№ 10 (347) 8–15 марта 2002

ИНФОРМАТИК

Еженедельная газета Издательского дома «ПЕРВОЕ СЕНТЯБРЯ»
ПОДПИСКА: (095) 249-47-58

LZ и LZW

25 лет назад, в 1977 году, был предложен новый метод сжатия информации без потерь, который стали называть “алгоритмом Лемпела — Зива” (LZ)

“...глядя на лес, мы не запоминаем очертания каждого дерева, однако впоследствии можем узнать конкретный лес или подробно его описать... Зная, как выглядит “среднестатистическое” дерево и “среднестатистический” лист, мы строим соответствующую модель “среднестатистического” леса. В качестве информации об определенном лесном массиве мы запоминаем различия между усредненной моделью в нашем представлении и реально наблюдаемым лесом: там листва рано пожелтела, там дерево согнулось, а там необычно высокое дерево. При необходимости по этим данным можно даже по памяти нарисовать картину. Вот это и есть один из видов сжатия графической информации” [1].

Окончание читайте на с. 32

Читайте в номере

Страницы повышения квалификации 4–12

Е.В. Андреева. Олимпиады по информатике. Пути к вершине

Ваши ученики не выступают на олимпиадах? Но ведь никогда не поздно начать. Однако для успешного выступления необходима серьезная подготовка. Знают ли они, например, какой алгоритм считается эффективным? Приглашаем на лекцию, тема которой — “Эффективные алгоритмы на графах”.

А.Г. Гейн. Введение в профессию “учитель информатики”

“Создание у учащихся мотивации... — краеугольный камень методики преподавания любого предмета, в частности, информатики”. При этом “важно, чтобы мотивированными были изучение каждой отдельно взятой темы, введение каждого понятия, овладение каждым умением, приобретение каждого навыка”.
Тема сегодняшней лекции — “Проблема мотивации: локальный аспект”.

Учебники..... 13–14

Н.Турлынович. Моделирование

Смоделируйте один шаг эволюции в игре “Жизнь”, а затем попробуйте создать модель калькулятора.
Предлагаем продолжить знакомство с главой из нового казахстанского учебника информатики, посвященной моделированию.

На стенд в кабинете информатики 16–17

“World Wide Web — Всемирная информационная паутина

История создания “Всемирной паутины” неразрывно связана с именами Тима Бернерса-Ли и Марка Андрессена.

Экзамены..... 18–24

Е.А. Еремин, А.П. Шестаков. Примерные ответы на примерные билеты

Имеют ли ваши ученики представление о характеристиках процессора, а также об основных характеристиках внутренней и внешней памяти компьютера?
Примерные билеты (и ответы на них) для проведения итоговой аттестации выпускников 9-х классов. Сегодня представлены третий и четвертый билеты.

Информатика всюду 25–29

Эрик Фрэнк Рассел. Дышите... не дышите

Новую рубрику открывает рассказ, подтверждающий, что информатика действительно всюду, причем не только на планете Земля...

ДАННЫЕ ДЛЯ ВЫПЛАТЫ АВТОРСКОГО ГОНОРАРА ЗА ПУБЛИКАЦИИ В “ИНФОРМАТИКЕ”

Дорогие авторы! Отправляя материалы для публикации в нашу газету, прикладывайте, пожалуйста, заполненный бланк с данными, необходимыми для выплаты гонорара.

Фамилия _____ **Имя** _____

Отчество _____

Приложение “ИНФОРМАТИКА”

Паспортные данные

серия _____

номер _____

когда выдан _____

кем выдан _____

Адрес прописки

индекс _____

город _____

улица _____

дом _____

корпус _____

квартира _____

Почтовый адрес для отправки гонорара
жители Москвы получают гонорар в редакции, все остальные должны обязательно указать данный адрес, даже если он совпадает с адресом прописки

индекс _____ город _____

_____

улица _____

дом _____ корпус _____ квартира _____

телефон _____

Дата рождения _____

Место рождения _____

Необходимость почтового перевода (да/нет)

_____

Номер страхового полиса Пенсионного фонда

_____

Номер свидетельства о постановке на учет в налоговом управлении (если есть) _____

"ЖАРКОЕ ЛЕТО-2002"

Уникальный комплект замечательных материалов к новому учебному году

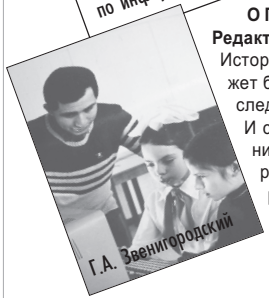
Как всегда, планируя очередное "Жаркое лето", мы учитывали пожелания наших читателей и их мнения о сериях тематических номеров прошлых лет. При этом мы стремились отдавать предпочтение материалам, которые наши подписчики смогут использовать на всем протяжении следующего учебного года. Исключение составляет, пожалуй, лишь номер, посвященный А.Г. Звенигородскому. Но мы считаем принципиально важным беречь память о тех, кто сделал для школьной информатики в России неизмеримо много. Обращаем ваше внимание на необычный номер — "Интеллектуальные игры". Не станем скрывать, что непосредственно к предмету "информатика" этот номер имеет мало отношения. Но все мы работаем в школе и знаем, что школьная жизнь не заканчивается звонком с последнего урока. Номер "Интеллектуальные игры" пригодится вам именно в этой "второй" жизни.

Д.М. Златопольский.
Задачник по электронным таблицам (Excel)
Вопросы, связанные с обработкой информации с помощью электронных таблиц, занимают важное место в школьном курсе информатики. Но специализированного школьного задачника по электронным таблицам нет. Вернее, не было, а теперь есть. Его первая часть будет опубликована в летних номерах нашей газеты.

А.И. Терентьев.
Организация школьного web-сайта
Как сделать школьный web-сайт "с нуля"? Как связать компьютеры в сеть, какое программное обеспечение выбрать и как его установить? Как, наконец, заставить все это "хозяйство" работать? В одном из летних номеров мы познакомим наших читателей с ответами на эти (и не только на эти!) вопросы.

А.И. Сенокосов.
Информатика и информатизация школы. Практические решения
Допустим, вы сделали школьный web-сайт. Протянули провода, настроили локальную сеть, установили программное обеспечение. Все работает. А что же дальше? Как организовать информационное наполнение сайта? Как "встроить" web-сайт в школьную жизнь? Все, кто решал эти вопросы, знают, что они-то и являются настоящими **вопросами**.

Д.М. Златопольский.
Внеклассная работа по информатике. Избранные задания
Летом мы предложим подписчикам целый номер с заданиями, которые можно использовать на викторинах, конкурсах и других внеклассных мероприятиях по информатике. В него войдут множество новых заданий, а также лучшие из опубликованных ранее материалов популярной рубрики нашей газеты.

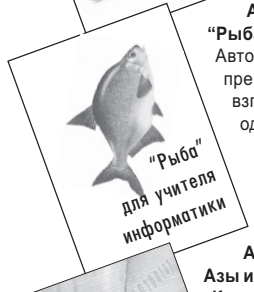


О Г.А. Звенигородском.
Редактор-составитель — Н.А. Юнерман
История становления школьной информатики не может быть написана без страниц, посвященных исследованиям и разработкам Г.А. Звенигородского. И сегодня остается актуальным воплощенное в них единство тематических и педагогических сторон работы с учащимися, живой практики и теоретического осмысления материала. 9 августа 2002 г. Г.А. Звенигородскому исполнилось бы 50 лет.

А.А. Дуванов.
DHTML-конструирование
"Бумажная" версия электронного учебника продолжает роботландский курс гипертекстового конструирования (ранее были опубликованы "HTML-конструирование" и "JavaScript-конструирование"). Новый учебник посвящен созданию динамических интерактивных приложений. В нем изложены основы CSS (каскадные таблицы стилей) и показаны способы управления содержимым страницы при помощи воздействий на гипертекстовую модель документа.

Л.О. Сергеев.
Уроки по теме "Базы данных"
В этом тематическом выпуске мы предложим вниманию читателей цикл уроков по теме "Базы данных". В качестве основного инструмента для изучения этой темы автор предлагает использовать язык SQL. Теоретический материал подкрепляется большим количеством разноуровневых заданий.

Интеллектуальные игры
Что такое спортивное "Что? Где? Когда?" и чем оно отличается от телевизионного? Какие головоломки решают на чемпионатах мира? Во что можно поиграть без компьютера? Это и многое другое, а также вопросы, вопросы, вопросы... в специальном выпуске "Интеллектуальные игры".



А.Г. Гейн.
"Рыба" для учителя информатики
Авторы учебников обычно много говорят о том, что преподавать, и мало про то, как это делать. Свой взгляд на тематическое планирование предлагает один из авторов учебников по информатике.

А.А. Дуванов.
Азы информатики.
Книга 3 — "Пишем на компьютере"
"Бумажная" версия третьей книги нового интерактивного курса для малышей "Азы информатики" (первая книга — "Знакомство с компьютером" — была опубликована в № 1, 2, публикация второй — "В мире информации" продолжается в текущих номерах). Заглавная тема третьей книги (современная обработка текстов) нагружена "анатомией" трех китов информатики: хранение, передача и обработка информации.

Дорогие коллеги! "Информатика" распространяется только по подписке.
Подписаться на нашу газету можно по каталогу "Роспечати", индекс подписки для индивидуальных подписчиков – 32291.

Олимпиады по информатике. Пути к вершине

Лекции читает Е.В. Андреева

Лекция 9. Эффективные алгоритмы на графах

В прошлой лекции мы привели основные определения теории графов и рассмотрели различные задачи поиска кратчайших путей в графах. Другие известные эффективные алгоритмы решения задач на графах будут приведены ниже. Напомним, что алгоритм считается эффективным, если количество операций в нем полиномиально зависит от размерности задачи (в нашем случае — от количества вершин в графе), при этом максимальная степень в полиноме и все коэффициенты фиксированы и не зависят от значения размерности. Подробно о рассматриваемых задачах, а также о некоторых других проблемах, для которых возможно построить эффективное решение, можно прочитать в [1—3].

