовательностей с частотой nn[k]}
     end
end;
begin {основная программа}
  ReadData;
  Solve
end.
```

Самое популярное профессиональное издание для учителей информатики — газета

ИНФОРМАТИК

пятый год издания • 4 номера в месяц • 48 номеров в год

Дорогие читатели!

“Информатика” выходит с января 1995 года. Цель и назначение газеты — быть надежной методической опорой любому учителю информатики.

Преподаватели с многолетним стажем и начинающие, обладатели современного компьютерного класса и те, кто учит детей, довольствуясь самым скромным оборудованием, те, кто ведет профильные курсы, и те, кто работает по минимальному учебному плану в самой обычной школе, находили и обязательно найдут на наших страницах материал, для них предназначенный. В течение прошедших четырех лет сформировались основные направления и рубрики. На страницах газеты — “Задачи”, “Экзамены”, “Олимпиады”, “Языки программирования”, “Новые информационные технологии”, “Как это делаю я”, “Учебники” (новые учебники!), “Документы” (официальные документы, их квалифицированное толкование, ответы на вопросы), “Материалы к уроку”, “Круглый стол”. Мы с трудом умещаемся в заданный газетный объем и стараемся, чтобы каждая публикация была существенной помощью учителю при подготовке к уроку.

В наших планах на 1999 год:

- вступительные экзамены в вузы (подборка наиболее типичных задач с решениями);
- материалы новых учебников (неопубликованных и недавно изданных);
- олимпиадные задачи по информатике (с решениями);
- компьютерная графика (методические материалы);
- компьютерная бухгалтерия (изучение основ бухгалтерии с применением компьютерного учета — продолжение);
- продолжение серии статей о том, как обустроить компьютерный класс

— и многое другое.

Москва, 121165, ул. Киевская, 24. Тел. 249-48-96. E-mail: inf@1september.ru

№ 12 (205) март 1999

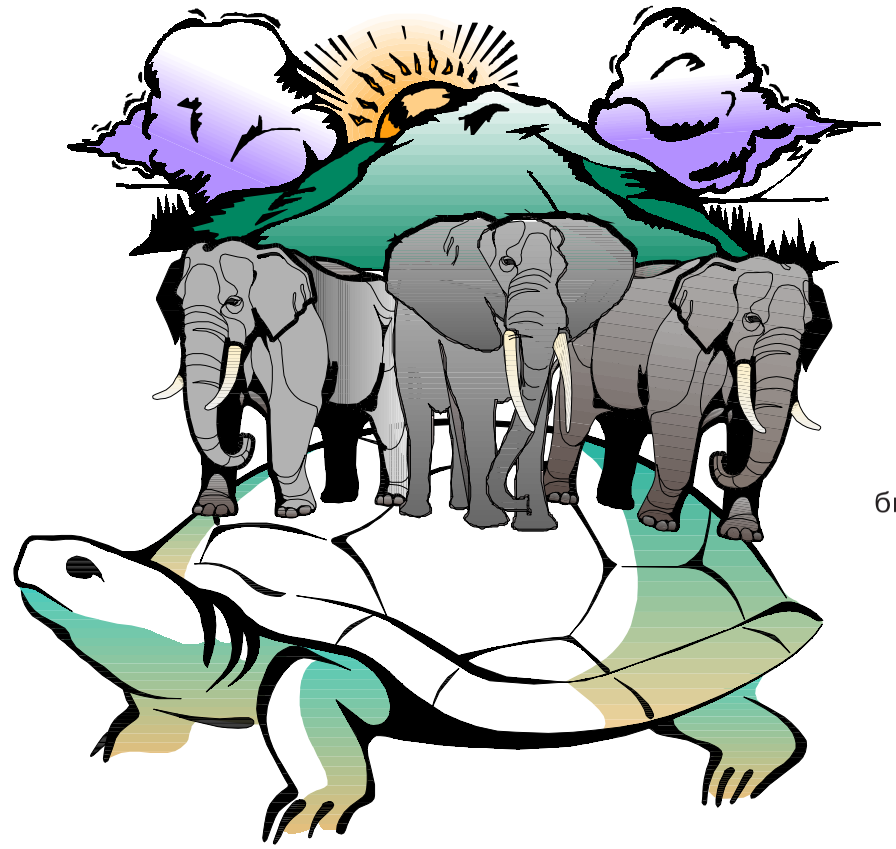
еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

12 лекций о том, для чего нужен школьный курс информатики и как его преподавать

Лекции 11, 12

А.Г. КУШНИРЕНКО,
Г.В. ЛЕБЕДЕВ



Введение, Лекции 1, 2, 3, 4, 5, 6, 7—8, 9, 10 были опубликованы в № 1, 3, 5, 6, 8, 10, 11/99

Выражаем признательность издательству “Дрофа” за содействие в подготовке публикации.

СОДЕРЖАНИЕ

Других сложных тем и разделов я в курсе не вижу. Соответственно, в почасовых разбивках в методпособии эти темы при уменьшении общего объема курса исчезают первыми. И в целом разбивки для разного числа часов полезны при составлении собственной расчасовки.

Ослабленный курс.

И, наконец, ко всему этому я еще добавлю несколько слов про ослабленный курс. Если вместо нормального общеобразовательного школьного курса надо просто “ликвидировать компьютерную безграмотность”, например, прочитать курс для “директоров предприятий”, то это тоже можно сделать. Первая часть сильно ослабляется и заканчивается перед “величинами” — здесь можно либо Робота “погонять”, либо порешать задачи с другими исполнителями (например, с Резчиком металла и пр.). Конечно, и эту часть надо облегчить — циклы с **если** внутри можно не давать, а решать простенькие задачки типа “довести Робота до стенки” или “прорезать дырку в металле”. Введение и начало курса (до величин) может быть изложено в ослабленном курсе примерно в объеме по трети урока на параграф в очень ослабленном виде.

Дальше все до конца главы 1 пропускается, на главу 2 тратится один час, в стиле “вот это — клавиатура, это — экран, это — дисковод”. Если есть возможность, то хорошо разобрать компьютер и показать, как он выглядит внутри. Чистый показ — никаких задач и никакого содержания. И, аналогично, из главы 3 берется также лишь беллетристическая часть про области применения ЭВМ. Можно показать, как работает текстовый редактор или как продаются билеты, но только внешне.

Все задачи и упражнения даются в таком курсе в начале (управление исполнителями), а курс в целом укладывается в 1—2 дня. Получается ослабленный миникурс, ликвидация компьютерной безграмотности.

Тем не менее даже после такого курса у человека возникает ощущение понимания, что же такое алгоритм. Это, пожалуй, единственное содержание, которое дает ослабленный курс, — представление об алгоритме и противопоставление программирования схеме непосредственного управления (первый “кит” курса).

Еще несколько слов о методике построения курса.

Помимо всего прочего, я хочу повторить, что методика нашего курса построена еще и на том, что, образно говоря, “навыки устной речи должны предшествовать изучению грамматики”, или — если привлечь пример из другой предметной области — сначала надо научиться складывать и вычитать (“набить руку”), а уж потом решать содержательные задачи. Применительно к информатике мы считаем, что сначала надо научиться писать простейшие алгоритмы и освоить алгоритмические конструкции (глава 1 учебника) и только потом приступать к решению содержательных прикладных задач (глава 3 учебника — “Применения ЭВМ”). При этом в первой части — при освоении основных алгоритмических конструкций и получении базовых навыков алгоритмизации — требуется, чтобы материал способствовал усвоению этих навыков и по возможности не содержал ничего лишнего. Соответственно, на старте у нас нет никаких информационных систем и ничего в этом духе.

Есть курсы, построенные по-иному. Например, у свердловчан [Гейн] с самого начала изложение строится на “жизненных” проблемах и примерах. И первым шагом любой такой деятельности считается построение математической модели исследуемого объекта или явления. Соответственно, с самого начала можно пытаться както мотивировать, почему решаются те или иные задачи.

У нас в первой главе абсолютно не обсуждается, ***почему*** надо решать ту или иную задачу. Образно говоря, мы учимся складывать, а не обсуждаем преимущества задачи складывания груш перед задачей складывания карандашей. Все вопросы создания мотивации в первой главе возлагаются на учителя. Учебник ничего такого не содержит.

Наконец, сама эта первая часть — освоение основных понятий и базовых навыков алгоритмизации — может быть условно разбита на два больших блока.

Первый блок — алгоритмические конструкции (§ 4—10 учебника): вспомогательные алгоритмы, циклы, ветвления. Этот блок можно назвать “программирование до величин”. Наличие такого блока, отработка основных алгоритмических конструкций на наглядных задачах нематематического характера — существенная особенность нашего курса. Как я уже сказал, можно либо изучать этот блок, следуя учебнику, либо менять порядок изложения, либо (особенно в сильных классах) сначала изложить сразу все конструкции, а потом долго решать задачи. Но какой бы ни была методика, важно, чтобы этот блок был, т.е. чтобы алгоритмические конструкции были освоены до величин.

Многие курсы начинаются с понятия величины и команды присваивания. Такой курс можно построить и на основе алгоритмического языка, но у него будет другая идеология, и, несмотря на единство языка, это будет совсем другой курс.

Второй блок — работа с величинами (§ 11—15). Здесь рассматриваются понятия величины, присваивания, различные типы величин (числовые, логические, табличные, литерные), обмен информацией между алгоритмами. Этот блок наиболее близок к традиционным курсам, и задачи для него можно найти практически в любой книге по программированию.

Список литературы

[Авербух] *Авербух А.В., Гисин В.Б., Зайдельман Я.Н., Лебедев Г.В.* Изучение основ информатики и вычислительной техники. М.: Просвещение, 1992. С. 302.
[ПдМ] *Кушниренко А.Г., Лебедев Г.В.* Программирование для математиков. М.: Наука, 1988.
[Гейн] *Гейн А.Г., Житомирский В.Г., Линецкий Е.В. и др.* Основы информатики и вычислительной техники. М.: Просвещение, 1991.
[Ершов] Под ред. *А.П. Ершова, М.В. Монахова.* Основы информатики и вычислительной техники. Ч. I и II. М.: Просвещение, 1986.
[ЗК] Знакомьтесь: компьютер. М.: Мир, 1989.
[Вейценбаум] *Дж. Вейценбаум.* Возможности вычислительных машин и человеческий разум. М.: Радио и связь, 1982.

на ЭВМ домашнее задание, после чего переходит к следующей теме. Если параграф сложный (типа § 9), то при таком подходе его просто разбивают на две части, говорят, что здесь две темы, а расчасовка остается стандартной — 1/1.

Интенсивный курс. Крайним выражением таких подходов является интенсивный курс, когда учитель вообще ничего не рассказывает на уроках. Ученикам на дом задается прочесть следующий параграф и подготовить какие-то упражнения. (Учебник это, вообще говоря, позволяет — т.е. достаточно задать школьникам прочитать материал, который написан достаточно просто.) Поэтому в интенсивном курсе говорят так: "На дом: решить то-то и то-то, прочесть очередной параграф или очередную тему, часть параграфа". Когда школьники приходят на урок, их усаживают за ЭВМ решать то, что им задавали на дом, а преподаватель отвечает на вопросы. И так урок за уроком — школьники работают на ЭВМ, а все остальное делают дома сами. Изредка по мере надобности учитель может проводить не урок, а своего рода консультацию, отвечая на накопившиеся по всему прошлому материалу вопросы или обращая внимание учеников на какие-то нюансы, которые они могли пропустить при чтении.

В таком интенсивном курсе все уроки проводятся на ЭВМ, изложения материала как такового вообще нет, а школьникам говорят: "Читайте учебник". Изредка устраиваются консультации для того, чтобы уровень всех подогнать до нужного. Мы обычно такую форму курса называем режимом практикума, поскольку школьники все время работают на ЭВМ.

Из других форм интенсивных курсов я отмечу только одну, о которой уже говорил, — это прохождение всей начальной части до "величин" путем "шоковой терапии", когда школьникам за один урок излагаются все управляющие конструкции, а потом начинается практика на ЭВМ. В отличие от последовательного изложения материала в учебнике, при "шоковой терапии" значительная часть учебника (теоретический материал) дается на одном уроке, а потом начинается решение задач из всего этого раздела. А уж школьники сами должны разбираться, какие им конструкции использовать и как решать задачи.

Дело в том, что последовательное и неторопливое — шаг за шагом — введение алгоритмических конструкций не всегда оказывается удачным. Если класс сильный, то ученикам может стать скучно. Тогда лучше изложить все управляющие конструкции на первом уроке, а потом уж их обсуждать и решать разные задачи. Так одним залпом можно пройти весь материал до "величин". Это хорошо работает, если класс достаточно сильный — существенно экономится время и сохраняется интерес. Зато потом, в третьей главе, можно дать больше применений ЭВМ.

И, наконец, последнее. Конечно, учитель вовсе не обязан выбирать какой-то один стиль и строго его придерживаться. Вы вполне можете, например, пройти управляющие конструкции методом "шоковой терапии", устройство ЭВМ — вообще в беллетристическом безмашинном стиле, что-то еще (скажем, раздел про величины) — в режиме практикума и т.д. В машинном курсе полезно сходить на экскурсию в агентство по продаже билетов, а потом попытаться составить информационную модель этой системы. Все эти стили, методики, возможности построения курса можно комбинировать между собой в соответствии с вашим ощущением класса, в соответствии с тем, как вы считаете нужным работать с вашими учениками. Другими словами, смотрите на все это как на возможности, из которых вы можете выбирать, набирать и строить свои собственные курсы.

Я, например, считаю, что даже в режиме практикума учителю следует периодически проверять тетради, а в чрезвычайно машинном курсе полезно с учениками говорить, писать что-то в их тетрадях и т.п. Все можно комбинировать, строить и искать что-то свое. Ведь это всего лишь набор возможностей и методов, из которых можно выбирать.

Сложные части учебника.

Теперь я хочу обрисовать наиболее сложные части учебника (они же первые кандидаты на пропуск в слабых классах или при преподавании в 7—8-х классах), а также то, что с ними связано. На мой взгляд, в целом в учебнике есть ровно две относительно сложные темы:

а) Методы алгоритмизации (§ 16) и наиболее математизированный § 25, а также частично § 26, поскольку в нем используются рекуррентные соотношения, выражающие следующее состояние парашютиста через предыдущее. Если § 16 и § 25 опустить, то § 26 надо либо тоже опускать, либо несколько переделать, чтобы не использовать рекуррентные соотношения. Эту часть, я считаю, в 7—8-х классах следует пропускать безусловно.

б) Рассмотрение алгоритмов как информации (кодирование алгоритмов величинами, компиляция, интерпретация и пр.). Кодирование стартует в § 20 (буквы "П", "Н", "А", "В", "К", понятие компиляции и интерпретации и пока не называемая так информационная модель алгоритма). Это используется далее в § 21. Если вы всю главу 2 пропустили, то можете либо ввести кодировки сами по себе в нужных местах (они простые, и трудности не должно быть), либо — если вы считаете материал сложным или не хотите в него углубляться — пропустите эту часть § 21. Здесь достаточно материала, скажем, с тем же кинозалом, транспортной сетью и пр. — есть что изучать и без кодирования алгоритмов. Сложность кодирования алгоритмов скорее психологического плана и связана исключительно с тем, что мы начинаем рассматривать алгоритмы как информацию и писать другие алгоритмы, которые эту информацию должны обрабатывать. Такой метапереход — алгоритмы выполнения закодированных алгоритмов.

Лекция 11. Применения ЭВМ

Теперь, усилив наш алгоритмический инструментарий понятиями информационной модели и информационной модели исполнителя (конструкция **исп...кон**), мы наконец можем заняться собственно применениями ЭВМ, которым посвящены § 23—27. Наиболее содержательными из них являются § 23, 24 и 26. Причем § 23 — самый содержательный и самый важный: в нем ярче всего показано применение введенного нами только что понятия исполнитель при решении конкретных прикладных задач. Да и материал — информационные системы — достаточно простой и благодатный.

§ 23. Информационные системы

Информационные системы — это системы, предназначенные для хранения, поиска и обработки той или иной информации. Как и всюду в этой части (§ 23—27), параграф начинается с беллетристики, с описаний конкретных систем.

Первым приводится описание системы "Экспресс" для продажи железнодорожных билетов с указанием реальных данных (1990 года) про количество задействованных мегабайтов памяти, число продаваемых за сутки билетов и пр. Эти фактические сведения, как мне кажется, достаточно интересны (во всяком случае, я с большим интересом все это узнал, когда мы получали информацию в процессе написания учебника). Узнать, сколько памяти нужно для представления информации обо всей железнодорожной сети и обо всех поездах страны, — это достаточно интересно.

Итак, в начале параграфа описываются система "Экспресс", система управления конвейером волжского автозавода, система учета междугородных переговоров и — чуть-чуть — базы данных. Естественно, все это — очень компактно.

А потом начинается содержание, т.е. изложение методов представления и обработки информации в описанных системах. Конечно, "настоящую" информационную систему нам здесь не описать — ведь это десятки и сотни тысяч строк на алгоритмическом языке. Но мы воспользуемся принципом "корыта" (а также нашим методическим приемом "очистки от всего лишнего") и покажем школьникам учебные информационные системы, очень простые, но поясняющие, как могут быть в принципе реализованы описанные выше системы ("объясним, почему оно плавает"). В параграфе рассматриваются две такие учебные системы. Первая — прямой учебный аналог "Сирены" — системы продажи билетов.

ЛИРИЧЕСКОЕ ОТСТУПЛЕНИЕ. При изложении материала младшеклассникам (что вполне возможно) требуются образные постановки задач. И если задачи про Робота могли начинаться словами "В некотором царстве, в некотором клетчатом государстве жил-был Робот", то материал про применения ЭВМ мы планировали изучать на примере сказочной страны Лилипутии, где все очень маленькое. В этой сказочной стране живет меньше тысячи лилипутов (но у каждого есть телефон); там всего два города, между которыми совсем недавно лилипуты проложили железную дорогу и пустили один самодвижущийся купейный вагон (*рис.* 13).

Рис. 13

Лилипутия, однако, очень прогрессивная страна, и лилипуты, чтобы идти в ногу со временем, решили компьютеризировать свою жизнь — продавать билеты в свой единственный вагон, учитывать, у кого какой номер телефона, и решать другие задачи с помощью компьютеров. Но на беду лилипуты пока не проходили информатику, поэтому они решили позвать нас (точнее, школьников) и попросили написать им все их лилипутские информационные системы. И вот теперь наша задача — помочь лилипутам, реализовать для них на алгоритмическом языке информационную систему для продажи билетов "Вагон" (п. 23.5 учебника), систему для учета абонентов лилипутской телефонной сети "Телефонная книжка" (п. 23.6 учебника) и т.п.

В порядке разрядки такое изложение может быть проведено и в старших классах. Но для старшеклассников мы пустили наш единственный вагон без паровоза и без других вагонов по вполне взрослому маршруту Москва — Петушки. Итак, простая учебная система "Вагон", которая должна уметь продавать билеты в этот вагон, узнавать, продано или не продано место, продать два места в одном купе и т.д. Маленькая учебная модель, перефраз системы "Экспресс".

Естественно, раз мы собираемся продавать билеты, то нам нужны информационная модель вагона (для класса задач, связанного с продажей билетов), а также исполнитель, который будет хранить эту информацию и выполнять команды типа “продай место”, “верни билет” и пр. Все это вместе в соответствии с § 22 мы назовем исполнителем Вагон. Модель, которая тут получается (п. 23.5), оказывается даже проще, чем модель кинозала. В отличие от кинозала (и именно для того, чтобы показать, что проданные места можно задавать по-разному), здесь вместо логической таблицы используется целочисленная таблица **M[1:36]** (**M** — от слова “место”: **M[i]** — это *i*-е место). Если **M[i] = 1** — место продано, если **M[i] = 0** — свободно.

Обратите внимание, что в учебнике не приведен фрагмент задания начальных значений таблицы **M[1:36]**. Вы можете попросить школьников либо написать соответствующий фрагмент сразу после описания величины **M**, либо реализовать команду (алгоритм) “начать работу” (некоторым больше нравится название “очистить вагон”), которая должна устанавливать, что все места свободны.

Из других содержательных элементов обратите внимание на **дано** алгоритма А102. Записанное в **дано** условие **1 ≤ i ≤ 36 и M[i] = 0** — абсолютно формальное и содержательное условие на алгоритмическом языке. Это значит, что всякий раз перед началом исполнения тела алгоритма А102 “учесть продажу места” ЭВМ проверит, что условие **1 ≤ i ≤ 36 и M[i] = 0** истинно, т.е. что *i*-е место существует и свободно. Если условие окажется невыполненным, то, значит, мы где-то при написании основного алгоритма допустили ошибку — возникнет отказ, и надо будет менять программу.

Все остальное здесь — проще некуда. Такими алгоритмами можно было заниматься с самого начала, они вполне доступны даже младшеклассникам. Вы можете как угодно разнообразить их, требуя продать два места в одном купе или место на нижнюю полку и т.п. Соответствующие задачи сформулированы в упражнениях, и — вспомните “черепаху” курса — именно в процессе решения таких задач школьники должны действительно (“через делание”) усвоить и понятие исполнителя, и простейшие методы представления и обработки информации в информационных системах. Вы можете легко повысить сложность этой задачи, усложняя представляемый объект, например, предусмотрев возможность бронировать места, введя разные типы вагонов (плацкартные, общие, вагон-ресторан и пр.), промежуточные станции и возможность продавать билеты на промежуточных станциях, и т.п.

Реальная система продажи билетов “Сирена” достаточно сложна, но в принципе устроена так же. Изучение учебной системы “Вагон” и решение задач в рамках этой системы формирует достаточно глубокое и адекватное представление о том, как устроены настоящие системы. Именно это я и имел в виду, когда говорил, что наша задача — показать не только ***что*** могут делать ЭВМ, но и ***как*** они могут это делать.

И еще один повтор — обратите внимание, насколько естественна здесь формулировка “реализовать исполнителя”, насколько естественна реализация алгоритмов не самих по себе, а в рамках заданного исполнителя. В реальной жизни крайне редко требуется написать один отдельно взятый алгоритм. Как правило, задачи формулируются как задачи создания программных систем (исполнителей) с некоторым набором команд.

Вторая учебно-информационная система учебника — система “Телефонная книжка” (п. 23.6). Если не говорить о Лилипутии, то проблему можно сформулировать так: реализовать систему для хранения информации об абонентах телефонной сети в небольшом поселке (меньше 1000 телефонных номеров). Это значит, что надо уметь отвечать на вопросы типа “Какой телефон у Иванова Петра Ивановича?” или, наоборот, “Кому принадлежит номер 555?”. Надо уметь получить свободный номер для установки новому абоненту или поменять номер, если он уже кому-то надоел. Такой вот обычный набор действий.

Приведенная в учебнике информационная модель состоит из целочисленной величины **n**, в которой хранится число телефонов в поселке, и литерной таблицы **Ф.И.О.[1:999]**, в которой хранятся фамилии, имена и отчества абонентов. В тексте параграфа можно было бы обойтись и без величины **n**: эта величина добавлена в модель, чтобы удобно решался ряд упражнений. Алгоритмы исполнителя, реализующие команды установки телефона и смены номера, также достаточно просты, и вы их посмотрите сами.

Алгоритмы поиска.

Я же хочу обратить ваше внимание на алгоритм “справка о номере телефона” и на вызываемый из него вспомогательный алгоритм “поиск абонента”. “Поиск абонента” — именно то, ради чего приведен в учебнике весь этот исполнитель. Дело в том, что задача поиска является очень частой задачей в информатике. Настолько частой, что вначале мы даже хотели посвятить таким задачам отдельный параграф в алгоритмической части (в главе 1). Потом, однако, мы решили ограничиться алгоритмом поиска в рамках темы “Информационные системы”— это алгоритм А109 и ряд упражнений после параграфа.

Напомню, что информационные системы — это системы, предназначенные для хранения, изменения и поиска информации. Как хранить большие объемы информации, мы уже знаем — с помощью таблиц и информационных моделей. Изменение информации достаточно продемонстрировано в системе “Вагон”. А вот поиск — одна из ключевых функций информационных систем — демонстрируется (в простейшем виде) именно алгоритмом “поиск абонента” (А109).

Алгоритм А109 учебника (с. 192) — простейший. Это так называемый “последовательный” поиск, когда мы просто перебираем элементы таблицы один за другим, пока не обнаружим нужный. Алгоритмы поиска —

1. ***Безмашинная*** — компьютеры вообще недоступны, организовать уроки с использованием ЭВМ нет никакой возможности. Этот случай требует от учителя наибольших усилий, потому что он должен придумывать “черные ящики”, разыгрывать спектакли и иным образом компенсировать отсутствие ЭВМ. Учитель сам (за счет своей личности) должен сделать предмет интересным. Для учителя это наиболее тяжелый курс.

Но важно понимать, что с точки зрения методики преподавания такой безмашинный курс ничем не хуже математики. Вы вполне можете излагать новый материал, задавать задачи, проверять домашние задания, проводить самостоятельные, контрольные и т.д., то есть делать все то, что делают учителя математики. Другая предметная область, но тот же самый стиль, та же методика.

Конечно, безмашинный курс менее интересен и, соответственно, менее эффективен, чем машинный. Когда есть интерес, учить лучше. Но, повторю, даже в этом случае учиться нужно и есть чему и математика является хорошим аналогом. В курсе математики ученики тоже решают задачи, пишут в тетрадях, сдают их, а учитель проверяет. Конечно, при этом учителю надо сидеть с тетрадками, проверять и т.д. Методика и стиль преподавания безмашинного курса информатики почти не отличаются от преподавания математики. Во всяком случае тот курс и тот учебник, про который я рассказываю, вполне можно излагать в таком стиле, задавать задачи, проверять решения, устраивать самостоятельные и контрольные, проверять, выставлять оценки, задавать новые задачи и т.п.

2. ***Маломашинная***. Это такой промежуточный вариант, когда есть возможность эпизодически водить школьников куда-нибудь, грубо говоря, “на экскурсию”. Нельзя планировать определенные уроки на ЭВМ, но можно, скажем, раз в месяц посетить вычислительный центр. (У меня есть знакомые учителя, которые водили школьников даже в агентство по продаже авиабилетов, а потом на уроках обсуждали, как эти системы и машины работают, как построить информационную модель того, что они видели, и пр.)