Обход вершин графа

При решении многих задач, как для ориентированных, так и для неориентированных графов, необходим эффективный метод систематического обхода вершин графа. На практике применяются два принципиально различных порядка обхода, основанные на *поиске в глубину* и *поиске в ширину* соответственно. Начнем рассмотрение с первого из них.

Сначала пометим все вершины графа как непосещенные. Поиск в глубину начинается с произвольной вершины графа, например, с первой, обозначим ее *v*. При этом значение метки *v* меняется на противоположное (вершина уже посещалась). Затем для каждой вершины, смежной с *v*, которая ранее не посещалась, рекурсивно вновь применяется поиск в глубину. Легко показать, что при этом все вершины, достижимые из начальной, то есть образующие одну компоненту связности, будут пройдены. Если некоторые вершины оказались не пройденными, то граф не связан. Для полного его обхода выбираем любую еще не посещенную вершину и продолжаем поиск. Очевидно, что таким образом можно не только проверять связность графа, но и подсчитывать количество компонент связности. Приведем процедуру поиска в глубину и фрагмент основной программы, решающей эту задачу. Считаем, что граф задан с помощью булевской матрицы смежности a (см. предыдущую лекцию), а метятся вершины графа с помощью массива

vert: **array**[1..nmax] **of boolean**.

```
procedure d_f_s(v: byte);
var i: byte;
begin
{ writeln(v); или другая обработка вершины v}
  vert[v] := false;{вершина посещена}
  for i := 1 to n do
    if a[v,i] and vert[i] then
      d_f_s(i)
end;
begin
   ...
  {заполняем матрицу смежности}
  {метим все вершины как непосещенные}
  fillchar(vert, sizeof(vert), true);
  cnt := 0;{счетчик компонент связности}
  for i := 1 to n do
    if vert[i] then
      begin
        inc(cnt);
        d_f_s(i)
      end;
  writeln(cnt)
end.
```

Подсчитаем вычислительную сложность предложенной реализации алгоритма. Так как для каждой вершины процедура d_f_s вызывается ровно один раз, а количество операций в ней пропорционально N — количеству вершин в графе, то количество операций в алгоритме есть $O(N^2)$, а его применимость ограничена лишь размером памяти, необходимой для представления матрицы смежности. Ниже будет приведена и более эффективная реализация данного алгоритма.

Перейдем теперь к рассмотрению поиска в ширину. Свое название он получил из-за того, что при достижении во время обхода любой вершины *v* графа в очередь на рассмотрение попадают сразу все еще не просмотренные вершины, связанные с вершиной *v*. На каждом шаге из начала очереди извлекается один элемент, а в конец добавляются все связанные с ним вершины, еще не находящиеся в очереди. Поэтому элементы метятся как обработанные в момент попадания в очередь, а не извлечения из нее. Так как максимальное количество элементов в очереди равно количеству вершин в графе, организовать ее можно и с помощью одномерного массива (структура данных *очередь* и способы ее представления подробно описаны в [4]). В приведенной ниже программе для этого используются массив list: **array**[1..nmax] **of byte** и целочисленные переменные p и q, являющиеся указателями на индексы элементов,

План публикаций лекций курса "Олимпиады по информатике. Пути к вершине" на "Страницах повышения квалификации"

Номер лекции	Номер газеты
1	38/2001
2	40/2001
3	42/2001
4	44/2001
5	46/2001
6	48/2001
7	6/2002
8	8/2002
9	10/2002
10	12/2002
11	14/2002
12	16/2002

соответствующих началу и концу очереди. Приведем только процедуру поиска в ширину, так как основная программа остается прежней (вызов процедуры поиска в глубину заменяется в ней на вызов поиска в ширину), отметим, что эта процедура нерекурсивна.

```
procedure b_f_s(v: byte);
var i, k, p, q: byte;
begin
  fillchar(list,sizeof(list),0);
  list[1] := v;
  vert[v] := false;
  p := 1; {указатель на начало очереди}
  q := 1; {указатель на конец очереди}
  while p <= q do {пока очередь не исчерпана}
   begin
    {обработать первую в очереди вершину,
     например, writeln(list[p]);}
     k := list[p]; p := p + 1;
     for i := 1 to n do
       if a[k, i] and vert[i] then
        {добавляем i-й элемент в очередь}
         begin
           vert[i] := false;
           q := q + 1;
           list[q] := i
         end
    end
end;
```

Время выполнения алгоритма поиска в ширину такое же, как и для алгоритма поиска в глубину. Однако для ряда задач он может оказаться предпочтительным, например, при проверке наличия циклов в графе или при поиске кратчайшего пути между двумя вершинами в невзвешенном графе. Нерекурсивный характер алгоритма позволит обойти технические ограничения, налагаемые на поиск в глубину размером предоставляемого программе стека.

Для полноты изложения приведем реализации обоих алгоритмов, основанные на другом представлении графа — одномерном массиве списков вершин, связанных с каждой из вершин. Для этого будем использовать такую структуру данных, как динамический связанный список (см. [4]). Так как основная программа остается прежней, приведем лишь описание основных структур данных, модифицированные процедуры поиска и начало программы, в котором и создаются упомянутые списки. Для описания графа используется массив указателей a на начала динамических списков и массив указателей на конечные элементы списков — b (последний нужен лишь для организации поиска в ширину). В основной программе может вызваться любая из описанных процедур. При поиске в ширину в очередь будут помещаться все элементы, связанные с вершиной, находящейся в начале очереди, зато за одну операцию соединения списков. Помечаться же как обработанные вершины будут лишь при удалении из очереди.

```
const nmax = 500;
type ptr = ^el; {указатель на элемент списка}
  el = record
         i: integer; next: ptr
       end;
var   a, b: array[1..nmax] of ptr;
      vert: array[1..nmax] of boolean;
      p: ptr;
      i, j, k, m, n, cnt: integer;
procedure d_f_s(v: integer);
begin
  vert[v] := false;
  while a[v] <> nil do
  {пока список вершин, связанных с v, не исчерпан}
  begin
    if vert[a[v]^.i] then
      d_f_s(a[v]^.i);
    {переходим к следующему элементу списка}
    a[v] := a[v]^.next
  end
end;
procedure b_f_s(v: integer);
var p,q: ptr;
begin
  vert[v] := false;
  p := a[v]; {указатель на начало очереди}
  q := b[v]; {указатель на конец очереди}
  while p <> nil do
  {пока очередь не исчерпана}
    begin
      if vert[p^.i] then
        begin
          vert[p^.i] := false;
          {добавим в очередь сразу все
           элементы, связанные с p^.i}
          q^.next := a[p^.i];
          q := b[p^.i];
        end;
      p := p^.next {меняем начало очереди}
    end;
end;
begin {основная программа}
  readln(n); {n - количество вершин}
  for i := 1 to n do a[i] := nil;
  readln(m); {m - количество ребер}
  for i := 1 to m do {считываем ребра}
  begin
    read(j, k); {ребро из j в k}
    {добавляем элемент в список a[j]}
    new(p);
    p^.i := k;
    p^.next := a[j];
    if a[j] = nil then b[j] := p;
    a[j] := p;
    {добавляем элемент в список a[k]}
    new(p);
    p^.i := j;
    p^.next := a[k];
    if a[k] = nil then b[k] := p;
    a[k] := p
  end;
  {структура для описания графа создана}
  …{далее программа совпадает с первой}
end.
```

Время выполнения каждой из приведенных процедур теперь составляет $O(M)$, то есть пропорционально количеству ребер в графе, что может дать существенный выигрыш для разреженных графов.

Поиск эйлерова пути в графе

Эйлеровым путем в графе называется произвольный путь, проходящий через каждое ребро графа в точности один раз.

Теорема 1. Эйлеров путь в неориентированном графе существует тогда и только тогда, когда граф связан и содержит не более двух вершин нечетной степени.

Степенью вершины v в неориентированном графе называют количество вершин, смежных с *v*. Таким образом, проверить эйлеровость графа не составит труда. Причем очевидно, что одна вершина с нечетной степенью в графе существовать не может, поэтому допустимым является лишь наличие двух таких вершин, одна из которых должна быть выбрана в качестве начала эйлерова пути, а вторая окажется его концом. Мы можем соединить эти вершины фиктивным ребром, сделав тем самым степени всех вершин четными, и, не снижая общности изложения, решать задачу поиска самого пути в предположении отсутствия вершин нечетной степени.

Алгоритм построения эйлерова пути начинает свою работу с произвольной вершины *v*, строя путь с началом в *v*, причем вершины этого пути помещаются в стек, а ребра удаляются из графа. Когда этот путь уже нельзя будет удлинить, включив в него новую вершину, то это означает, что из графа был удален некоторый цикл, а степень всех вершин по-прежнему осталась четной. Текущая вершина записывается в результат и извлекается из стека. Верхний элемент стека становится текущим, и путь ищется уже из него. Процесс продолжается до тех пор, пока стек не станет пустым. Приведем нерекурсивную схему реализации алгоритма, не зависящую от способов представления графа, организации стека и формы записи результата.

```
v := произвольная вершина графа, обычно первая;
STACK ← v; {вершина заносится в стек}
while (STACK не пуст) do
begin
  v := верхний элемент стека STACK;
  if (в графе еще есть вершины,
      связанные с v) then
    begin
      u := любая вершина, связанная с v;
      STACK <= u;
      удаляем ребро {v — u} из графа
    end
  else
    begin
      v ← STACK; {вершина удаляется из стека}
      RES ← v {вершина заносится в результат}
    end
end;
```

В схеме программы показано, что если связанный граф без вершин нечетной степени представлен массивом списков, то количество операций в алгоритме есть $O(M)$, т.е. пропорционально числу ребер. Программа работает с такой структурой данных, как стек, основной характеристикой которой является то, что добавленный в стек элемент оказывается в нем верхним, то есть первым же поступит на обработку или удаление. Такая схема доступа к памяти автоматически предоставляется программисту механизмом реализации рекурсии. Поэтому перепишем приведенный алгоритм в виде рекурсивной подпрограммы, для простоты использующей матрицу смежности в качестве способа описания графа. Результат будем размещать в массиве `res`, указывать на первый свободный элемент, в котором будет глобальная переменная `p`.

```
procedure search(v: byte);
var j: byte;
begin
  for j := 1 to n do
    if a[v, j] = 1 then
      begin
        a[v, j] := 0; a[j, v] := 0;
        search(j)
      end;
  inc(p);
  res[p] := v
end;
```

Рассмотрим теперь особенности алгоритма поиска эйлерова пути в ориентированном графе. Будем называть степенью вершины *v* ориентированного графа, применительно к этой задаче, разницу между количеством ребер, выходящих из вершины *v*, и количеством ребер, входящих в нее. Тогда условие эйлеровости можно сформулировать так.