Маломашинный курс практически ничем не отличается от безмашинного. Единственное, что для повышения мотивации иногда бывают экскурсии. Если при этом есть возможность что-то выполнить на компьютерах, то желательно, чтобы школьники заранее подготовились, решили какие-то задачи, которые они в ходе экскурсии выполнят.

Если удается выполнять алгоритмы на компьютерах регулярно — скажем, раз в месяц, то ***маломашинный*** курс фактически превращается в ***полумашинный***: можно месяц изучать материал в безмашинном математическом стиле, а потом проводить самостоятельную или контрольную работу с выходом на компьютеры.

3. ***Традиционная машинная.*** Слово “машинная” мы применяем в ситуации, когда учитель может планировать, какие уроки ему проводить на ЭВМ, а какие — без. Другими словами, когда учитель свободен в этом планировании. В маломашинном или полумашинном курсе компьютеры не очень доступны, и надо загодя договариваться о занятиях школьников. Машинным же мы называем курс, если компьютерный класс есть либо прямо в школе, либо рядом, если учитель может составить расписание, спланировать занятия на ЭВМ тогда, когда это нужно по логике преподавания. Именно для такого традиционного машинного курса приведены все почасовые разбивки в методическом пособии [Авербух].

4. ***Интенсивная.*** Наконец, есть еще курс, называемый интенсивным. Довольно много педагогов считает, что если компьютеры есть, то абсолютно все занятия надо проводить на ЭВМ. Обычно интенсивный курс проводится, когда компьютеры стоят прямо в классе и их в любой момент можно включить.

Я сейчас более подробно остановлюсь на машинном и интенсивном курсах (про безмашинный уже все сказано — есть материал, его можно задавать, спрашивать, проверять в тетрадях и т.д.).

Машинный курс. При изложении материала параграф за параграфом одной из лучших является, видимо, схема “лекция, семинар, практика”. По каждой теме одно занятие проводится в форме лекции — учитель задает простые задачи (проблемный подход), рассказывает, излагает новый материал и пр. Второе занятие — семинар, когда школьники без машин решают задачи, а учитель смотрит, подправляет и консультирует. Третье занятие — практика: на нем школьники уже решенные (и записанные в тетради) задачи набирают и выполняют на ЭВМ. Практики может быть не одно занятие, а два или три. Авербух, как правило, рекомендует схему 1/1/2 — “лекция, семинар, практика, практика”. Если материал сложный, то, соответственно, увеличиваются лекционные или семинарские часы.

Для такого традиционного курса в методическом пособии имеется не только почасовая разбивка, но и поурочное планирование для первых 24 уроков с указанием, какие упражнения надо решать на уроке, какие задавать на дом и пр. Конечно, это не догма, а всего лишь пример. Если вы преподаете курс в первый раз, это может оказаться полезным. Потом, по мере роста понимания и опыта, вы сможете поурочное планирование менять.

Кроме традиционной, имеются и другие формы проведения машинного курса. Например, есть подход, при котором любая тема занимает ровно два урока. На первом уроке учитель рассказывает новый материал и задает упражнения на дом, а на втором — принимает

которая бы учитывала количество транспорта на разных направлениях и соответствующим образом переключала светофор с целью оптимизации дорожного движения (и такие проекты существовали в действительности). Можно, однако, вместо этого построить дорожную развязку, чтобы исчезли и перекресток, и светофоры, и необходимость регулировать движение. Другой пример: переход на порошковые технологии изготовления металлических деталей делает ненужной чистовую обработку детали (и ЭВМ, управляющие такой обработкой).

Все сказанное здесь о роли ЭВМ как консерваторов существующих технологий — почти дословный пересказ из книги Дж. Вейценбаума "Возможности вычислительных машин и человеческий разум", которая приведена под номером 2 в списке рекомендуемой литературы на с. 224 учебника. Лично я считаю, что это — лучшая книга всех времен и народов о компьютерах, о месте компьютеров в мире и об их роли в развитии человечества. Хотя она написана довольно сложным языком, насыщена философией и потому читается очень медленно, я настоятельно рекомендую вам эту книгу прочесть. Я ее перечитывал раз десять — и каждый раз как новую, открывая для себя что-то, чего не заметил или не понял при предыдущем прочтении.

Конечно, есть технологии, которые без компьютера просто невозможны. Один пример такой технологии — томография — есть в учебнике. Очень яркий пример принципиально новых возможностей, которые может дать компьютер. Раньше при подозрении на опухоль мозга врач должен был вскрывать пациенту череп — нередко лишь для того, чтобы убедиться, что никакой опухоли нет. С появлением томографа стало возможным просто просветить череп и выяснить, что там внутри.

Но, с другой стороны, часто (и даже "как правило") компьютеры применяются для выполнения рутинных действий в рамках уже существующих технологий и в этом смысле ***тормозят*** научно-технический прогресс. Поэтому стремление просто "внедрить ЭВМ", превращение компьютеров из средства в самоцель чрезвычайно опасно.

Подпункты 2—4 пункта 28.3 менее фундаментальны и лишь демонстрируют, что ЭВМ не всемогущи и при их применении возможны ошибки и в алгоритмах, и в исходных данных — и к этому надо быть готовым. Более подробное изложение приведенных в учебнике и других примеров ошибок в работе ЭВМ можно найти в книге Майерса "Надежность программного обеспечения" [Майерс].

А вот заключительный подпункт 5 опять содержит чрезвычайно важное замечание, что компьютеры и их применения становятся частью нашей жизни. Поэтому вопрос о том, надо или не надо применять компьютеры при решении каких-то задач, должен в первую очередь решаться исходя из того, как изменится наша жизнь. В учебнике это не очень удачно названо "социальными последствиями". Более глубокое изложение материала вы можете найти также в книге [Вейценбаум].

* * *

На этом глава 3, а с ней и весь учебник заканчиваются.

Лекция 12

D. ЗАКЛЮЧЕНИЕ

D1. Методики преподавания курса

В методическом пособии [Авербух] приведены (с. 297—301) почасовые разбивки, рассчитанные на традиционный машинный курс. Считается, что в школе или рядом со школой есть кабинет и при желании учитель имеет возможность провести урок на ЭВМ. Общий объем курса может составлять 102, 68 или 32 часа — и показана рекомендуемая разбивка этих часов: сколько занятий в классе без ЭВМ и сколько — на ЭВМ для каждой темы.

По сути, эта разбивка отчасти отражает картинку, которую я рисовал (см.: Лекция 2, № 3/99, с. 7), хотя почасовая разбивка больше зависит от сложности, а столбики на картинке были интегральные — отражали и сложность, и важность одновременно. Важная или не важная тема — часы выделяются в зависимости от сложности, чтобы тема была нормально усвоена.

Вы можете использовать эти почасовые разбивки вместе с картинкой на с. 50 для понимания относительной ценности разного материала учебника — по мере уменьшения общего объема курса в нем остается время только на самый важный материал, а остальное постепенно выпадает. Если вы почему-либо не успеваете, не укладываетесь во времени, то оцените, что можно пропустить, а что надо обязательно оставить.

Повторю, что названные почасовые разбивки относятся к так называемому традиционному машинному курсу. "Традиционный" означает, что как учебник написан, так мы его и изучаем — параграф за параграфом.

Если говорить о более общем подходе, то можно выделить четыре основные методики преподавания курса.

это отдельная тема информатики, которой вполне можно посвятить полгода или год занятий на факультативе. Здесь же мы лишь обозначили эту тему.

Если вы желаете затронуть ее хоть чуть более глубоко, можете воспользоваться упражнением 4*б*, в котором требуется, чтобы абоненты в таблице Ф.И.О. были упорядочены не по номерам телефонов, а по именам, как в обычных телефонных книжках: сначала — все на "А", потом — на "Б" и т.д. В этом случае индекс таблицы уже нельзя использовать как номер телефона, и поэтому в соответствующей модели в упр. 4 появляется еще одна таблица, содержащая номера телефонов абонентов. В этой модели вы можете попросить учеников реализовать поиск абонента не последовательным перебором, а методом деления пополам: сначала взять середину заполненной части таблицы Ф.И.О. и посмотреть, левее или правее этой середины находится фамилия искомого абонента (ведь элементы таблицы упорядочены по алфавиту!), потом, соответственно, взять середину левой или правой части и т.п. Этот — бинарный — алгоритм поиска, а также разные другие алгоритмы поиска вы можете найти почти в любых книжках по программированию, в частности, в [ПдМ].

Заключение.

В упражнениях после параграфа есть стандартные, требующие дописать или изменить что-то в исполнителях Вагон и Телефонная книжка. Наиболее интересным из них, по-моему, является упр. 5. Но я хочу обратить ваше внимание на мое любимое упр. 6 на с. 193 учебника — хранение информации о владельцах автомобилей и поиск всех владельцев белых "Волг", в номере которых есть буква "М" и цифра "7". У школьников тут возникает масса всяких гипотез (шпионских и прочих) о происхождении задачи, и поэтому обычно ее решают с большим интересом. Вы можете и сами формулировать эту задачу вполне жизненным образом: "Поступила информация, что напротив нашей школы белая "Волга" сбила человека и скрылась. Свидетели запомнили только, что в номере есть буква "М" и цифра "7". Надо найти все такие машины, чтобы их проверить".

Итак, резюме. Параграф состоит из двух частей. Вначале идет беллетристика (в хорошем смысле этого слова), т.е. описание конкретных областей применения информационных систем, демонстрирующих, ***что*** могут ЭВМ. Потом, во второй части, излагаются учебные информационные системы, в очень упрощенном виде показывающие, ***как*** хранится, изменяется, ищется информация в информационных системах. С алгоритмической точки зрения материал здесь очень простой — по-моему, гораздо проще цикла **пока** — и вполне доступен для изучения начиная с 6—7-го класса.

Повторю, что § 23 является фундаментальным с точки зрения описания применений ЭВМ: он посвящен непосредственно хранению, изменению, переработке и поиску информации, т.е. практически предмету информатики (вспомните определение информатики на с. 3 учебника). По этой же причине он наилучшим образом иллюстрирует введенное в § 22 понятие исполнителя как конструкции алгоритмического языка.

Поэтому я вам рекомендую в случае нехватки времени лучше пропустить остальные параграфы про применения ЭВМ (§ 24—27), но в полном объеме пройти § 23.

§ 24. Обработка текстовой информации

При работе в системе КуМир нет необходимости пользоваться каким-то отдельным текстовым редактором — соответствующие возможности встроены непосредственно в систему КуМир.

Но существуют системы, в которых создание текста программы отделено от его обработки и выполнения. Кроме того, многие люди используют ЭВМ для редактирования обычных текстов, не имеющих отношения ни к каким алгоритмам и программам, например, для ввода и изменения писем друзьям или текстов научных статей по географии. Говорят, что (если исключить компьютерные игры) редактирование текстов составляет чуть ли не 80% всех применений ЭВМ в мире. Поэтому наш второй (после "чисто" информационных систем) параграф посвящен именно обработке текстовой информации.

Естественно, сначала идет беллетристика под названием "Системы обработки текстов", где говорится, что тексты можно обрабатывать на ЭВМ, и это хорошо. Что можно исправить только неправильное место и напечатать весь текст заново. И т.д. Поскольку все вы работали с теми или иными системами обработки текстов (текстовыми редакторами), вам легко рассказать школьникам об используемой вами системе, показать ее и дать поработать. Это вещи, известные всем.

Задача, которую мы перед собой поставили, — показать, как устроен внутри, как реализуется современный экранный текстовый редактор. Создать у ребят представление, что происходит, когда они нажимают на клавиши, почему текст на экране начинает смещаться и пр. Здесь мы снова от вопроса, ***что*** может делать текстовый редактор, переходим к вопросу, ***как*** он это делает.

Для этого вначале (п. 24.2) вводятся понятия "текст", "окно", "курсор". А потом (п. 24.3) рассматривается реализация учебного редактора текстов.

Я не буду повторять материал учебника. Отмечу лишь, что "курсор" в п. 24.2 соответствует системам обработки текстов, в которых позиция в тексте (курсор) отмечается квадратиком поверх символа (как в КуМире и МикроМире) или черточкой под символом. Если в вашей системе обработки текстов курсор — вертикальная черточка ***между*** символами, то в п. 24.2 и 24.3 следует внести соответствующие изменения либо продемонстрировать школьникам систему, в которой "курсор" будет совпадать с "курсором" из п. 24.2 учебника.

Кроме того, я обращаю ваше внимание, что в п. 24.2 ошибочно указана ссылка на фото 18 вклейки. Здесь должна быть ссылка на фото 5 вклейки, на котором изображено много "окошек" с разными текстами. Впрочем, это — фото, сделанное с экрана Ямахи. Если есть возможность, то вместо него лучше продемонстрировать любую современную многооконную систему, например, Windows.

После введения всех трех понятий в п. 24.3 разбирается учебная модель редактора текстов, рассчитанная на 100 символов в строке и текст максимум из 200 строчек. Эти параметры, впрочем, без труда можно заменить на другие. Для хранения и представления текста, а также окна и курсора нам необходимо построить информационную модель этих понятий, задать их величинами алгоритмического языка.

В качестве информационной модели текста на с. 195 учебника задается символьная таблица **`T[1:200, 1:100]`**, а также целочисленная величина **`N`**, в которой будет храниться реальное число строк в тексте, поскольку текст может содержать и меньше чем 200 строк.

Информационная модель окна — это две величины **`цел Y0, X0`**, содержащие смещение окна от верхнего (**`Y0`**) и левого (**`X0`**) краев текста. (Ширина и высота окна считаются фиксированными.)

Информационная модель курсора — также две величины — **`цел Yk; Xk`**, задающие координаты курсора внутри окна, т.е. смещение курсора от верхнего (**`Yk`**) и левого (**`Xk`**) краев окна.

Таким образом, **`Y0`** — это количество строк над окном, **`X0`** — количество символов в строке левее окна, **`Yk`** — номер строки с курсором в окне (считая сверху), **`Xk`** — номер позиции курсора в окне по горизонтали (считая по символам слева).

Все это, вместе взятое, — **`сим таб T[1:200, 1:100], цел N, Y0, X0, Yk, Xk`**, — и есть информационная модель текста, окна и курсора.

Теперь, имея информационную модель, мы можем начинать писать алгоритмы исполнителя "Простейший редактор". Например, перемещение курсора вправо на один символ или алгоритм вставки пустой строки. А также любые другие обычные для текстового редактора команды.

В учебнике реализованы алгоритмы (команды) "вниз на строку" и "удалить символ". Еще некоторое количество команд сформулировано в упр. 1 (с. 198 учебника).

Обратите внимание, что приведенные в учебнике алгоритмы меняют только информационную модель текста, окна, курсора. Чтобы изменения в тексте отобразились на экране компьютера, нужно провести отдельную работу — составить соответствующий алгоритм (алгоритмы А112 и А113 учебника). Здесь приведено простейшее решение — переизображение всего экрана целиком. Обычно в экранных редакторах меняют только часть экрана, в зависимости от того, какое действие было выполнено. Мы же — для простоты — написали общий алгоритм изображения экрана целиком. Этого хватает для демонстрации принципов и может оказаться достаточно для реальной работы, если у нас быстрый компьютер. Если же компьютер медленный, то при выполнении каждой команды редактора будет видно, как переизображается экран.

Конечно, написать команды **`вниз на строку`**, **`удалить символ`** и даже алгоритм переизображения экрана еще недостаточно. Ведь мы хотим реализовать текстовый редактор целиком. Поэтому надо еще написать основной алгоритм, который будет ожидать нажатия человеком на клавишу; после нажатия — анализировать, какая именно клавиша нажата, и в зависимости от клавиши вызывать соответствующую команду (например, **`удалить символ`**); а после всего этого — еще и переизображать состояние информационной модели на экране (вызывать алгоритм А112 "переизобразить экран и курсор").

Этот основной алгоритм в учебнике не приведен, а лишь описан (примерно как я его сейчас описал) на с. 197. Мы не стали приводить его по следующим причинам. После ввода клавиши в этом алгоритме идет огромный выбор, в котором последовательно перебираются разные коды клавиш, и если была нажата клавиша с данным кодом, то вызывается соответствующая команда редактора. Например, если код нажатой клавиши равен 12 (это код стрелки влево), то вызывается команда **`влево на символ`**. Другими словами, здесь должны быть выписаны все коды клавиш. А они на разных компьютерах разные и в любом случае требуют каких-то пояснений. Поэтому мы основной алгоритм приводить не стали.

> ПРОГРАММНОЕ ОТСТУПЛЕНИЕ. В комплект поставки системы КуМир входят и Вагон, и Телефонная книжка, и учебный экранный редактор. Поэтому, хотя в самом учебнике текст основного алгоритма не приведен и школьникам его составлять не нужно, посмотреть этот текст на экране в системе КуМир они могут. При реальном добавлении новой команды в редактор им надо будет подправить и основной алгоритм. Таким образом, внутренности экранного редактора будут им предъявлены полностью. С этим можно разобраться, понять, как редактор работает, и даже самому добавить в редактор те или иные команды. В частности, составив очередной алгоритм из упр. 1 (с. 198 учебника) и добавив его в исполнитель "Простейший редактор", школьник может превратить редактор, который только что не понимал, например, команду **`удалить конец строки`** (упр. 1*б*), в редактор, понимающий эту команду, и затем немедленно проверить, как она будет работать, и пр.

§ 25. Научные расчеты на ЭВМ

Этот параграф информационных моделей не содержит и посвящен исключительно методам алгоритмизации при решении конкретных вычислительных задач.

Полученные в результате вычислений значения запоминают, потом уменьшают шаг вдвое, выполняют алгоритм снова и запоминают вычисленные значения. Если разница между полученными таким способом результатами невелика, то считается, что шаг достаточно мал. Если же разница велика, то шаг уменьшают еще вдвое и повторяют вычисления.

При таком подходе к выбору шага по времени:

— с одной стороны, шаг очень быстро уменьшается, поэтому мы очень скоро достигнем "достаточно малого" для данной задачи шага;

— с другой стороны, все "предварительные" вычисления, вместе взятые, занимают не больше времени, чем одно "итоговое" вычисление, т.е. мы тратим максимум в два раза больше времени, чем если бы провели только одно "итоговое" вычисление (если бы из каких-то иных соображений нам было точно известно, каким должен быть шаг по времени).

Пункт 26.5 про выбор шага по времени — это некий заключительный нюанс, заключительный штрих к тому, как на самом деле моделируются процессы на ЭВМ. Этот пункт в принципе можно опустить. Проблема выбора шага по времени останется нераскрытой, но это уже менее важная тема, чем сам метод дискретизации непрерывных процессов. Что же касается сути метода дискретизации, то тут ничего потеряно не будет.

§ 27. Компьютерное проектирование и производство

Как и все остальные параграфы про применения ЭВМ, этот параграф начинается с беллетристической части: использование ЭВМ для подготовки чертежей; замена реальных экспериментов (когда, например, автомобиль с чучелом вместо человека на всей скорости врезается в неподвижную стену) на моделирование эксперимента на ЭВМ; использование ЭВМ для управления станками с программным управлением и т.п.

> МЕТОДИЧЕСКОЕ ОТСТУПЛЕНИЕ. Преобразование Чертежника в устройство для резки металла (п. 27.3 учебника) идейно восходит к исполнителю Резчик металла из нашего вузовского учебника [ПдМ], откуда и взяты команды **`поднять резак`**, **`опустить резак`** и пр. Хотя Резчик металла и проще Робота, он ввиду его очевидной "производственности" и кажущейся непридуманности лишен того ореола "игрушечности", который так мешает нормально работать с Роботом.

Поэтому если вам не удалось преодолеть "игрушечность" Робота, то вы можете попробовать простые алгоритмы проработать с Резчиком металла (систему команд и примеры алгоритмов можно найти в [ПдМ]), а к Роботу вернуться после прохождения команд циклов.

В содержательной части параграфа — п. 27.5 — изложена лишь простейшая информационная модель (М20) поверхности сложной формы, например, капота или крыла автомашины. В [ПдМ] и в учебниках [Ершов] есть картинки, изображающие различные детали автомобиля, представленные методом конечных элементов (т.е. треугольниками, четырехугольниками и пр.). В учебнике ни картинок, ни алгоритмов обработки таких моделей не приведено. Как я уже говорил (см. с. 260), соответствующий материал просто не уместился в отведенный размер учебника. Мы лишь привели модель М20 и сформулировали несколько упражнений, из которых упр. 1 и 2 мне кажутся обязательными, если эту тему вообще проходить.

§ 28. Заключение

Заключение начинается с описания еще нескольких областей применения ЭВМ: электронные деньги, штриховые коды, компьютерные игры и проч. Все это можно прочесть в газетах, в книжках и т.д. Материал является беллетристическим и изложен в том же стиле, что и беллетристические части § 23—27.

Основное содержание параграфа сосредоточено в п. 28.3 "Ошибки в применениях ЭВМ" на страницах 215—217. Этот материал кажется мне чрезвычайно важным, особенно с точки зрения формирования общего мировоззрения учеников. И самое главное, пожалуй, написано в подпункте 1 п. 28.3, где говорится, что ЭВМ не столько развивают научно-технический прогресс, сколько тормозят его. Компьютеры настолько окружены ореолом "современных технологий", ореолом чего-то новейшего, что их реальная роль в научно-техническом прогрессе осознается очень редко. Дело в том, что мы всегда передаем ЭВМ рутину, только то, что уже осознано, то, что делать несложно, а все наше творческое начало остается при нас.

Конечно, замечательно, что мы тем самым можем освободиться от рутинных дел и заняться творческими задачами. Но есть и вторая сторона медали. Когда мы перестаем справляться с потоком рутины, у нас есть два пути. Первый — попытаться осознать, почему рутины так много, и изменить жизнь таким образом, чтобы ее не было. Это и есть развитие, прогресс. Другой путь — поставить ЭВМ и заставить ее в больших объемах перерабатывать всю рутину, никак не меняя нашу жизнь по существу. В этом смысле ЭВМ всегда являются своего рода консерваторами существующего состояния дел. Когда человек не справляется с огромными потоками рутины, применение ЭВМ позволяет некоторое время выжить, не меняя существа дела. Если говорить о привнесении чего-то принципиально нового, то ЭВМ этому не способствуют — они сохраняют уже существующие технологии и делают потребность в изменениях менее насущной.

В учебнике приведено несколько примеров таких ситуаций. Например, при усилении транспортного потока через перекресток можно пытаться поставить ЭВМ,

МЕТОДИЧЕСКОЕ ОТСТУПЛЕНИЕ. Здесь используется ссылка на метод рекуррентных соотношений из § 16. Если при прохождении курса вы § 16 пропустили и метод рекуррентных соотношений не проходили, то алгоритм А120 все равно можно изложить (он очень прост), но вместо ссылки на метод рекуррентных соотношений надо соответствующий материал (про "исчезновение индексов") повторить здесь явно.

ЛИРИЧЕСКОЕ ОТСТУПЛЕНИЕ. Несколько слов для вас как учителей, которые должны знать больше, чем ученики. Если нарисовать графики функций $h(t)$ и $v(t)$, задающих высоту и скорость парашютиста в зависимости от времени, то для реального процесса падения это будут плавные (непрерывные) кривые. В нашей дискретной модели эти кривые ***аппроксимируются*** (т.е. задаются приближенно) с помощью ступенчатых функций, которые постоянны на протяжении шага по времени, потом скачком меняются и опять постоянны на протяжении следующего шага (*рис.* 14). С математической точки зрения "дискретизация" — это замена непрерывной функции такого рода ступеньками.

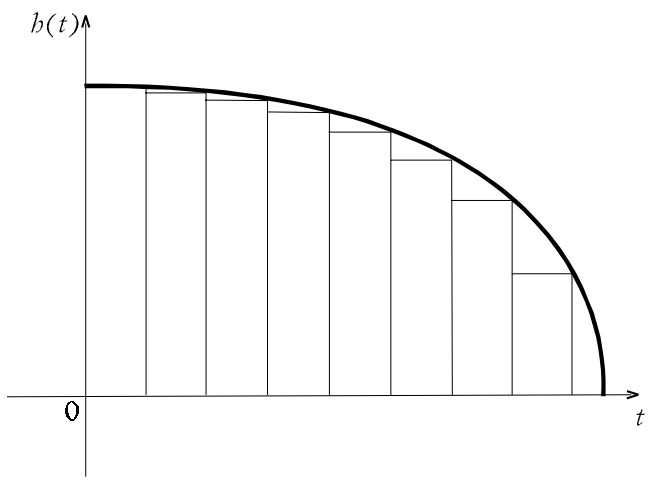


Рис. 14

На этом собственно изложение метода дискретизации непрерывных процессов заканчивается, и остаток параграфа посвящен двум небольшим уточнениям (разъяснениям), касающимся этого метода, а именно: сравнению приближенного и точного решений (п. 26.4) и выбору величины шага по времени (п. 26.5).

Сравнение приближенного и точного решений.

Если сопротивление воздуха отсутствует ($k = 0$), то получается обычная задача из школьного курса физики, для которой можно написать точные формулы, вычислить все положения "парашютиста" точно. В то же время мы можем подставить $k = 0$ в наш алгоритм А120 и посмотреть, насколько приближенное решение, полученное методом дискретизации непрерывных процессов, отличается от точного. Именно это и проделано в п. 26.4.

Мы, конечно, понимаем, что, поскольку мы разбили непрерывный процесс на дискретные состояния, ответ (например, через сколько времени тело ударится о землю) получится приближенный. Но насколько мы при этом ошибемся — на минуту или на год? Надо же хотя бы возможный порядок ошибки себе представлять. Именно для этого и рассматривается случай $k = 0$ — это единственный случай, для которого мы знаем точные формулы.

Самый главный вывод, который надо сделать из этого пункта: при уменьшении шага по времени dt, например, с $dt = 1$ с до $dt = 0{,}5$ с наше приближенное решение стремится к настоящему. Уменьшение шага по времени приводит к более точным решениям.

И последнее замечание. Конечно, сравнение с точным решением носит чисто учебный характер. В реальных задачах, да и в нашей начальной задаче никакое "точное" решение неизвестно. Мы потому и моделируем эти процессы на ЭВМ, что не знаем для них точных формул. Как я уже говорил, мы пытались всюду в учебнике приводить в качестве примеров только такие задачи, которые не сводятся к задачам из школьной математики или физики, которые без информатики, без компьютеров в каком-то смысле "не решаются" вообще.

Выбор шага по времени.