Теорема 2. Эйлеров путь в ориентированном графе существует тогда и только тогда, когда граф связан, все вершины, кроме, быть может, двух, имеют нулевую степень. Допустимым является лишь наличие одной вершины степени 1 и одной — степени −1.

Очевидно, что при наличии двух вершин ненулевой степени в ориентированном графе начинаться эйлеров путь должен в вершине со степенью 1, а заканчиваться — в вершине со степенью −1. Интересно, что схема поиска самого пути при этом остается неизменной. А в процедуре `search` следует лишь убрать обнуление элемента матрицы смежности `a[j, v]`, так как в ориентированном графе каждому ребру соответствует один элемент матрицы, а не два.

Попробуйте теперь решить следующую задачу, которая была рекомендована для областных олимпиад по информатике в 2001/2002 уч. году.

Задача 1. Для решения транспортной проблемы в некотором городе до недавнего времени использовались N ($N \le 100$) автобусных маршрутов. Каждый маршрут начинался на одной из M ($M \le 200$) площадей и там же заканчивался. В процессе проезда по маршруту автобус мог несколько раз проезжать одну и ту же площадь, но не мог проезжать более одного раза по одной и той же улице в том же самом направлении.

В определенный момент местные власти решили сократить количество автобусных маршрутов в городе до одного. По их мнению, должен был остаться лишь один кольцевой маршрут, который проходил бы по всем улицам, по которым раньше проходили автобусные маршруты, причем в том же направлении (но не обязательно в том же порядке). Если по каким-либо улицам автобусы ездили в обоих направлениях, то и новый маршрут должен проходить по этим улицам в обоих направлениях, но дважды по одной улице в одном направлении автобус проезжать не может. По тем улицам и в тех направлениях, по которым раньше автобусы не ездили, новый маршрут проходить не должен. Требуется написать программу, которая для заданных исходных данных определяет требуемый автобусный маршрут.

Указание. Несмотря на небольшую на первый взгляд размерность задачи и возможность использования матрицы смежности, для записи результата может понадобиться массив, состоящий из 40 000 элементов, который придется либо создавать в

динамической памяти (см. лекцию 3), либо не заводить совсем, выводя результат на печать по ходу его получения. По той же причине для анализа полного графа рекурсию использовать невозможно, размера программного стека для этих целей явно недостаточно. Поэтому придется написать нерекурсивный вариант процедуры `search` для ориентированного графа.

Построение минимального остова во взвешенном неориентированном графе

Пусть дан связанный неориентированный граф, для каждого ребра которого задан неотрицательный вес. Задача состоит в нахождении такого подмножества ребер этого графа, которое по-прежнему связывает все вершины, а суммарный вес ребер из этого подмножества минимален. Такое подмножество обязательно окажется *деревом* (т.е. графом без циклов), ведь в любом цикле одно ребро можно удалить, не нарушая связности графа. Поэтому искомый подграф называют "минимальным покрывающим деревом", или "остовом минимального веса".

Алгоритмы, решающие эту задачу, относятся к классу так называемых *жадных*. Наиболее простым в реализации является алгоритм Прима. Он в некоторой степени похож на алгоритм Дийкстры поиска кратчайшего пути во взвешенном графе. Формирование остова начинается с произвольной вершины графа. Первым в остовное дерево включается ребро наименьшего веса, выходящее из выбранной вершины. На каждом шаге к дереву добавляется ребро наименьшего веса среди ребер, соединяющих вершины этого дерева с вершинами, пока в дерево не вошедшими. При реализации важно уметь быстро выбирать требуемое ребро минимального веса. Для того чтобы на каждом шаге не пересчитывать расстояние от текущего остовного дерева до всех не вошедших в него вершин, эти расстояния удобно хранить в линейном массиве (в программе `d`), пересчитывая его значения после добавления в остов нового элемента. Как и ранее, для пометки вершин, уже вошедших в остовное дерево, будем использовать булевский массив `vert`. Следующие две процедуры показывают, как для произвольного графа, вес ребер которого задается с помощью функции `a(i, j)`, а отсутствие ребра между двумя вершинами обозначается весом, равным ∞ (конкретное числовое значение для этого параметра подбирается исходя из условия задачи), построить остовное дерево.

```
procedure calc(i: integer);
{пересчитывает расстояние до остова,
 i — вершина, включенная в остов последней}
var j: integer;
begin
  for j := 1 to n do
    if not vert[j] then
      if d[j] > a(i, j) then
        begin
          d[j] := a(i, j);
          res[j] := i
        end
end;
procedure build;
{строит минимальный остов}
var i, j, imin: integer;
    min: extended;
begin
  fillchar(vert, sizeof(vert), false);
  for i := 1 to n do d[i] := ∞;
  vert[1] := true;
  calc(1);
  for j := 1 to n - 1 do
  {остов состоит из n - 1 ребра}
    begin
      min := ∞;
      for i := 1 to n do
        if not vert[i] then
          if min > d[i] then
            begin
              min := d[i];
              imin := i
            end;
      vert[imin] := true;
      calc(imin);
      {в остов вошло ребро imin - res[imin]}
      writeln(imin,' ', res[imin])
    end
end;
```

Используя приведенные процедуры, попробуйте решить следующую задачу (аналогичная задача "*Highways*" в англоязычной формулировке предлагалась на полуфинале студенческого чемпионата мира по программированию в Санкт-Петербурге в 1999 г., тесты к ней можно найти на сайте **neerc.ifmo.ru**).

Задача 2. Правительство России проанализировало систему дорог в России и выяснило, что только часть из них соответствует международным стандартам. Было принято решение создать систему скоростных автодорог, соединяющих крупные города. Сеть дорог требуется спроектировать так, чтобы по ним и по уже существующим хорошим дорогам можно было проехать из любого данного города в любой другой, однако затраты на строительство должны быть минимальны (будем считать, что минимальные затраты соответствуют минимальной суммарной длине построенных дорог).

Первая строка входного файла содержит целое число городов N ($1 \le N \le 750$), которые должны быть соединены скоростными дорогами. Следующие N строк содержат по 2 целых числа, разделенных пробелами, x_i и y_i — декартовы координаты городов в километрах от Москвы. Следующая строка содержит целое число M ($0 \le M \le 1000$), соответствующее количеству уже имеющихся хороших дорог между городами. В каждой из следующих M строк содержится пара номеров городов, которые уже соединены хорошей дорогой (номера городов соответствуют порядку описания их координат во входном файле начиная с 1).

Запишите в выходной файл описание всех дорог, которые необходимо построить, каждое в отдельной строке. Каждую дорогу описывать номерами городов, разделенных пробелами, которые эта дорога соединяет (аналогично входному файлу). Если существующих хороших дорог уже достаточно, то выходной файл должен быть пустым.

Построение максимального паросочетания в двудольном графе

Граф называют *двудольным*, если множество его вершин можно разбить на 2 непересекающихся множества (каждая вершина должна обязательно войти в одно из

этих множеств), причем каждое ребро графа начинается в одном из этих множеств, а заканчивается в другом. Максимальное паросочетание — это максимально возможное количество ребер, не имеющих общих концов.

Примером может служить задача, благодаря которой и была поставлена эта проблема: как заключить максимально возможное число удачных браков среди пар, симпатизирующих друг другу, если один человек может симпатизировать сразу нескольким лицам другого пола.

Пусть в графе построено некоторое паросочетание, такое, что из оставшихся свободными вершин новые пары составить уже нельзя (оно не обязательно максимально). Чередующейся цепью относительно этого паросочетания назовем такую последовательность вершин x_0, y_1, x_2, y_3, ..., x_k, y_{k+1}; $k > 0$, что вершины x_i принадлежат одному из множеств двудольного графа и все, кроме x_0, входят в текущее паросочетание, а вершины y_i принадлежат другому множеству и все, кроме y_{k+1}, входят в текущее паросочетание, причем паросочетанию принадлежат ребра (y_i, x_{i+1}), $i = 1, ..., k-1$, а ребра (x_i, y_{i+1}), $i = 0, ..., k$ в графе присутствуют, но в текущее паросочетание не входят. Тогда мы можем исключить из паросочетания первую группу ребер и включить вторую, увеличив тем самым количество пар на единицу. Следующая теорема позволяет применить описанную процедуру при решении задачи нахождения максимального паросочетания.

Теорема 3. Паросочетание в двудольном графе является максимальным тогда и только тогда, когда относительно него не существует чередующейся цепи, а не вошедшие в паросочетание ребра добавить к нему невозможно.