После осознания, что уменьшение шага по времени приводит к более точным решениям, немедленно возникает вопрос: почему бы сразу не взять какой-нибудь маленький шаг, например, $dt = 0{,}000001$, чтобы решение получилось достаточно точным?

Здесь надо отметить два момента.

Во-первых, в каких-то задачах и шаг $dt = 0{,}000001$ может оказаться "слишком большим". Является ли шаг "достаточно маленьким" или все еще "большим", зависит от решаемой задачи.

Во-вторых, в других задачах шаг $dt = 0{,}000001$ будет "чересчур маленьким". Если, например, требуемую точность решения можно достигнуть уже при $dt = 1$, то запуск алгоритма сразу с $dt = 0{,}000001$ приведет к тому, что алгоритм будет работать в 1 000 000 (миллион!) раз дольше, чем это необходимо. Ведь чем меньше шаг по времени, тем больше состояний придется вычислить в алгоритме А120 и тем дольше будет работать алгоритм.

Поэтому стандартный практический прием состоит в том, что сначала алгоритм выполняют с каким-то кажущимся соответствующим задаче шагом. (В задаче падения парашютиста я лично начал бы с шага $dt = 0{,}1$.)

Если не говорить о § 16 (который слишком многоплановый, чтобы его с чем-то сравнивать), то § 25 является самым математизированным параграфом учебника. Его можно пропустить, если уровень математической подготовки учеников недостаточен, например, если речь идет о 7-м классе. Хотя учебник предназначен для 10—11-х классов, по нему часто преподают в 7-, 8-м или 9-м классах. Это сейчас уже достаточно распространенная практика. Мы в учебнике постарались весь математизированный материал локализовать (это § 16, § 25 и в меньшей степени § 26), и эти параграфы можно пропустить при преподавании в младших или ослабленных классах.

§ 25, как и любой другой параграф "Применений", начинается с беллетристики, описывающей специфику этой области применений ЭВМ. В данном случае рассказывается про томографию, т.е. получение "срезов" человеческого тела. Этот пример вы найдете и в книге [3К]. Впрочем, есть и другие примеры, требующие огромных вычислений. Например, вы можете рассказать (в учебнике этого нет), что для запуска ракеты на нужную орбиту необходимо учитывать не только сопротивление воздуха, но и то, что это сопротивление падает с набором высоты. Надо учитывать изменение веса ракеты по мере расхода топлива. Надо учитывать не только притяжение Земли и его изменение, но и вращение Земли, силу Кориолиса. И еще целый ряд факторов.

В отличие от предыдущих параграфов раздел "как" этого параграфа практически не имеет никакого отношения к первой части (к беллетристике). Математические основы, используемые для построения картинки плотности тела в томографе или для точного расчета запуска ракеты на нужную орбиту, слишком сложны и громоздки для школьного учебника. Поэтому здесь разбираются традиционные методы приближенных вычислений: вычисление корня функции методом деления отрезка пополам (п. 25.4), вычисление интеграла методом трапеций (п. 25.5) и метод Монте-Карло (п. 25.6 и 25.7). Изложение первых двух методов практически дословно повторяет соответствующий материал из учебников [Ершов]. (Тогда этот материал рассматривался как основной, поскольку все изложение строилось на математическом материале.)

Я думаю, что все это вам хорошо знакомо и не нуждается в повторениях.

Метод Монте-Карло.

Третий метод — метод Монте-Карло. Хорошее изложение этого метода вы можете найти в учебнике [Гейн]. Там же имеется замечательный образ, объясняющий суть метода: представьте себе, что на квадратной детской площадке расположена песочница неправильной формы. Представьте себе далее, что прошел снегопад и выпало много снега. Понятно, что количество снега (снежинок), попавшего в песочницу, будет относиться ко всему снегу, выпавшему на детскую площадку, так же, как площадь песочницы относится к площади всей площадки. Поэтому для подсчета площади песочницы мы можем подсчитать, какая часть снега попала в песочницу, и умножить эту величину на площадь всей площадки.

В нашем учебнике фигурка на плоскости (внутри квадрата) засыпается ровным слоем "песка" — и считается, сколько "песчинок" попало внутрь фигурки. Конечно, и "снег", и "песок" — не более чем образы. Важно, что "снежинок" ("песчинок") много и что они покрывают весь квадрат (детскую площадку) равномерно.

При вычислениях на ЭВМ "снег" ("песок") надо проимитировать, т.е. следует случайным образом "бросать" точки ("снежинки", "песчинки") в квадрат так, чтобы в целом они оказались равномерно распределенными по квадрату. Для этого у нас используется специальная функция алгоритмического языка **rnd** (от англ. *random* — случайный).

Тогда если мы умеем для каждой точки быстро вычислить, внутри она или снаружи нашей фигурки (т.е. площадь фигурки считать не умеем, а определить, находится точка внутри или снаружи фигурки, можем), то можно поступить следующим образом. "Накидаем" достаточно много точек и подсчитаем, какая часть (k) из них попала внутрь фигурки. Если точек достаточно много и они распределены равномерно, то площадь фигурки можно приближенно вычислить, умножив k на площадь квадрата. Соответствующий алгоритм — алгоритм А117 — приведен в учебнике на с. 204.

В пункте 25.7 этот метод применяется для приближенного вычисления числа π. Предположим, что мы не знаем, чему равно π, но знаем, что площадь круга равна πR^2. Мы можем взять круг единичного радиуса (его площадь равна π) и воспользоваться методом Монте-Карло, т.е. накидать случайных точек ("снежинок", "песчинок") и подсчитать приблизительно площадь круга (т.е. число π). Соответствующий алгоритм А118 приведен в учебнике на с. 204. На с. 205 даны результаты выполнения этого алгоритма, где при 5000 точек π получается равным уже 3,1424. Естественно, если мы выполним алгоритм еще раз, то при том же количестве точек (5000) число π может получиться другим, ведь точки "бросаются" случайно и результаты каждый раз могут меняться.

Специфика вычислений на ЭВМ.

Самое важное в этом параграфе — п. 25.3 на с. 200 — специфика вычислений на ЭВМ и их отличие от того, что принято в математике.

Главное, что надо понять, — это:

а) то, что машина считает приближенно, — мы получаем не точный ответ, а приблизительный;

б) за счет того, что мы считаем приближенно, мы можем подсчитать что-то, не имея точных формул и не выводя их; это — совершенно другой, отличный от школьной математики, подход к решению задач.

Ни для томографа, ни для запуска ракеты невозможно написать ***математическую формулу*** получения результата. Например, при расчете запуска ракеты невозможно написать формулы, выражающие местонахождение и скорость ракеты через начальные величины (вес топлива, направление старта и т.д.) с учетом вращения Земли, изменения сопротивления воздуха, изменения массы топлива и пр. Для получения ответов в обоих случаях требуются вычисления — миллионы и миллиарды операций.

Имеется одна очень распространенная ошибка, часто встречающаяся при преподавании информатики, когда учитель приводит, например, формулу корней квадратного уравнения, а потом просит составить по этой формуле алгоритм вычисления корней. Естественно, возникает вопрос, зачем это надо. Ведь по формуле можно и так подсчитать. Информатика при этом как бы лишается своего предмета, спускается до уровня умения пользования калькулятором. Конечно, и калькулятором надо уметь пользоваться. Но не может же это быть предметом школьного курса.

Здесь очень важно подчеркнуть, что мы можем использовать компьютер для нахождения величин, для которых мы никаких формул не знаем, и что алгоритмы при этом могут получаться достаточно простые.

Возьмем хотя бы нахождение корня уравнения $\cos x = x$ методом деления отрезка пополам (п. 25.4 учебника). Мы не знаем и не можем вывести ***никакой*** формулы для корня этого уравнения. В курсе школьной математики такую задачу вообще решить нельзя. А приближенно можем подсчитать корень с любой заданной точностью. Именно на этот нюанс надо обратить внимание: вычисления на ЭВМ не точные, а приближенные, но зато мы их проводим, не имея точных формул, не имея математического аппарата для нахождения корней, интегралов или площадей и т.д. Важно, что это — другая область, не охватываемая обычной математикой, обычными формулами.

§ 26. Моделирование и вычислительный эксперимент на ЭВМ

Это третий (после § 23 и § 24) содержательный параграф про применения ЭВМ. Начинается он, как всегда, с беллетристической части, на этот раз совсем небольшой, которая фактически служит просто введением. Единственная мысль этого введения состоит в том, что всегда, когда нам нужна информация о некотором процессе или явлении, мы можем попытаться составить информационную модель этого явления, проанализировать ее на ЭВМ и получить желаемую информацию. Особенно это применимо к таким явлениям, как исследование последствий взрывов, катастроф, влияния жестких излучений и пр., которые в реальности лучше не проводить. Что будет с Землей, если озоновый слой исчезнет? Мы не можем для ответа на этот вопрос провести полномасштабный эксперимент, но можем попробовать смоделировать интересующие нас явления на ЭВМ и просчитать возможные последствия.

Вообще говоря, все, что мы делаем на ЭВМ, — это всегда моделирование. Но в данном случае речь идет о более конкретных, физических задачах. Моделирование физических явлений (вычислительный эксперимент на ЭВМ) рассматривается в учебнике на примере падения парашютиста с учетом сопротивления воздуха. Вы можете в качестве преамбулы к параграфу повторить также эссе про расчет запуска ракеты на нужную орбиту, где надо учитывать сопротивление воздуха, изменение этого сопротивления и уменьшение массы ракеты с ростом высоты и т.п.

Расчет запуска ракеты — сложная задача. А вот падение парашютиста с нераскрытым парашютом с учетом сопротивления воздуха — задача, с одной стороны, достаточно простая, а с другой — привычными школьными физическими формулами не описываемая.

Формулировка этой задачи обычно вызывает оживление, поскольку фраза “падение парашютиста с нераскрытым парашютом” естественно предполагает печальный конец. Потом в упражнениях (упр. 1 и 2 на с. 211) появятся более “щадящие” парашютиста формулировки, когда парашютист падает с нераскрытым парашютом первые 700 м, а потом парашют все-таки раскрывает. Но это в упражнениях. А в тексте параграфа мы исследуем падение с нераскрытым парашютом до самой земли.

Итак, парашютист падает с нераскрывшимся парашютом. Наша задача — промоделировать этот процесс на ЭВМ. Самое главное, существо всего параграфа здесь — это метод дискретизации непрерывных процессов (п. 26.1). Все содержание параграфа — ровно один метод, метод дискретизации непрерывных процессов.

Что значит “промоделировать” на ЭВМ процесс падения? Такого у нас еще не было. Мы моделировали кинозал, транспортную сеть и прочие ***объекты*** — это понятно. А что значит ***модель процесса*** падения? Как промоделировать на ЭВМ такой непрерывный (плавный) процесс? Ответу на этот вопрос и посвящен § 26.

Как всегда, когда мы хотим пояснить что-то новое, лучше всего использовать аналогию с чем-нибудь хорошо знакомым. Тогда даже если полного понимания и не появляется, то во всяком случае появляются иллюзия понимания, успокоенность и возможность решать задачи. В данном случае в качестве такой аналогии для метода дискретизации непрерывных процессов используется киносъемка.

Дискретизация — это замена непрерывного, плавного процесса последовательностью его состояний. Представьте себе, что мы засняли падение бедного парашютиста на кинопленку (обычную старинную кинопленку, которую можно рассматривать на свет, в которой один кадр сменяет другой). Такая кинопленка — это некоторая информация о процессе падения. Если ее прокрутить на киноаппарате, то процесс падения можно воссоздать на экране, причем с вполне приличной точностью. Но если рассмотреть саму кинопленку как объект, то она никакого плавного процесса не содержит, а состоит из последовательности кадров. На одном кадре — парашютист в одной точке, на другом — в другой, а между ними ничего нет. Плавный процесс падения парашютиста оказался заменен на последовательность его состояний. Этот переход и называется дискретизацией непрерывных процессов.

Как и в примере с кинопленкой, при дискретизации непрерывных процессов, как правило, состояния процесса фиксируются с некоторым шагом по времени, например, с шагом 0,01 с. Мы можем считать, что снимали кино с частотой 100 кадров в секунду. Тогда два соседних кадра на кинопленке — это два состояния процесса, отличающиеся друг от друга на 0,01 с по времени. А непрерывный процесс падения в итоге заменен на последовательность его состояний с шагом 0,01 с по времени.

Переходя теперь от аналогии с кинопленкой к собственно моделированию на ЭВМ, мы должны отметить две вещи:

1) Мы не можем “заснять” процесс падения — ведь никакого реального процесса у нас нет. Мы, наоборот, хотим промоделировать этот процесс на ЭВМ.
2) Мы не умеем моделировать на ЭВМ непрерывные процессы, но можем попытаться рассчитать последовательность состояний парашютиста с некоторым шагом по времени. При этом в “состояние” парашютиста мы можем включить любую интересующую нас информацию: высоту, скорость, ускорение и пр. Каждое состояние парашютиста в алгоритмическом языке можно задать набором значений соответствующих величин алгоритмического языка. Последовательные состояния парашютиста могут получаться как последовательные состояния (значения) этого набора величин в ходе выполнения некоторого алгоритма.