Наиболее подробно теория паросочетаний изложена в [5]. К сожалению, несмотря на наличие конструктивного алгоритма построения максимального паросочетания, его реализации в литературе по алгоритмам на графах достаточно громоздки. Программа, предлагаемая ниже, основана на схеме решения задачи о стабильных браках, приведенной в [4]. Несмотря на то, что механизм чередования цепочек в ней присутствует, логика выполняемых операций несколько отлична от описанной выше. В основной программе каждой вершине одного из множеств мы пытаемся найти допустимую пару (очевидно, что если каждой вершине пара будет найдена, то максимальное паросочетание построено). Делается это с помощью рекурсивной функции `try`. Если для вершины `j` свободной пары в противоположном множестве нет, то делается попытка построить чередующуюся цепочку, начинающуюся с `j`-й вершины.

```
const max =…{размер большего из двух множеств}
var v: array[1..max] of boolean;
  res: array[1..max] of integer;
  n, m, i, j, cnt: integer;
function try (j: integer): boolean;
{пытается найти вершине j пару}
var i: integer;
begin
  if v[j] then
  {j в текущем паросочетании уже просмотрена}
    begin
      try := false; exit
    end;
  v[j] := true;
  for i := 1 to n do
    if a(i, j) {ребро между i и j существует}
      and ((res[i] = 0) {у i еще нет пары}
      or{пару i можно пристроить к другой вершине,
    для чего нужно построить чередующуюся цепочку}
      try(res[i])) then
        begin
          try := true;
          res[i] := j;
          exit
        end;
  try := false
end;

begin
 …{здесь вводим описание графа}
  fillchar(res, sizeof(res), 0);
  cnt := 0; {счетчик ребер в паросочетании}
  for j := 1 to m do
  {каждой вершине одного из множеств
   пытаемся найти пару}
    begin
      fillchar(v, sizeof(v), false);
      if try(j) then inc(cnt)
    end;
  writeln(cnt);
  for i := 1 to n do
    if res[i] <> 0 then
      write(i, ' ', res[i])
end.
```

Задача, при решении которой используется описанный алгоритм, предлагалась на I Всероссийской командной олимпиаде школьников по программированию в 2000 г.

Задача 3. "*Кубики*".

Родители подарили Пете набор детских кубиков. Поскольку Петя скоро пойдет в школу, они купили ему кубики с буквами. На каждой из шести граней каждого кубика написана буква. Теперь Петя хочет похвастаться перед старшей сестрой, что научился читать. Для этого он хочет сложить из кубиков ее имя. Но это оказалось довольно сложно сделать — ведь разные буквы могут находиться на одном и том же кубике и тогда Петя не сможет использовать обе буквы в слове. Правда, одна и та же буква может встречаться на разных кубиках.

Даны набор кубиков и имя сестры. Выясните, можно ли выложить ее имя с помощью этих кубиков, и если да, то в каком порядке следует выложить кубики.

На первой строке входного файла находится число N ($1 \le N \le 100$) — количество кубиков в наборе у Пети. На второй строке записано имя Петиной сестры — слово, состоящее только из больших латинских букв, не длиннее 100 символов. Следующие N строк содержат по 6 букв (только большие латинские буквы), которые написаны на соответствующем кубике.

Литература

1. *Ахо А.А., Хопкрофт Д.Э., Ульман Д.Д.* Структуры данных и алгоритмы. М.: Вильямс, 2000.

2. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы. Построение и анализ. М.: МЦНМО, 2000.

3. *Липский В.* Комбинаторика для программистов. М.: Мир, 1988.

4. *Вирт Н.* Алгоритмы и структуры данных. СПб.: Невский диалект, 2001.

5. *Асанов М.О.* Дискретная оптимизация. Екатеринбург: УралНаука, 1998.

Введение в профессию "учитель информатики"

Лекции читает А.Г. Гейн

Лекция 3. Проблема мотивации: локальный аспект

...И опыт, сын ошибок трудный,
И гений, парадоксов друг.
А.С. Пушкин

Создание у учащихся мотивации к изучению того или иного фрагмента учебного материала и курса в целом — краеугольный камень методики преподавания любого предмета, в частности, информатики. В предыдущей лекции мы обсуждали проблему создания мотивации к изучению курса в целом. Но важно, чтобы мотивированными были изучение каждой отдельно взятой темы, введение каждого понятия, овладение каждым умением, приобретение каждого навыка. Такую мотивацию мы и назвали локальной.

Конечно, создание локальной мотивации нередко связано с личным педагогическим талантом учителя и даже просто с его обаянием. Но это относится к той грани педагогической деятельности, которую можно назвать одним словом — *искусство*. Нас же интересует больше, как принято теперь говорить, технология, т.е. вполне конкретные приемы и методы создания такой мотивации. Ниже мы будем их описывать и приводить примеры ситуаций использования, взятые из преподавательской практики и школьных учебников. Сразу оговоримся, что порядок, в котором приводятся приемы создания локальной мотивации, никак не свидетельствует об их предпочтительности.

Прием первый: апелляция к жизненному опыту учащихся

Этот прием заключается в том, что учитель обращает внимание учащихся на хорошо знакомые учащимся ситуации, понимание сути которых возможно лишь при условии изучения предлагаемого материала. Вот иллюстрирующий пример (мы, естественно, приводим тот фрагмент урока, который касается объяснения нового материала; кроме того, в приводимых ниже примерах диалогов учителя с классом мы не будем персонифицировать учащихся, указывая везде "*Ученики*", подразумевая, что отвечает кто-то один из учащихся).

Тема урока. Понятие информационной модели и ее структура.

Цель урока. Ввести понятие информационной модели. Обосновать необходимость этого понятия. Указать ситуации применения этого понятия. Выявить общую структуру информационной модели.

Учитель. Каждому из вас наверняка приходилось делать подарки: другу на день рождения, или родителям, или еще кому-нибудь. Как вы этот подарок организовывали? Пусть, скажем, речь идет о подарке другу.

Ученики. Шли в магазин и покупали[1].

Учитель. Как, в первый попавшийся магазин?

Ученики. Ну почему же? В общем-то обычно знаешь, что хотел бы иметь твой друг. Вот то и даришь.

Учитель. Стоп! Остановимся и подумаем, что происходит. Чтобы решить жизненную задачу, в данном случае "Купить подарок другу", нам потребовалась информация, в данном случае о том, что бы ему хотелось иметь. Можно сказать, что эта информация — существенный фактор для решения нашей задачи. Никто не возражает против этого? Тогда пойдем дальше. Предположим, что ваш друг горит желанием иметь собственный автомобиль "Мерседес". Вы ему подарите?

(В классе оживление.)

Учащиеся: Неплохо бы, да где ж деньги взять?

Учитель. Правильно ли я понимаю, что у нас появился второй существенный фактор — финансовые возможности дарителя[2]?

Ученики. Ну конечно.

Учитель. Хорошо. А если ваш друг мечтает о трубке, чтобы выглядеть как настоящий морской волк? Неужели вы ему подарите трубку, хотя знаете, что курить вредно?

(Возникает дискуссия. Одни утверждают, что забота о здоровье — личное дело каждого, другие высказывают мнение, что такой подарок не простят родители, третьи категорически выступают за то, что подарок не должен приносить вред.)

Учитель. Остановились. Посмотрим, какие факторы у нас еще появились. Я понял, что их по крайней мере два: мнение родителей о подарке и полезность подарка для вашего друга. Правда, кое-кто считает, что ни один из этих факторов или какой-то один из них не является существенным для решения задачи.

Подведем первые итоги решения жизненной задачи "Купить подарок другу". Первый шаг на пути к решению — выделение существенных факторов. Но ясно, что этого недостаточно. Надо каждый фак-

План публикации лекций курса "Введение в специальность "учитель информатики" на "Страницах повышения квалификации".

Номер лекции	Номер газеты
1	6/2002
2	8/2002
3	10/2002
4	12/2002
5	14/2002
6	16/2002

[1] Ответы здесь, конечно, могут быть самыми разными. Например, "делали своими руками". Но на существо диалога это почти не влияет.

[2] Здесь очень важно не дать классу увести на обсуждение другой жизненной задачи — где взять деньги? Это тоже интересная задача, но обсуждение интересных жизненных задач не является целью урока, а лишь служит средством достижения поставленных целей.

тор описать тем или иным образом. Финансовый фактор описать легко — это число, показывающее стоимость предполагаемого подарка. А как описать фактор "желание друга"?[3] Это можно сделать, например, так: в списке предполагаемых подарков указывать "Да" или "Нет". Вот какое описание нашей задачи тогда получится:

Предполагаемый подарок	**Фактор**			
	Желание друга	Стоимость	Полезность	Одобрение родителей
А/м "Мерседес"	Да	30 000 у.е.	Да	?
Мотоцикл "Харлей"	Да	20 000 у.е.	Да	Нет
Аквариум с рыбками	Да	1050 руб.	Нет	Да
Книга о вкусной и здоровой пище	Нет	130 руб.	Да	Да

Что же у нас получилось?

Далее обсуждается, что мы заменили не вполне ясный объект "подарок другу" информационным описанием в виде набора существенных факторов и их значений. Такое описание — это и есть информационная модель объекта. Делается обобщающий вывод, что решение любой жизненной задачи начинается с построения информационной модели. Структура информационной модели — это набор существенных факторов и связи между ними. Но это уже собственно изложение нового материала, мотивационно опирающееся, как видно из примера, на жизненный опыт учащихся.

Учитель, чувствующий себя абсолютно уверенно в данной теме, может пойти на еще более эффектное мотивационное введение, предложив учащимся самим выбрать жизненную задачу для обсуждения того, с чего начинается процесс ее решения. В этом случае надо быть готовым к весьма остро звучащим постановкам, например, "Как удачно выйти замуж?". И уклониться от обсуждения этой задачи уже нельзя, поскольку объявлено, что решение любой задачи начинается с построения информационной модели. Вот и здесь надо ставить перед учащимися вопрос, какими существенными факторами определяется удачность замужества. Богатством избранника? А если богатый, но старый? Красотой кандидата в женихи? А если красивый, но жестокий? Еще чем-то? Кстати, практика показывает, что в таких обсуждениях ученики далеко не первым называют, пожалуй, один из главных факторов — любить и быть любимым. Но, по счастью, все же до него дело, как правило, доходит.

Отметим, что апелляция к собственному опыту учащихся всегда сопровождается, как говорят психологи, рефлексией, т.е. анализом собственных действий, собственного состояния, своих ощущений. Это накладывает определенные ограничения на выбор того, что может использоваться для создания мотивации, в частности, уже потому, что такая рефлексия должна вызывать положительные эмоции.

Прием второй: ссылка на то, что приобретаемое сегодня знание понадобится при изучении какого-то последующего материала, важность овладения которым сомнения не вызывает

Этот прием применяется настолько часто и широко, что мы не будем на нем подробно останавливаться, а лишь приведем один пример его использования в учебнике [1] (см. с. 41).