Таким образом, метод дискретизации непрерывных процессов как метод алгоритмизации состоит в том, что:

1) мы представляем (воображаем) себе непрерывный процесс как некоторую дискретную последовательность его состояний, обычно с каким-то шагом по времени;
2) далее мы задаем информационную модель (набор величин) для описания одного, отдельно взятого ***состояния*** процесса;
3) наконец, мы пишем алгоритм, в ходе выполнения которого происходит переход из одного состояния в другое (т.е. меняются значения величин информационной модели) и вычисляются интересующие нас характеристики всего процесса в целом. Обратите внимание, что информационная модель (как набор величин алгоритмического языка) при этом строится для представления ***одного состояния,*** а не всего моделируемого процесса.

Моделью самого процесса можно считать ***ход выполнения алгоритма,*** который переводит значения величин от одного состояния к другому, как бы воспроизводя (моделируя) процесс.

Самым главным в методе дискретизации непрерывных процессов является то, что переход от одного состояния к другому в этом методе производится дискретно — “скачком”. Другими словами, по набору значений для одного состояния вычисляется набор значений в следующем состоянии. А это уже хорошо нам знакомое, обычное явление в информатике.

Возвращаясь к модели процесса падения, состояние парашютиста можно задать моделью М19 (с. 208 учебника), т.е. тройкой величин (t, h, v), задающих время, расстояние до земли и скорость парашютиста. Для перехода к следующему состоянию нужно выразить новые значения этих величин через предыдущие:

$$t_{\text{новое}} = t_{\text{старое}} + dt$$
$$h_{\text{новое}} = h_{\text{старое}} - v_{\text{старое}} \cdot dt$$
$$v_{\text{новое}} = v_{\text{старое}} + a_{\text{старое}} \cdot dt$$

— где dt — шаг по времени. Надо понимать, что при написании этих формул мы считаем, что между последовательными состояниями нашей модели (между соседними кадрами кино) ничего не происходит, а значит, ничего и не меняется. На протяжении всего шага по времени (например, 0,01 с) ни время, ни высота, ни скорость не меняются. Потом, по истечении этого шага по времени, время, высота и скорость меняются скачком: время сразу увеличивается на шаг по времени, высота и скорость также скачком устанавливаются в новые значения. Затем на протяжении следующего интервала времени опять ничего не меняется. И т.д.

Вопрос: А где формула для $a_{\text{новое}}$?
Ответ: По условиям нашей задачи $a = g - kv^2$, т.е. ускорение однозначно определяется по значению скорости. Эта формула носит “абсолютный” характер, верна для любого момента времени и не зависит от “старых” и “новых”.

Поэтому $a_{\text{старое}} = g - kv^2_{\text{старое}}$, $a_{\text{новое}} = g - kv^2_{\text{новое}}$.

А поскольку ускорение вычисляется через значение скорости, величина для ускорения в информационную модель М19 не входит.

Написанные выше соотношения, выражающие новые значения (t, h, v) через их старые значения, — это и есть (в терминологии § 16) рекуррентные соотношения, задающие последовательность состояний в моделируемом процессе падения парашютиста. Соответствующий алгоритм далее записывается очевидным образом (алгоритм А120 на с. 209 учебника).

Интернет для начинающих

Продолжение. Начало на с. 1

Вася. Конференции в Интернете? Когда они проходят?

Папа. У этих конференций нет сроков проведения. Они работают постоянно.

Вася. И как же можно принять в них участие?

Папа. Прежде всего нужно найти интересную сетевую конференцию. Потом можно читать сообщения, которые там публикуются, и писать свои статьи.

Вася. Давай начнем по порядку. Как найти в сети конференцию?

Папа. Конференции в сети поддерживают специальные серверы, их называют серверами новостей. В сети нет единого "склада", куда помещаются "труды" участников. Серверы новостей постоянно обмениваются друг с другом по сети списками сообщений. Если какая-то статья отсутствует у одного их них, он переписывает ее к себе от соседа. Копии статей конференций хранятся и на сервере новостей у нашего провайдера. Если ты пошлешь в конференцию статью, она прежде всего появится на нашем сервере, а уж потом он "поделится" новой информацией со своими "товарищами".

Вася. Как часто серверы новостей сверяют свои списки?

Папа. Серверы новостей проводят тысячи и даже миллионы сравнений ежедневно. Поскольку они — компьютеры, делать им это нетрудно.

Вася. В сети должно существовать много конференций. Ведь интересы у людей такие разные!

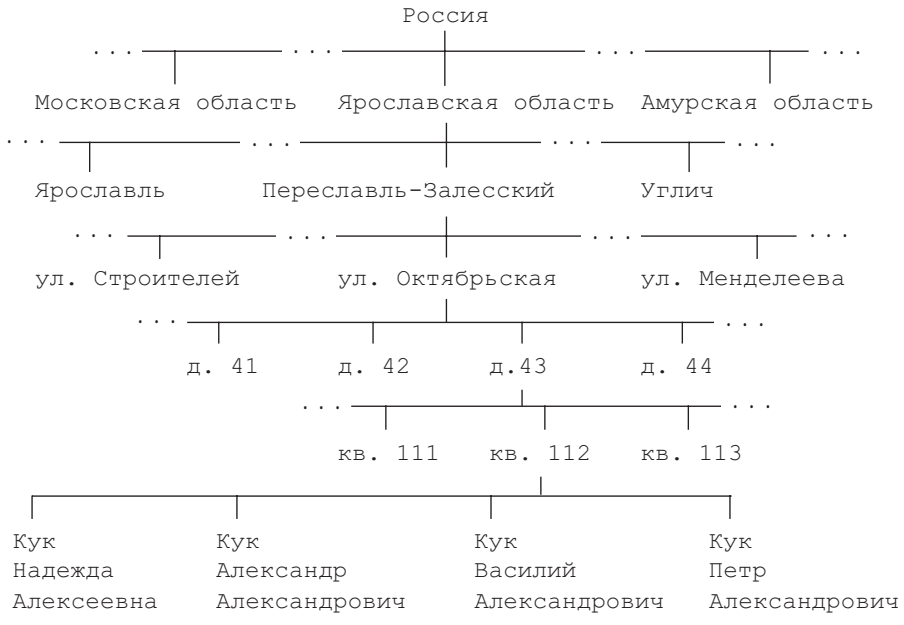
Папа. Конференции "растут" в сети деревьями.

Вася. Красиво сказано! А что это за деревья — дубы или березы?

Папа. Я имею в виду иерархические схемы построения конференций. Как ты должен помнить, иерархия — это зависимость по подчинению или включению одних объектов в другие.

Вася. Мы говорили об иерархии, когда обсуждали принципы построения обычных почтовых адресов и адресов в сети, способы упорядочивания файлов на диске компьютера и в ftp-архивах.

Папа. Каждая иерархия напоминает дерево, у нее есть корень, ветви и листья. Например, иерархия почтовых адресов в каждой стране имеет свой корень — название страны. От этого корня отходят ветви первого уровня — области. Каждая ветвь-область имеет свои веточки — города, города — улицы, улицы — дома, дома — квартиры, квартиры имеют своих жильцов — листья. Листом в иерархическом дереве называется ветвь, от которой нет отростков:



Вася. Схема похожа на дерево, только это дерево растет не вверх, а вниз. А мы с тобой на нем — листочки, правда?

Папа. Получается, что так. Деревья можно рисовать и по-другому:

```
Россия
├─ Московская область
├─ Ярославская область
│     ⋮
│     ├─ Ярославль
│     ├─ Переславль-Залесский
│     │     ⋮
│     │     ├─ ул. Строителей
│     │     ├─ ул. Октябрьская
│     │     │     ⋮
│     │     │     ├─ д. 41
│     │     │     ├─ д. 42
│     │     │     ├─ д. 43
│     │     │     │     ⋮
│     │     │     │     ├─ кв. 111
│     │     │     │     ├─ кв. 112
│     │     │     │     │     ├─ Кук Надежда Алексеевна
│     │     │     │     │     ├─ Кук Александр Александрович
│     │     │     │     │     ├─ Кук Василий Александрович
│     │     │     │     │     └─ Кук Петр Александрович
│     │     │     │     ├─ кв. 113
│     │     │     │     ⋮
│     │     │     ├─ д. 44
│     │     │     ⋮
│     │     ├─ ул. Менделеева
│     │     ⋮
│     ├─ Углич
│     ⋮
├─ Амурская область
⋮
```

Вася. Дерево почтовых адресов мне понятно. А какие деревья растут в Интернете?

Папа. Конференции в Интернете организованы как тематические деревья. В сети работают международные конференции, объединенные в следующие группы:

comp — Компьютеры и относящиеся к ним темы;
news — Новости о самой системе конференций;
rec — Хобби, игры и отдых;
sci — Наука;
soc — Социальные группы;
misc — Дискуссии, которые больше некуда поместить.

Существует достаточно много национальных групп конференций. В России самой популярной является группа **relcom**. Я нарисую только очень небольшую часть релкомовского дерева:

```
relcom
├─ relcom.archives              Новые поступления на файловые серверы
├─ relcom.arts.magic            Обсуждение магии и сопутствующих тем
├─ relcom.arts.photo.img        Публикация авторских фотографических работ
├─ relcom.auto                  Все об автомобилях и управлении ими
├─ relcom.cinema                Общие разговоры про кино
│  ├─ relcom.cinema.soap        Для любителей "мыльных опер"
│  ├─ relcom.commerce.computers Компьютеры, модемы, др.
│  ├─ relcom.commerce.tour      Туризм, отдых, развлечения
│  ├─ relcom.comp.animation     Компьютерные анимационные программы
│  └─ relcom.comp.binaries      Свободно распространяемые программы
├─ relcom.education             Проблемы образования
├─ relcom.games                 Обсуждение игр
│  └─ relcom.games.pbem         Игры посредством электронной почты
├─ relcom.humor                 Юмор
│  └─ relcom.humor.lus          Шутки, юмористические произведения
├─ relcom.kids                  О детях, детям
├─ relcom.music                 Для любителей/ценителей музыки
├─ relcom.rec.puzzles           Головоломки, задачи, логические игры
├─ relcom.talk                  Беседы на произвольные темы
│  └─ relcom.talk.sport         Обсуждение вопросов, связанных со спортом
├─ relcom.test                  Конференция для тестовых сообщений
⋮
```

Вася. Я заметил, что название конференции состоит из нескольких английских слов, разделенных точками.

Папа. Каждая точка в названии конференции указывает на отросток в дереве. Например, конференция **relcom.cinema.soap** находится в подгруппе **cinema** группы **relcom**.

Вася. Расположение конференций в сети напоминает мне расположение файлов в каталогах на нашем компьютере.

Папа. Это неудивительно, ведь и для файлов, и для конференций используется один и тот же способ упорядочивания — иерархия. Давай подведем предварительные итоги нашей беседы:

— конференция — это набор сообщений (статей);
— конференции организованы в группы, и каждая группа имеет иерархическую структуру;
— материалы конференций хранят специальные серверы новостей, которые обмениваются новинками друг с другом;
— статьи конференций можно читать;
— в конференциях можно публиковать свои собственные заметки.

Вася. Мне хочется принять участие в конференции **relcom.games**. Я почитал бы ее статьи и, может быть, даже написал свою заметку. Как это сделать?

Папа. Ты уже не новичок в Интернете. Попробуй догадаться сам, как можно получать материалы конференций и посылать свои.

Вася. Мой скромный опыт позволяет сделать предположение о существовании специальных роботов-исполнителей, которым я могу посылать запросы на чтение статей и свои собственные материалы для публикаций.

Папа. Ты прав! Такие роботы существуют, и с ними можно работать по электронной почте.

Вася. С каким роботом будем переписываться мы?

Папа. Как правило, на каждом сервере новостей есть почтовый робот, обслуживающий клиентов своего сервера, но можно пользоваться услугами роботов "чужих" серверов. Именно так мы и поступим, потому что наш провайдер не имеет собственного почтового робота на своем сервере новостей. Мы будем работать с исполнителем **news@demos.su**.

Вася. Какая СКИ у этого робота?

Папа. Я приготовил список самых необходимых команд этого исполнителя:

Команда	Комментарий
HELP	Выдать краткую инструкцию.
LIST	Выдать список конференций.
GROUP конференция	Перейти в указанную конференцию. Эта конференция становится "текущей" (для команд **ARTICLE** и **INDEX**).
INDEX -n	Выдать список **n** последних статей текущей конференции. Например, команда **INDEX-20** — заказ на получение заголовков последних 20 публикаций.
ART номер	Выслать статью с указанным номером.
QUIT	Конец задания для робота.

Продолжение на с. 14

Интернет для начинающих

Продолжение. См. с. 1, 13

Вася. Хотелось бы посмотреть на "живой" пример письма роботу.

Папа. Первое письмо нужно послать совсем короткое...

Вася. Знаю, знаю! Это письмо должно содержать всего одну команду — **HELP**. Так надо начинать работу с любым сетевым исполнителем.

Папа. Верно! Если нужно получить свежий список всех конференций в сети, то нужно направить по адресу **news@demos.su** простое письмо из двух команд:

```
LIST
QUIT
```

Затем можно написать, например, такое письмо:

```
GROOP relcom.kids
INDEX —50
QUIT
```

Вася. Думаю, мы получим в ответ на эту программу список последних пятидесяти статей конференции **relcom.kids**.

Папа. Конечно! Этот список будет иметь примерно такой вид:

```
ART 5403 3.9K kurs@robotland.users.botik.ru Университет: новости.
                          ..................
ART 5412 1.4K kurs@robotland.users.botik.ru Университет: список участников.