Созданием модели завершается первый этап решения задачи с помощью ЭВМ. Для того чтобы ЭВМ произвела необходимые вычисления и получила ответ, нужно составить для нее четкую инструкцию, строго указать необходимую последовательность действий. Такая инструкция называется **алгоритмом** решения задачи. Составление алгоритма — второй этап решения задач с помощью ЭВМ. Значит, теперь нужно научиться составлять алгоритмы для ЭВМ. О том, как составлять алгоритмы, и рассказывается в этой главе.

Как мы видим, учащимся объявляется: чтобы научиться применять компьютер к решению задач, придется познакомиться с понятием алгоритма и овладеть умениями создавать алгоритмы.

Прием третий: создание проблемной ситуации

В педагогической литературе этот прием рассматривается едва ли не как самый главный и универсальный. Состоит он в том, что перед учащимся ставится некоторая проблема, и, преодолевая ее, ученик осваивает те знания, умения и навыки, которые ему и надлежит усвоить согласно программе курса. Но все дело в том, что ученик должен хотеть решать поставленную перед ним проблему, она должна быть ему интересна. А само по себе создание проблемной ситуации такого интереса вовсе не гарантирует. Интерес же может возникать, например, в силу парадоксальности описываемой в этой проблеме ситуации. Вот соответствующий пример.

Тема урока. Компьютерное моделирование физических процессов.

Цель урока. Ввести понятия компьютерной модели и компьютерного эксперимента. Подготовить компьютерный эксперимент с моделью движения тела в среде с сопротивлением.

Учитель. Каждый из вас не раз, наверно, попадал под теплый веселый летний дождь. Или под осенний моросящий дождь. Давайте прикинем, какую скорость имеет около поверхности Земли капля, сорвавшаяся, скажем, с высоты 8 км. На уроках физики вы получали формулу для скорости тела при его движении в поле силы тяжести, если начальная скорость была нулевая:

$$v = \sqrt{2gh},$$

где h — высота, с которой тело начинает движение к поверхности Земли. Поскольку мы только хотим прикинуть скорость, будем считать $g \approx 10$ м/с^2.

(Ученики подсчитывают скорость и получают значение $v \approx 400$ м/с.)

[3] Здесь, конечно, снова возможно (и даже весьма желательно) подключение класса к обсуждению способа задания этого фактора. Мы для экономии места приводим готовый ответ.

Учитель. Но капля, летящая с такой скоростью, подобна пуле, ее удар пробивал бы насквозь оконное стекло. А этого, однако, не происходит. В чем же дело?

Парадокс налицо. Как его разрешить, обычно интересно всем. Но рассказ об этом — не тема данной лекции[4].

Прием четвертый: использование занимательного сюжета

Занимательность (но не развлекательность!) — это сильный прием. Повторим то, что уже говорилось в предыдущей лекции: в создании мотивации интерес всегда имеет приоритет над прагматикой — человек готов тратить значительные усилия на усвоение совершенно интересного, хотя, быть может, и бесполезного знания, но как трудно заниматься каким-либо делом только из осознания его необходимости!

Иллюстрирующий пример, который приведен ниже, в свое время лег в основу изложения соответствующего вопроса в учебнике [1], с. 32—34.

Тема урока. Соотношение между моделью и прототипом. Проблема адекватности модели.

Цель урока. Указать на неоднозначность модели для одного и того же прототипа. Ввести понятие адекватности модели.

Учитель. В одном сборнике старинных задач я нашел такую задачу:

Двумя ударами топора разрубить подкову на наибольшее число частей, не перекладывая части после удара.

Ученики. Это легкая задача: ответ — 5 частей.

(В обоснование приводят следующий рисунок.)

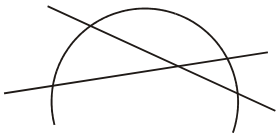


Рис. 1

Учитель. Это, конечно, не подкова, а ее модель — дуга окружности. А удары топора, по-видимому, моделируются двумя прямыми линиями. Все согласны, что мы построили модель объекта и процесса "разрубания" подковы?

Ученики. Да, конечно.

Учитель. А вот ответ в сборнике дан другой: 6 частей! В чем тут дело?

(В классе оживление.)

Ученики: Так ведь подкова не так выглядит!

(Делают новый рисунок.)

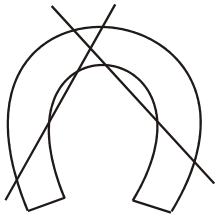


Рис. 2

[4] Данный пример — несколько измененное введение к одному из параграфов нового учебника "Информатика для 10—11-х классов естественно-научного профиля" (авторы: А.Г. Гейн, А.Б. Ливчак, Н.А. Юнерман), готовящегося в настоящее время к изданию. В нем подробно описываются построение соответствующей компьютерной модели и проведение компьютерного эксперимента.

Учитель. Что же произошло? Ответ поменялся, потому что изменилась модель объекта. Она стала учитывать еще один фактор — наличие у подковы ширины. Но ведь у подковы есть еще и толщина. Она не может сказаться на результате?

(Ученики строят "объемную" модель подковы.)

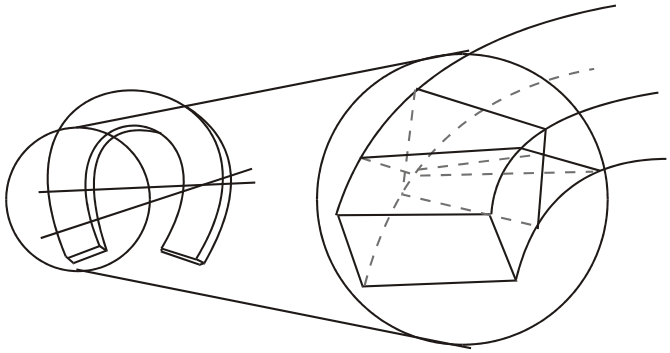


Рис. 3

Учитель. Вот видите: ответ уже стал равен 7. Может, можно учесть еще какой-нибудь фактор? Что еще есть у подковы?

Ученики. Дырки.

Учитель. Замечательно. Дома опробуете модель с дырками[5]. А мы постараемся сделать выводы из того, что уже имеем...

Мы же прервем обсуждение, поскольку к проблеме мотивации оно отношения не имеет.

Прием пятый: ролевой подход

В этом случае ученику (или группе учащихся) предлагается выступить в роли того или иного действующего лица, например, формального исполнителя алгоритма. Исполнение роли заставляет сосредоточиться именно на тех существенных условиях, усвоение которых и является учебной целью. Если, скажем, речь идет об усвоении конструкции "цикл", то это точное исполнение команд, посредством которых данная конструкция реализована. Да и при изучении просто понятия формального исполнения алгоритма ученик в роли исполнителя должен сосредоточиться именно на точном и совершенно формальном, т.е. без вопросов, относящихся к цели действия, исполнении каждого действия в алгоритме.

Впрочем, ролевой подход мотивационно используется и во многих других ситуациях. Вот примеры из разных учебников.

"Представьте себе, что у вас есть небольшая фирма по выдаче напрокат видеофильмов. К вам постоянно обращаются клиенты в надежде получить интересующую их запись. Кому-то требуется последний хит сезона, у кого-то есть любимый актер, другие испытывают ностальгию по музыке давно прошедших лет. Вы, скажем, не бедствуете. Одно удручает вас: все-таки медленно обслуживаются клиенты, поскольку нелегко в длинных списках отыскать нужное название. Мало помогают и выложенные на прилавок списки записей" ([2], с. 31).

Это начало параграфа, посвященного рассказу о базах данных.

[5] Сообщим читателю, что ответ в этом случае 11.

"Представьте, что вы стали директором завода и, изучив спрос, решили организовать участок для производства двух видов товаров повышенного спроса — мясорубки и скороварки. Для краткости обозначим эти товары буквами "А" и "Б". Допустим, что вам удалось заключить договоры с другими предприятиями на поставку ресурсов (металла, электроэнергии и т.п.) и выделить определенное число рабочих. Изучение рыночной конъюнктуры позволило определить минимальные объемы производства для каждого изделия. Всякий хороший директор стремится к тому, чтобы прибыль была наибольшей. Будем считать это и вашей задачей" ([1], с. 169—170).

По этому вступлению даже не скажешь, что речь пойдет об использовании табличной формы представления данных.

Прием шестой: деловая игра

Деловую игру можно рассматривать как развитие ролевого метода: в деловой игре у каждого участника своя вполне определенная роль. Однако организация деловой игры требует многосторонней и тщательной подготовки, рассмотрение которой никак не относится к теме данной лекции. Отметим только, что две компьютерные деловые игры описаны в [3].

Приведенный список приемов создания локальной мотивации вовсе не претендует на полноту и завершенность. Да и каждый из этих приемов мог бы быть подвергнут еще большей детализации. Но нам представляется, что это каждый читатель сделает уже сам, руководствуясь своим педагогическим вкусом.

Вопросы и задания

1. Перед вами фрагменты первого урока по теме "Язык как средство хранения, передачи и обработки информации". Для каждого из фрагментов укажите, какой способ создания мотивации был выбран учителем (отметьте соответствующие фрагменты указанными далее цифрами):

1 — апелляция к необходимости знания материала при изучении других тем;

2 — апелляция к опыту учащихся;

3 — создание проблемной ситуации;

0 — мотивация в данном фрагменте не создавалась.

а) *Учитель (в 5-м классе):* Ребята, вы все знаете, что компьютер изобрели прежде всего для того, чтобы освободить человека от решения громоздких вычислительных задач. Давайте попросим наш компьютер найти сумму 2 + 2. Вы, конечно, ответ знаете, поэтому мы как бы устроим нашему компьютеру проверку. Как нам это сделать?

Ученица: Надо на клавиатуре набрать 2 + 2.

Учитель: Что ж, попробуйте.

Ученики набирают, компьютер никак не реагирует.

Ученик: Компьютер ведь не знает, что ему делать с этим выражением. Надо, как в учебнике математики, написать задание: вычислить значение 2 + 2.

Учитель: Хорошо, попробуем так.

Ученики пробуют, компьютер не реагирует.

Кто-то из учеников: Ну и глупый же этот компьютер! Ему русским языком объясняют, что надо делать, а он и не шевелится.

Учитель: Ну почему же глупый? А представьте себе, что кто-нибудь обратится к вам на незнакомом языке и предложит что-то сделать. Вы будете знать, что именно надо делать?

Ученик: А, так он русского языка не понимает!

Учитель: Конечно. Ведь у компьютеров свой язык, и, чтобы они вас понимали, этот язык надо выучить и хорошо знать свойства этого языка. Вот мы и займемся изучением свойств языка...

б) *Учитель (в 10-м классе):* Люди постоянно обмениваются информацией. Скажите, как это происходит?

Ученик 1: Разговаривают друг с другом, пишут письма...

Учитель: А еще?

Ученик 2: Смотрят телик.

Учащиеся приводят еще несколько примеров обмена информацией.

Учитель: Хватит, хватит примеров. Теперь проанализируйте их и ответьте все-таки на мой вопрос: как, с помощью чего происходит обмен информацией?

После нескольких попыток учитель добивается нужного ответа и резюмирует: "Значит, для обмена информацией нужен язык, который был бы понятен как источнику информации, так и ее приемнику. Вот давайте и займемся изучением вопроса, каким должен быть язык, чтобы служить средством для передачи информации".

в) *Учитель (9-й класс):* Все вы помните, что информатика — это наука о способах получения, хранения, передачи и обработки информации. Но любой из этих процессов возможен лишь тогда, когда информация представлена в виде сообщения на том или ином языке. Поэтому, чтобы изучать свойства информации и процессы ее обработки, надо знать свойства языка, посредством которого эта информация фиксируется.

2. Проанализируйте с психолого-педагогической точки зрения удачность выбора подхода к созданию мотивации во фрагментах, приведенных в пунктах а) — в) задания 1 (наиболее удачный отметьте слева цифрой 3, далее по убыванию 2 и 1).

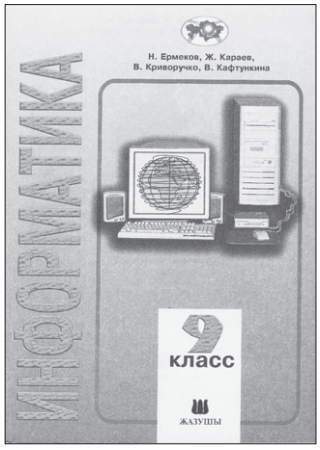
3. Автор этой лекции убежден, что для любой темы, изучаемой в общеобразовательном курсе информатики, может быть использован любой из перечисленных выше приемов создания мотивации (кроме, быть может, деловой игры, поскольку ее проведение требует выполнения еще целого ряда условий). Выберите какую-либо тему и попытайтесь подтвердить этот тезис, придумав для каждого из приемов создания мотивации соответствующий план педагогических действий для учителя.

Литература

1. *Гейн А.Г., Линецкий Е.В., Сапир М.В., Шолохович В.Ф.* Информатика: Учебник для 8—9-х классов общеобразовательных учреждений. М.: Просвещение, 1994, 256 с.

2. *Гейн А.Г., Сенокосов А.И., Шолохович В.Ф.* Информатика: Учебник для 7—9-х классов общеобразовательных учебных заведений. М.: Дрофа, 1997, 224 с.

3. Основы экологии и природопользования. Компьютерный курс: Учебное пособие для 9—10-х классов общеобразовательных учреждений / Шолохович В.Ф., Гейн А.Г., Комов С.В. и др. М.: Просвещение, 1995, 127 с.



Моделирование

Н.Турлынович,
Казахстан

Продолжение. Начало в № 8/2002

Игра "Жизнь"

Что наша "Жизнь"? Игра!

Игру "Жизнь" придумал в конце 60-х годов американский математик Джон Хортон Конуэй. Она сразу стала популярной, поскольку позволяет выполнять интересные исследования и получать неожиданные результаты. Возникающие в процессе игры ситуации очень похожи на реальные процессы, происходящие при зарождении, развитии и гибели колоний живых организмов. Данная игра относится к категории моделирующих игр, которые в той или иной степени имитируют реальную жизнь.

Игра моделирует жизнь поколений гипотетической колонии живых клеток, которые выживают, размножаются или погибают в соответствии со следующими правилами:

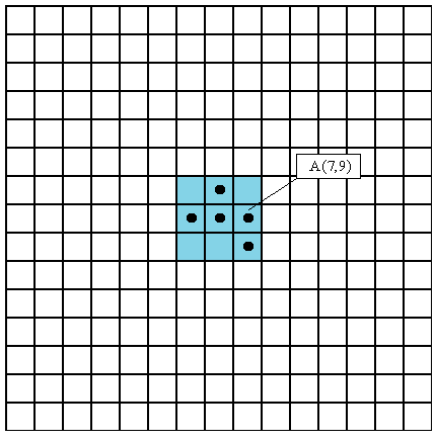
1. **Выживание**. Каждая фишка, имеющая двух или трех из восьми соседей, выживает и переходит в следующее поколение.

2. **Гибель**. Каждая фишка, у которой только один сосед или вовсе ни одного, погибает в изоляции. Если фишка имеет четырех и более соседей, она погибает от перенаселения.

3. **Рождение**. В любой пустой позиции, у которой ровно три соседа, в следующем поколении появляется новая фишка.

Рождение и гибель происходят одновременно. Вместе взятые, они образуют одно поколение, или, как говорят, один шаг эволюции.

Смоделируем этот процесс.



В качестве поля используем таблицу 15 × 15. Начальную конфигурацию фишек расположим в центральных клетках 3 × 3, как показано на рисунке.

Для того чтобы в памяти ЭВМ зарезервировать место для данной таблицы в языке QBasic, можно использовать оператор `DIM (N, M)`,
где `N` — количество строк,
`M` — количество столбцов таблицы.
Каждый элемент полученного двухмерного массива имеет обозначение `A(I, J)`, где `I` — номер строки, `J` — номер столбца.

Например, `A(7, 9)` располагается в 7-й строке и 9-м столбце таблицы (см. рисунок). Тогда для заполнения центральных клеток двухмерного массива, показанного на рисунке, используем программу:

Протокол исполнения									
FOR I = 7 TO 9	I = 7			I = 8			I = 9		
FOR J = 7 TO 9	J = 7	J = 8	J = 9	J = 7	J = 8	J = 9	J = 7	J = 8	J = 9
INPUT A(I, J)	A(7, 7) = 0	A(7, 8) = 1	A(7, 9) = 0	A(8, 7) = 1	A(8, 8) = 1	A(8, 9) = 1	A(9, 7) = 0	A(9, 8) = 0	A(9, 9) = 1
NEXT J, I									

Идея решения задачи:

1. Распечатать таблицу `A` — одно из поколений бактерий.
2. Определить для каждого элемента таблицы сумму значений `S` окружающих его восьми элементов.
3. Если `S = 2` (два соседа) или `S = 3` (три соседа), то значение переписать во вспомогательную таблицу `B`: `B (I, J) = A (I, J)`. Если `S = 3` и клетка пустая — `A(I, J) = 0`, то в клетке зарождается новая жизнь `B(I, J) = 1`. Если `S >= 4`, то клетка гибнет, следовательно, `B(I, J) = 0`.
4. Скопировать таблицу `B` в таблицу `A` для получения следующего поколения.

```
DIM A(15, 15), B(15, 15)
CLS
FOR I = 7 TO 9                          Заполнение центральных клеток фишками.
FOR J = 7 TO 9
INPUT A(I, J)
NEXT J, I : K = 0 : C = 0
10 X$ = INKEY$
IF X$ = "" THEN 10
IF X$ = "K" THEN END
CLS : K = K + 1
C = C + 1 : COLOR K, 0                  Определение номера поколения C и номера цвета K.
IF K = 30 THEN K = 0
LOCATE 2, 20: PRINT C; "ПОКОЛЕНИЕ"
```

```
FOR I = 1 TO 15
FOR J = 1 TO 15
IF A(I, J) = 1 THEN LOCATE 3 + I, 20 + J: PRINT CHR$(8) _
ELSE LOCATE 3 + I, 20 + J: PRINT ""
NEXT J
NEXT I
FOR I = 2 TO 14
FOR J = 2 TO 14
s = a(i, j - 1) + a(i, j + 1) + a(i - 1, j - 1) + _
a (i - 1, j ) + a(i - 1, j + 1) + a(i + 1, j - 1) + _
a(i + 1, j ) +  a(i + 1, j + 1)
IF S = 2 AND A(I, J) = 1 THEN B(I, J) = A(I, J)
IF S = 3 AND A(I, J) = 1 THEN B(I, J) = A(I, J)
IF S = 3 AND A(I, J) = 0 THEN B(I, J) = 1
IF S >= 4 AND A(I, J) = 1 THEN B(I, J) = 0
NEXT J
NEXT I
FOR I = 1 TO 15
FOR J = 1 TO 15
A(I, J) = B(I, J) : B(I, J) = 0
NEXT I
GOTO 10
```

Определение суммы значений S.

Если два или три соседа — бактерия продолжает жить. Если три соседа — зарождается новая жизнь, а иначе бактерия гибнет.

Переписываются значения из массива B в A, теперь это новая конфигурация.

Введите данную программу, попробуйте отыскать интересные, неожиданные конфигурации, проследить за их превращениями. Без сомнения, вы получите много удовольствия, погрузившись в эту игру.

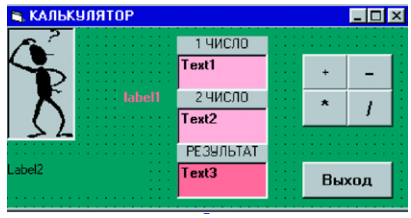
Модель калькулятора

Наиболее приемлемой средой программирования для решения данной задачи является Visual Basic.

Для модели калькулятора понадобятся следующие элементы:

1. Четыре кнопки (`CommandButton`) для выполнения арифметических операций: сложения, вычитания, умножения, деления.
2. Кнопка для окончания работы с приложением.
3. Три текстовых поля (`TextBox`) для ввода двух аргументов и показа результата.
4. Пять надписей (`Label`) для оформления.
5. Графическое поле (`PictureBox`) для оформления Справки. Чтобы заполнить графическое поле рисунком, необходимо выделить объект, найти свойство `Picture` и выбрать необходимую картинку из объектов `ClipArt`.