```

Кто поместил статью — Название статьи

Объем материала

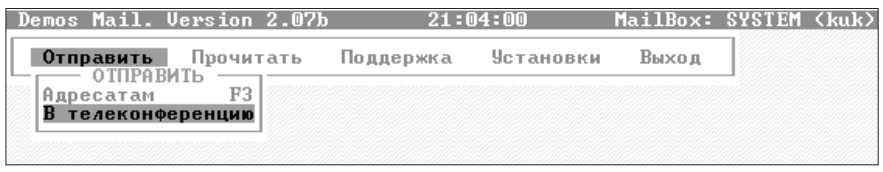
Номер статьи в конференции

Теперь можно направить письмо-заказ роботу на получение конкретных публикаций:

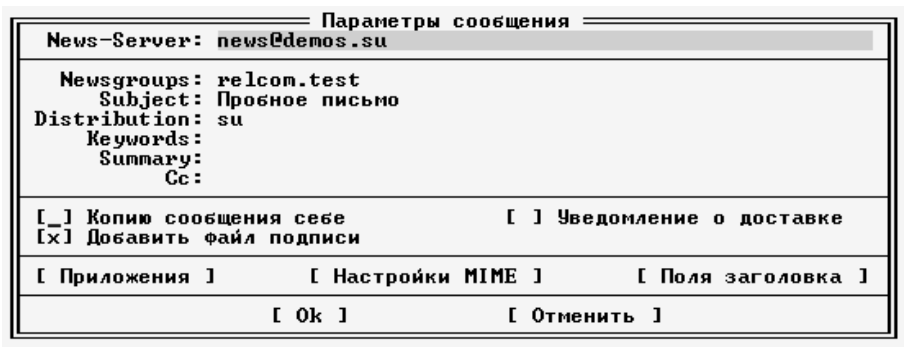
```
GROOP relcom.kids
ART 5403
ART 5412
QUIT
```

Вася. Замечательно! Следить за конференцией и получать нужные статьи, оказывается, очень просто. А как мне послать свою собственную заметку?

Папа. Это тоже просто. В программе DMAIL в главном меню нужно выбрать позицию **Отправить**, а в открывшемся меню — пункт **В телеконференцию**.



Вася. На экране появляется окно **Параметры сообщения**.



Папа. Рассмотрим поля этого окна.

News-Server — адрес робота новостей, это поле — не что иное, как адресное поле заголовка: **To:** обычного письма.

Newsgroups — название конференции, в которую посылается сообщение.

Subject — это поле служит для указания темы сообщения. Содержание этого поля показывается в списке статей конференции.

Distribution — в этом поле нужно указать область распространения сообщения. Можно использовать следующие сокращения:

- **world** — международные конференции;
- **eunet** — европейские конференции;
- **su** — конференции на территории СНГ.

Keywords — это поле является необязательным и служит для указания ключевых слов сообщения. Эта информация нужна для автоматического поиска в конференциях нужных статей по ключевому слову.

Summary — это необязательное поле служит для указания краткого содержания посылаемого сообщения.

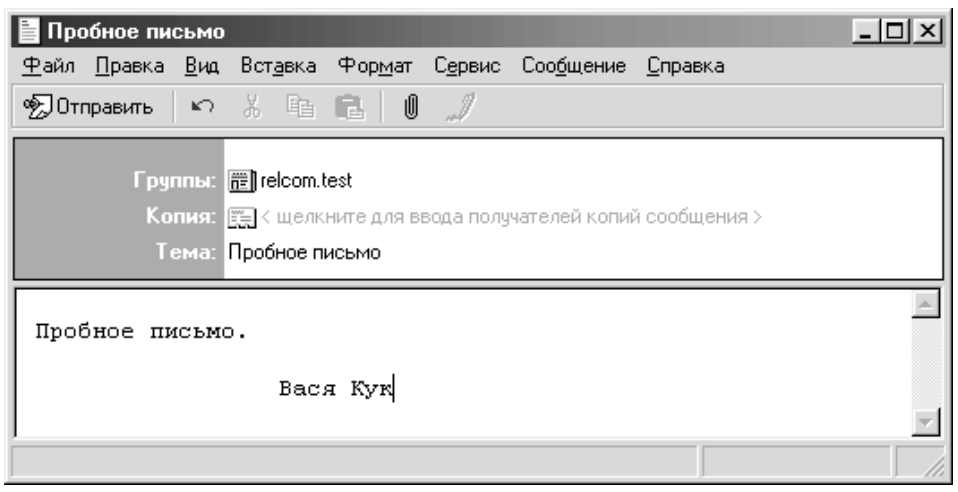
Cc — необязательное поле для адресов получателей, которым будут отправлены копии вашего сообщения. Заполняется аналогично такому же полю при отправке почты конкретному адресату.

Вася. Меня немного смутило поле **Distribution**. Разве можно контролировать распространение информации в сети?

Папа. Ты забыл, что письмо, отправленное в конференцию, попадает сначала на тот сервер новостей, с которым мы работаем, а дальше по сети статья распространяется только при сеансах связи нашего сервера с другими. Вот здесь-то наш сервер и будет учитывать значение поля **Distribution**. Если с ним свяжется сервер, не входящий в регион, указанный в этом поле, наша статья передаваться не будет. Такой способ распространения публикаций существенно уменьшает бесполезный трафик в сети.

Вася. А как отправить письмо в конференцию с помощью почтовой программы Windows?

Папа. В Windows существуют специальные программы для работы с серверами новостей. Как правило, эти программы встроены в единый комплект программ для работы с Интернетом наряду с программой обычной электронной переписки, с которой мы работали раньше. Просмотр конференций в Windows происходит в интерактивном режиме, что очень удобно. Интерактивный режим означает, что заголовки статей и сами статьи можно просматривать непосредственно после установления связи с сервером новостей. Отправить письмо в конференцию также очень просто. Для этого в программе работы с конференциями существует специальная кнопка **Создать сообщение**, аналогичная такой же кнопке в обычной почтовой программе. Если щелкнуть по этой кнопке мышкой, на экране появится бланк письма, подобный тому, с которым мы работали раньше, отправляя письма роботландскому Паучку:



Для учителя
(Комментарии для серьезных читателей)

Дополнения к диалогам Куков
Протоколы работы с серверами новостей

Начало развитию системы конференций в Интернете было положено в 1979 году двумя аспирантами университета штата Каролина Томом Траскоттом и Джимом Эллисом. За основу системы был взят протокол UUCP.

В 1986 году работа с конференциями была впервые реализована на основе нового специального протокола NNTP (*Network News Transfer Protocol* — сетевой протокол передачи новостей). Этот протокол основывается на стандартном для Интернета протоколе TCP/IP. Именно введение протокола NNTP предоставило возможность пользователям просматривать конференции в интерактивном режиме без использования электронной почты.

При передаче новостей по протоколу UUCP все новые сообщения передаются от сервера к серверу, и таким образом в сети возникает столько баз данных, сколько в сети серверов новостей. Именно такой механизм передачи информации в сети конференций обсуждался в диалогах папы и Васи. Он не очень эффективен, потому что информация в сети неоправданно дублируется: на сервер попадают даже те статьи, которые клиентам этого сервера совсем не нужны.

По протоколу NNTP серверы обмениваются информацией только о наличии сообщений, а переписывают к себе только те статьи, на которые есть заказы у клиентов.

Срок хранения сообщений в конференциях

Может случиться так, что вы заказали статью из конференции, а она не пришла. Причин может быть несколько:

- на сервере новостей произошел сбой;
- автор снял свою статью из конференции;
- истек срок хранения данной статьи на сервере.

Последняя причина — наиболее вероятная. Обычно статьи хранятся одну-две недели, а потом помещаются в архив.

Модерация конференций

Сетевые конференции бывают двух типов: свободные и модерируемые. В конференциях первого типа сообщения никем не контролируются. Вы посылаете произвольный текст, и он сразу появляется в Интернете. Во главе конференций второго типа стоит редактор — модератор. Ваше сообщение сначала попадает к нему, и он принимает решение о публикации. Такие конференции называются модерируемыми. Английское слово *moderator* можно перевести как арбитр, посредник, ведущий дискуссию. Модератор принимает решение о публикации материала исходя из устава конференции, который принимается открытым сетевым голосованием. Уставы существуют не только в модерируемых конференциях. А кроме того, конференции могут из открытых становиться модерируемыми и наоборот. Например, конференция **relcom.education** была открытой конференцией, а после того как качество и тематика ее сообщений перестали устраивать большинство ее членов, она стала модерируемой. Это произошло осенью 1998 года, после открытого сетевого голосования участников.

Нити обсуждений

Материалы, публикуемые в сетевых конференциях, можно условно разделить на два типа: одиночные и нитевидные.

Одиночные материалы — это, как правило, реклама, информационные сообщения, объявления. На такие материалы откликов в конференции обычно не бывает. Если материал заинтересовал кого-нибудь, посылают письмо автору сообщения.

Другое дело — проблемные публикации. Такие сообщения вызывают дискуссионную цепочку сообщений — в конференции образуется нить. Определить нить в конференции очень просто: обычно это цепочка сообщений с одинаковой темой, перед которой проставлен префикс **Re:**. Двигаясь "вниз" по датам таких публикаций, можно в конце концов найти исходное сообщение, породившее всю цепочку, — префикс **Re:** у этого сообщения отсутствует.

На первых порах, вероятно, вы будете просто читать чужие сообщения. Но рано или поздно вам захочется написать свое мнение в нитевидное обсуждение или образовать свою собственную нить.

Кросс-постинг

Может случиться, что вы решите послать сообщение не в одну, а в несколько конференций. Вместо того чтобы посылать несколько писем, можно послать одно, указав в поле названия несколько имен конференций. Такой способ отправки статьи называется кросс-постингом.

Помещать сообщение в большое число конференций или неподходящие конференции считается дурным тоном. Кросс-постинг — любимое занятие спамеров и людей, надоедливо рекламирующих свои услуги или продукты.

"Наезды"

Интернет — прекрасный способ общения людей. Он разрушает политические границы и делает людей ближе друг другу.

Однако по непонятным причинам сетевое общение превращает некоторых людей в злобных фурий, полных сарказма, цинизма и необъяснимого раздражения по любому поводу.

Может быть, такое поведение людей в сети объясняется тем, что, записывая текст на компьютере и отсылая его в конференцию, автору не приходится смотреть в глаза своим читателям.

Если вы новичок в сети и посылаете в конференцию первое радостное сообщение, приготовьтесь заранее: не исключено, что на вас "наедет" какой-нибудь псих. Не надо расстраиваться и не надо поддаваться на провокацию, иначе ваша перебранка будет идти в конференции неделями, образуя целую нить бесполезных ругательств. Ваше первое пробное сообщение лучше всего послать в конференцию **relcom.test** — она специально предназначена для сообщений такого рода.

Общепринятые последовательности символов и сокращения

Символы	Толкование
***	Три звездочки в начале поля **Тема** означают официальное сообщение (как правило, это сообщение от модератора).
AFAIK	Насколько я знаю *(As Far As I Know)*.
BTW	Кстати, между прочим *(By The Way)*.
SY	Увидимся *(See You)*.
SYL	Увидимся позже *(See You Later)*.
FAQ	Часто задаваемый вопрос *(Frequently Asked Question)*.
FYI	Для вашей информации *(For Your Information)*.
etc.	И так далее.
IS	Я понимаю *(I See)*.
IMHO	По моему скромному мнению *(In My Humble Opinion* или *In My Honest Opinion)*.
IOW	Другими словами *(In Other Words)*.
OTOH	С другой стороны *(On The Other Hand)*.
POV	Точка зрения *(Point Of View)*.
NB!	Обратите внимание *(Nota Bene)*.
SY	Искренне ваш *(Sinserely Yours)*.
THX	Спасибо *(Thanks)*.
текст или _текст_	Выделение текста жирным шрифтом.
текст или ^^^^^ текст ~~~~~	Подчеркивание текста.

Этические проблемы

Участвуя в конференциях, следуйте простым и понятным правилам: не надо использовать нецензурные выражения в сообщениях, оскорблять людей, разжигать национальную рознь, призывать к свержению существующего строя...

Коммерческую рекламу следует посылать только в специально отведенные для этого конференции. Пробные и тестовые сообщения — только в тестовые конференции. Личные письма лучше посылать напрямую корреспонденту, а не публиковать для всеобщего обозрения.

Старайтесь не посылать в конференцию сообщения с вложениями и никогда не посылайте сообщения более 20 Кб.

Следите за тем, чтобы запись в теме сообщения точно отражала содержание публикации.

Не пишите сообщений, в которых есть строки длиннее 80 символов (лучше ограничить длину строки числом 64).

Следите за грамотностью письма и пользуйтесь программами-корректорами перед посылкой своих статей.

Подписка на конференции

Сетевую конференцию можно не просто просматривать время от времени, на нее можно подписаться. Это очень удобно: уже не нужно будет следить за новинками — почтовый робот будет сам посылать короткие сообщения о новых публикациях (примерно 5—10 наименований в день). Интересные публикации можно заказать, отредактировав присланный список и послав его обратно на сервер.