Составим программу, которая выполняет функции калькулятора:




```
Private Sub Command1_Click()
Label2.Caption = ""
Text3.Text = Val(Text1.Text) + Val(Text2.Text)
Label1.Caption = "+"
End Sub
```

Выполняется сложение.

```
Private Sub Command2_Click()
Label2.Caption = ""
Text3.Text = Val(Text1.Text) — Val(Text2.Text)
Label1.Caption = "—"
End Sub
```

Выполняется вычитание.

```
Private Sub Command3_Click()
Label2.Caption = ""
Text3.Text = Val(Text1.Text) * Val(Text2.Text)
Label1.Caption = "*"
End Sub
```

Выполняется умножение.

```
Private Sub Command4_Click()
Label2.Caption = ""
If Val(Text2.Text) <> 0 Then
Text3.Text = Val(Text1.Text)/Val(Text2.Text)
Label1.Caption = "/"
Else
MsgBox ("деление на ноль")
End If
End Sub
```

Выполняется деление, если делитель не равен нулю, иначе функция Visual Basic MsgBox выдает сообщение “деление на ноль”.

```
Private Sub Command5_Click()
End Sub
```

Окончание работы с приложением.

```
Private Sub Form_Load()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Label1.Caption = ""
Label2.Caption = ""
End Sub
```

Очистка текстовых полей и удаление надписей с элементов Label.

```
Private Sub Picture1_Click()
Label2.Caption = "СПРАВКА"
End Sub
```

Окончание следует

Номер лота	Название	Цена
10-01-0	**С.Л. Соловейчик. Пушкинские проповеди**	60 руб.
10-02-0	**С.Л. Соловейчик. Последняя книга**	75 руб.
10-05-0	**С.Л. Соловейчик. Педагогика для всех**	90 руб.
10-06-0	**М.А. Балабан. Школа-парк.** Как построить школу без классов и уроков	65 руб.
▼10-03-0	**Школа сотрудничества** (Сборник статей по педагогике сотрудничества)	46 руб.
27-01-9	**Я иду на урок физики. 7 класс (в 3-х книгах)**	138 руб.
27-02-0	**Я иду на урок астрономии. Звездное небо. 11 класс**	46 руб.
29-01-0	**Я иду на урок химии. 8–11 классы**	46 руб.
29-02-0	**Я иду на урок химии.** Летопись важнейших открытий. XVII–XIX вв.	46 руб.
21-01-0	**Я иду на урок математики. 5 класс**	46 руб.
21-01-7	**Я иду на урок математики. Тесты. 5 класс**	25 руб.
21-02-0	**Я иду на урок математики. 6 класс**	46 руб.
21-03-0	**Я иду на урок математики. Алгебра. 7 класс**	46 руб.
17-01-0	**Я иду на урок информатики. Задачи по программированию.** 7–11 классы	46 руб.
24-01-0	**Я иду на урок русского языка. Диктанты. 5–9 классы**	46 руб.
12-01-0	**Я иду на урок биологии. Зоология. Беспозвоночные**	46 руб.
12-02-0	**Я иду на урок биологии. Зоология. Рыбы и земноводные**	46 руб.
12-03-0	**Я иду на урок биологии. Зоология. Пресмыкающиеся**	46 руб.
12-04-0	**Я иду на урок биологии. Человек и его здоровье**	46 руб.
19-02-0	**Я иду на урок истории.** Древнейшая и древняя история	46 руб.
19-04-0	**Я иду на урок истории.** Материалы по истории средних веков	46 руб.
▼19-03-0	**Л.А. Кацва. Россия в 1917–1918 годах. 10–11 классы**	46 руб.
19-05-1	**Занимательная история России.** С древнейших времен до середины XVI века	46 руб.
19-05-2	**Занимательная история России.** Середина XVI века — конец XVII века	46 руб.
14-01-0	**Я иду на урок географии.** Физическая география материков и океанов	46 руб.
14-02-0	**Я иду на урок географии.** История географических открытий	46 руб.
14-03-0	**Я иду на урок географии. Физическая география России**	46 руб.
20-02-0	**Я иду на урок литературы. 5 класс**	46 руб.
20-03-0	**Я иду на урок литературы. 9 класс**	46 руб.
20-01-0	**Я иду на урок литературы. 10 класс**	46 руб.
20-04-0	**Я иду на урок литературы.** Готовимся к экзаменационному сочинению. 11 класс	46 руб.
20-05-0	**Я иду на урок литературы.** Современная русская литература. 1970—1990-е годы	46 руб.

▼ Тираж ограничен.

Указанные цены действительны до 1 июня 2002 года

ЧТОБЫ ПРИОБРЕСТИ ЗАИНТЕРЕСОВАВШИЕ ВАС КНИГИ НАЛОЖЕННЫМ ПЛАТЕЖОМ:

1. Укажите в купоне ваши фамилию, имя, отчество, индекс и почтовый адрес.
2. Выберите нужные вам книги из списка (не более 5 названий в один купон) и укажите соответствующий номер лота и количество экземпляров.
3. Вырежьте и вышлите купон в конверте с пометкой «Книга — почтой» по адресу: 150000, г. Ярославль, ул. П. Морозова, д. 5, а/я «Первое сентября».

Телефоны для справок: (0852) 45-07-77, (095) 249-28-77.

КУПОН *Книга – почтой*

Издательский дом **Первое сентября**

ЗАПОЛНЯЕТСЯ ПЕЧАТНЫМИ БУКВАМИ!

ФАМИЛИЯ

ИМЯ

ОТЧЕСТВО

ИНДЕКС АДРЕС

НОМЕР ЛОТА — — КОЛИЧЕСТВО ЭКЗЕМПЛЯРОВ

НОМЕР ЛОТА — — КОЛИЧЕСТВО ЭКЗЕМПЛЯРОВ

НОМЕР ЛОТА — — КОЛИЧЕСТВО ЭКЗЕМПЛЯРОВ

НОМЕР ЛОТА — — КОЛИЧЕСТВО ЭКЗЕМПЛЯРОВ

НОМЕР ЛОТА — — КОЛИЧЕСТВО ЭКЗЕМПЛЯРОВ

00-1-02 ОБРАЗЕЦ: 0 1 2 3 4 5 6 7 8 9

Указанная в купоне цена включает в себя стоимость доставки наземным транспортом от издательства до вашего почтового отделения (при авиаперевозках сумма наложенного платежа увеличивается на авиатариф). Рассылка производится только на территории РФ. Если полученная бандероль содержит неполный заказ, это означает, что заказ разбит на две или более бандероли.

World Wide Web – Всемирная информационная паутина

Революционизирующее влияние Интернета на мир компьютеров и коммуникаций не имеет исторических аналогов. Изобретение телеграфа, телефона, радио и компьютера подготовило почву для происходящей ныне беспрецедентной интеграции. Интернет одновременно является и средством общемирового вещания, и механизмом распространения информации, и средой для сотрудничества и общения людей и компьютеров, охватывающей весь земной шар.

Но Интернет так и остался бы компьютерной сетью для специалистов, если бы не два поистине революционных изобретения, сделанных в области программного обеспечения. Они, во-первых, сделали Интернет предметом первой необходимости для огромного числа людей, а во-вторых, обеспечили необыкновенную легкость работы с сетевым программным обеспечением. Первым изобретением стала электронная почта, вторым — Всемирная информационная паутина.

История WWW (**World Wide Web**) — Всемирной информационной паутины началась в марте 1989 года, когда **Тим Бернерс-Ли** из Европейской лаборатории физики элементарных частиц (*CERN*) предложил новый способ обмена результатами исследований и идеями между участниками коллектива исследователей-физиков, работавших в разных странах. Для передачи документов и установления связи предлагалось использовать просто систему **гипертекста** (тогда никто еще не задумывался о возможности передачи фотографических изображений, звука или видео; речь шла только о распространении текстовых документов, содержащих гиперссылки на фрагменты других таких же текстовых документов, но располагающихся на удаленных компьютерах, подключенных к глобальной сети Интернет).

Собственно, гипертекст не был изобретением Бернерса-Ли. Любой, кто пользовался справочной системой какой-либо программы, это подтвердит. Да и **HTML** вовсе не был первым языком описания страниц. Но в нем впервые были связаны воедино не просто отдельные главы одного документа, а документы, располагающиеся на самых различных серверах Интернета по всему земному шару.

До конца 1992 года программа оставалась лишь частью академических исследований — демонстрировалась на семинарах в CERN, да еще была представлена на конференции “Гипертекст'91”.

Переломный момент наступил в декабре 1992-го: **Марк Андрессен** вместе с **Эриком Бина** задумали написать свою собственную программу-клиент для просмотра гипертекстовых WWW-документов. И в марте следующего года (всего через три месяца!) на свет появилась **Mosaic** — мощная графическая интерфейсная программа для работы с Всемирной информационной паутиной и другими ресурсами Интернета. С этого момента пользователь уже мог не вдаваться в технические подробности реализации протоколов компьютерных сетей, а, используя очень простой и удобный инструмент, посвятить всего себя работе с интересующей его информацией, распределенной по огромному множеству компьютеров, расположенных в разных городах, странах и даже на разных континентах.