Для того чтобы подписаться на конференцию, нужно отправить в адрес почтового робота новостей письмо с пустой темой сообщения и с одной строкой-командой:

SUBSCRIBE <название конференции>

Подписка действует некоторое время, потом автоматически прекращается. Продолжить подписку можно, послав повторно команду **SUBSCRIBE**.

Отменить подписку можно в любой момент, если послать в адрес робота письмо:

UNSUBSCRIBE <название конференции>

Вопросы и упражнения

1. Что такое сетевая конференция?

Ответ. Сетевые конференции — это специальный сервис Интернета, который позволяет публиковать информацию в сети для всеобщего обозрения.

2. Где в сети хранятся материалы конференций?

Ответ. Сообщения конференций хранятся в сети на специальных серверах, которые называют серверами новостей. Эти серверы постоянно обмениваются новинками, предоставляя таким образом для своих клиентов доступ к сообщениям со всего мира.

3. В чем различие работы конференций по протоколам UUCP и NNTP?

Ответ. По протоколу UUCP на каждом сервере новостей создаются копии статей со всего мира, по протоколу NNTP — только копии названий статей, а сами статьи попадают на сервер при заказе материала клиентом. Кроме того, протокол NNTP позволяет просматривать конференции в интерактивном режиме без использования электронной почты.

4. Как организована система конференций в Интернете?

Ответ. Отдельные конференции объединяются в иерархические тематические группы. Существуют немного международных групп и большое число региональных (национальных). Самой популярной русскоязычной группой является группа **relcom**.

5. Как обозначаются конференции в Интернете?

Ответ. Исходя из иерархического принципа упорядочивания тематических групп, имена конференций строятся из слов-названий, разделяемых точками. Каждое следующее слово уточняет тематику, задаваемую предыдущим.

6. Как принять участие в конференции по электронной почте?

Ответ. Нужно узнать адрес почтового робота новостей, изучить его систему команд и с его помощью следить за сетевыми публикациями. Послать свое сообщение в конференцию можно при помощи почтовой программы в специальном режиме посылки письма в конференцию. Специальный режим необходим потому, что протоколы обмена новостями требуют специального оформления заголовка письма: наличия в нем полей **Newsgroups** и **Distribution**.

7. Как задать область распространения вашего сообщения?

Ответ. Область распространения сообщения задается полем **Distribution** на конверте письма. Можно использовать следующие обозначения:

world — международные конференции;
eunet — европейские конференции;
su — конференции на территории СНГ.

8. Как получить свежий список всех групп сетевых конференций?

Ответ. Нужно послать почтовому роботу команду **LIST**.

9. Каким правилам надо следовать, составляя статью для сетевой конференции?

Ответ:

— подумайте, так ли важно ваше сообщение миру;
— правильно выберите тематическую группу для публикации;
— указывайте как можно меньшую область распространения вашего сообщения;
— не посылайте одно и то же сообщение в несколько групп;
— пишите коротко и ясно, следите за грамматикой;
— запись в теме сообщения статьи для конференции должна точно отражать ее содержание;
— не пишите сообщения со строчками длиннее 80 символов;
— без крайней необходимости не присоединяйте к сообщению файлы и никогда не превышайте объем в 20 Кб;
— не злоупотребляйте компьютерным сленгом;
— не совершайте сетевых "наездов" и пишите корректные, вежливые сообщения;
— посылайте рекламу только в специальные группы;
— пробные сообщения посылайте только в тестовые группы;
— не посылайте в сеть чужие сообщения без разрешения автора;
— не пользуйтесь конференцией для личной переписки.

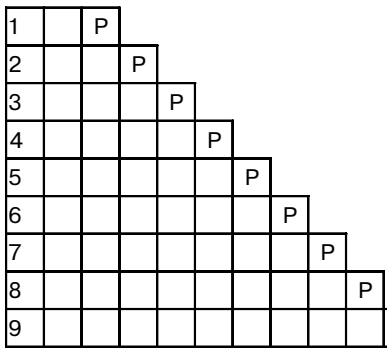
10. Закажите список статей из конференции.

11. Выберите интересные статьи из полученного списка и закажите их сетевому роботу.

12. Напишите пробное письмо в конференцию **relcom.test** и убедитесь, что она опубликована.

Кроссворд “тРапеция”

Каждое слово в этом кроссворде оканчивается на букву “Р”



1. Отечественная ЭВМ второго поколения. **2.** Изображение на экране, зафиксированное в некоторый момент времени. **3.** Область памяти для временного хранения информации. **4.** Указатель позиции на экране дисплея. **5.** Электронное устройство, имеющее два устойчивых состояния. **6.** Устройство ЭВМ для выполнения простейшей арифметической операции. **7.** Центральное устройство ЭВМ. **8.** Механический аппарат для выполнения арифметических действий, изобретенный петербургским инженером-изобретателем Однером. **9.** Карманное вычислительное устройство.

ОТВЕТЫ

1. Мир. 2. Кадр. 3. Буфер. 4. Курсор. 5. Триггер. 6. Сумматор. 7. Процессор. 8. Арифмометр. 9. Калькулятор.

Кто больше?

В № 10/99 мы опубликовали условие простой игры, которую можно использовать на викторинах, конкурсах, КВН и т.д. Суть ее в следующем: из букв данного (желательно достаточно “богатого” слова) требуется составить как можно больше других слов. Приведем несколько примеров.

1. Из слова КОДИРОВКА можно составить следующие слова:

Код, кар, корова, ковка, корка, кивок, кадр, кирка, кора, кок, кров, ода, око, окрик, орда, дока, док, дрова, диво, дар, двор, ива, икар, икра, ров, рок, род, рак, вор, вар, вода, вид, вика, ад, ар, аккорд.

2. ИНФОРМАЦИЯ:

Норма, нора, нация, нимфа, форма, фирма, фора, формация, ром, рация, роман, риф, рифма, мина, мор, мафия, мир, миф, омар, яр, яма, ярмо.

3. НАКОПИТЕЛЬ:

Нота, напиток, нить, нитка, оптика, отель, колье, кон, капель, кант, кот, канитель, китель, кит, кета, кол, клеть, клоп, клип, клан, конь, плот, пакт, пот, пика, пенка, пена, пила, пилка, пень, плен, плита, план, плоть, пони, плеть, питон, пол, полка, пинок, икона, икота, ель, лень, лань, липа, лак, линька, лето, линь.

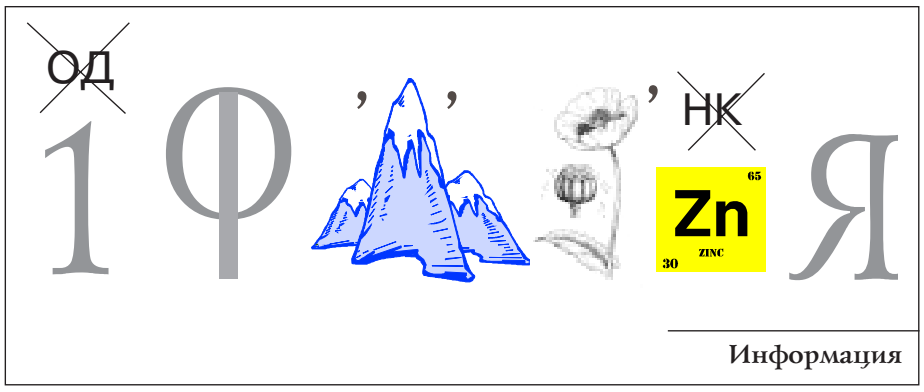
4. КОПИРОВАНИЕ:

Кров, корова, кар, карп, крен, кран, креп, кепи, кон, корона, кора, коран, опора, опока, опера, око, окно, овен, перо, пир, пар, пика, пена, право, порка, повар, пони, пинок, пан, пари, парик, пенка, пора, парк, покров, пиво, икона, инок, ива, икра, икар, ров, репа, река, рак, рок, ре, вор, век, вина, вино, вена, вера, вар, веко, вика, ворон, ворона, впрок, ар, нора, нива, норка, нерка, нерпа, норов, нрав, нерв.

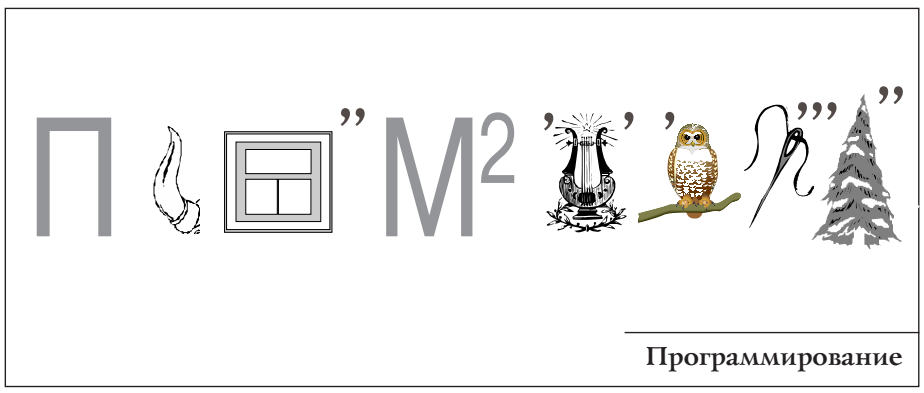
5. АССЕМБЛЕР:

Ар, месса, мера, марс, мел, бас, бес, барс, брасс, бал, бар, бра, лес, ре, раб, срам, сера.

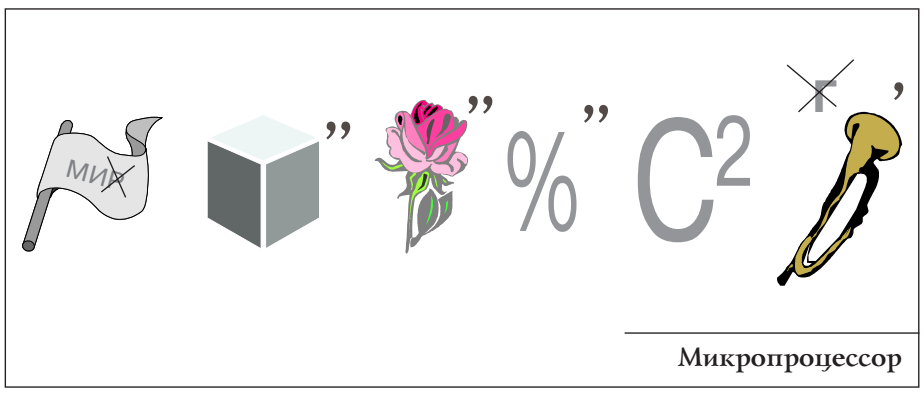
Ребусы по информатике



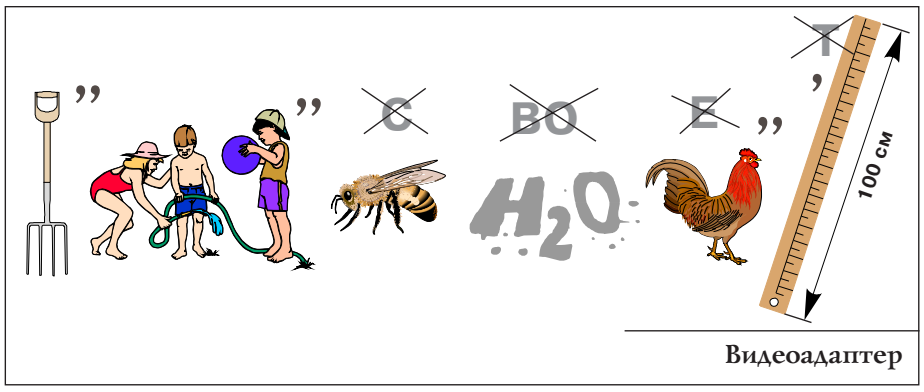
Информация



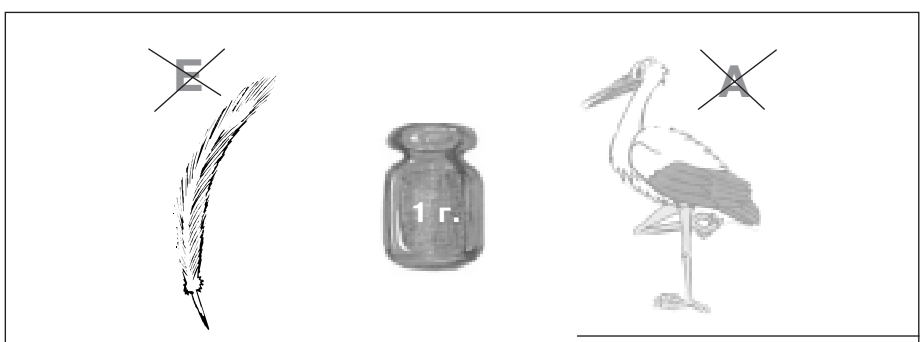
Программирование



Микропроцессор



Видеоадаптер



Программист

ВНЕКЛАССНАЯ РАБОТА ПО ИНФОРМАТИКЕ

Материалы подготовлены ***В.Г. Федориновым***

Отпечатано с готовых диапозитивов редакции в типографии “ПРЕССА”, 125865, Москва, ул. Правды, 24

Тираж 7000 экз.
Заказ №

121165, Москва, Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 249 9870

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
для предприятий и организаций 32591
комплекта приложений 32744

Internet: inf@1september.ru
Fidonet: 2:5020/69.32
WWW: http://www.1september.ru