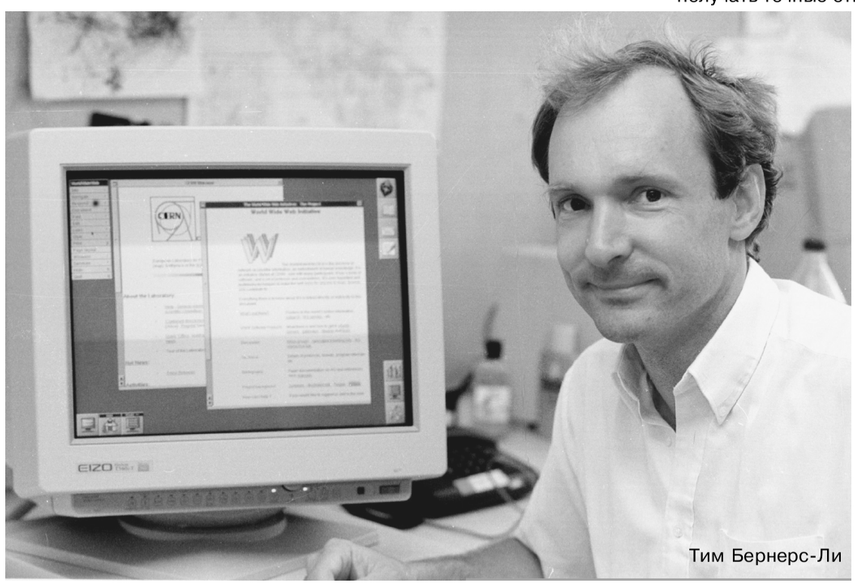
Mosaic в кратчайший срок стала самой быстро распространяющейся по миру программой. Таким образом, недавний выпускник колледжа **Марк Андрессен** добавил последний штрих, позволивший превратить всю Сеть, насчитывавшую в то время от одного до двух миллионов компьютеров, в единый гигантский суперисточник информации, работа с которым ничуть не труднее, чем для 10-летнего ребенка запуск любимой игры на домашнем персональном компьютере.

Чуть позже Андрессен вместе с **Джимом Кларком** основали в Калифорнии компанию **Netscape Communications**, на долгие годы ставшую законодателем мод в Интернете.

Представляется, что изобретение “Мозаики” было, быть может, даже более важным, чем “общесистемный” гипертекст. Мало того, величайшим бедствием для Интернета было то, что у его истоков оказались физики, не имевшие никакого понятия о том, что такое профессиональное качественное программное обеспечение, и ухитрившиеся в конце “золотого века теоретического программирования” навязать всему цивилизованному миру абсолютно любительскую поделку под названием HTML. В итоге мир до сих пор расхлебывает чудовищное нагромождение заплат и примочек, которое лишь по недоразумению называется “языком описания страниц”.

И поэтому вполне заслуженно звучат следующие слова, размещенные на сайте Netscape:

- Вначале была “Мозаика”.
- И поглядел Марк Андрессен на “Мозаику” и увидел, что это хорошо, и породил Netscape.
- И Netscape породил Livescript, язык сценариев, придуманный специально для Netscape. Sun Microsystems (разработчик Java) посмотрел на Livescript и увидел, что это хорошо.
- Sun & Netscape породили Netscape 2 со встроенным Javascript.



Тим Бернерс-Ли

- Netscape 2 породил Netscape 3, тоже с Javascript.
- И Microsoft посмотрел на Netscape 3 и увидел, что это хорошо, и породил Internet Explorer 3 (IE3) с JScript (неполноценным Javascript)...

Как видите, нигде не сказано, что HTML — это хорошо...

С каждым месяцем, днем и часом в гигантской паутине гипертекстовых связей появляются все новые и новые мегабайты информации, и теперь всякий исследователь, прокладывающий себе путь в необъятных просторах киберпространства, с первых же минут ощущает потребность в надежном путеводителе. **Дэвид Фило** и **Джерри Янг** из Станфордского университета в апреле 1994 года пришли к выводу, что у каждого из них накопилось такое большое количество ссылок на самые разные информационные источники, что пора каким-то образом их упорядочить. Так родилась идея создания специализированной базы данных, которая вскоре стала использоваться тысячами пользователей для эффективного поиска информации в Сети. Называется эта поисковая система **Yahoo!**, что расшифровывается как “*Yet Another Hierarchical Officious Oracle*” (“Еще Один Иерархический Официозный Оракул”). В начале следующего, 1995 года уже упоминавшийся выше Марк Андрессен помог Фило и Янгу переместить их детище на более мощные компьютеры Netscape, и вскоре система Yahoo! стала самым популярным и полным иерархическим предметно-ориентированным путеводителем по Всемирной паутине и Интернету в целом. В Yahoo! все рассортировано по темам и категориям, вся информация разложена по полочкам так, что даже самому ленивому пользователю не составит большого труда найти искомый ресурс в необозримом пространстве Сети.

Наряду с Yahoo! существуют и другие подобные системы, в том числе и русскоязычные.

Так, одним из самых известных является поисковый сервер **“Яндекс”**. Его история началась в 1990 году, когда в компании “Аркадия”, возглавляемой Аркадием Борковским и Аркадием Воложем, начались разработки поискового программного обеспечения. Сайт “Яndex” появился в Интернете в 1996 году, после того как руководством CompTek и разработчиками системы было принято решение о дальнейшем развитии поисковых технологий, ориентированных на широкую аудиторию пользователей Интернета.

Слово “Яndex” придумал за несколько лет до этого один из основных и старейших разработчиков поискового механизма. “Яndex” означает “Языковой index”, или, если по-английски, “Yandex” — “*Yet Another indexer*”. За 4 года существования Яndex возникли и другие толкования. Например, если в слове “Index” перевести с английского первую букву (“I” — “Я”), получится “Яndex”.

Официально поисковая машина Yandex.Ru была анонсирована 23 сентября 1997 года на выставке Softool. Основными отличительными чертами Yandex.Ru на тот момент были проверка уникальности документов (исключение копий в разных кодировках), а также ключевые свойства поискового ядра Яndex, а именно: учет морфологии русского языка (в том числе и поиск по точной словоформе), поиск с учетом расстояния (в том числе в пределах абзаца, точное словосочетание) и тщательно разработанный алгоритм оценки релевантности (соответствия ответа запросу), учитывающий не только количество слов запроса, найденных в тексте, но и “контрастность” слова (его относительную частоту для данного документа), расстояние между словами и положение слова в документе.

Через два месяца, в ноябре 1997 года, был реализован естественно-языковый запрос. К Yandex.Ru можно обращаться просто “по-русски”, задавать длинные запросы, например: “где купить компьютер”, “генетически модифицированные продукты” или “коды международной телефонной связи” — и получать точные ответы. Сейчас средняя длина запроса в Yandex.Ru — 2,7 слова. К примеру, в 1997 году она составляла 1,2 слова, тогда пользователи поисковых машин были приучены к телеграфному стилю.

Не менее известен и поисковый сервер **“Рамблер”**. Его история началась в 1991 году с появления команды единомышленников. В 1996 году был запущен в эксплуатацию поисковый механизм Rambler.ru, который первым из российских поисковых систем был включен в перечень “стандартных” поисковиков в браузере Internet Explorer. В феврале 1997 года начала работать рейтинговая система Rambler's Top100, которая с первых дней существования и по сей день считается лучшим классификатором российского Интернета и пользуется репутацией независимого арбитра российского Интернет-рынка. Спустя 3 года, 7 марта 2000 года, был зафиксирован миллиардный посетитель страниц, зарегистрированных в рейтинге.

Интересно, что и Рамблер, и Яндекс, будучи логически одним web-сервером, реализуют разделение задач по нескольким мощным серверам, что и обеспечивает очень высокую скорость обработки запросов.

Под конец определим основные понятия, сложившиеся в процессе развития системы гипертекстовых связей.

WWW — множество web-страниц, размещенных на узлах Интернета и связанных между собой гиперссылками.

Web-страница — структурная единица WWW, которая включает в себя:

1) собственно информацию (текстовую и графическую);

2) ссылки на другие страницы или ресурсы Интернета.

Система гипертекстовых web-страниц, контролируемых одним человеком или одной организацией, обычно называют **web-сайтом**.

Программа, пересылающая web-страницы на компьютер клиента с использованием протокола HTTP (*Hyper Text Transfer Protocol* — протокол передачи гипертекста), называется **web-сервером**. На одном компьютере могут работать несколько web-серверов, один web-сервер может использовать несколько компьютеров.

Навигация по миллионам web-серверов была бы совершенно немыслима без строгого порядка, главным элементом которого является DNS — доменная система имен.

Использованные источники информации

1. World Wide Web — Всемирная информационная паутина (**http://www.compulenta.ru**).
2. История Яндекса (**http://yandex.ru**).
3. История Рамблера (**http://rambler.ru**).

Примерные ответы на примерные билеты

Е.А. Еремин, А.П. Шестаков,
г. Пермь

Продолжение. Начало см. в № 9/2002

Билет № 3

1. Характеристики процессора и внутренней памяти компьютера (быстродействие, разрядность, объем памяти и др.).

2. Разработка алгоритма перемещений и действий для исполнителей типа "Робот" или "Черепашка".

* * *

1. Характеристики процессора и внутренней памяти компьютера (быстродействие, разрядность, объем памяти и др.).

Необычайно быстрое развитие вычислительной техники приводит к тому, что одновременно в употреблении находится большое количество компьютеров с достаточно разнообразными характеристиками. Поэтому очень полезно знать, каковы основные характеристики узлов компьютера, на что они влияют и как их подбирать. Здесь будут рассмотрены параметры наиболее важных устройств компьютера, таких, как *процессор* и *внутренняя память*.

Начнем с **процессора**. Очевидно, что пользователя в первую очередь интересует его производительность, т.е. скорость выполнения предложенной процессору задачи. Традиционно **быстродействие** процессора измерялось путем определения **количества операций в единицу времени**, как правило, в секунду. До тех пор, пока машины выполняли только вычисления, такой показатель был достаточно удобен. Однако по мере развития вычислительной техники количество видов обрабатываемой информации возрастало, и обсуждаемый показатель перестал быть универсальным. В самом деле, в простейшем случае даже количество арифметических действий над целыми и над вещественными числами может для одного и того же компьютера отличаться на порядок! Что говорить о скорости обработки графической или видеоинформации, которые к тому же зависят не только от самого процессора, но и от устройства видеоблоков компьютера... Кроме того, современные процессоры, например Pentium, имеют очень сложное внутреннее устройство и могут выполнять машинные команды параллельно. Иными словами, процессор может *одновременно* выполнять несколько разных инструкций, а значит, время завершения команды уже зависит не только от нее самой, но и от "соседних" операций! Таким образом, количество выполняемых за секунду операций перестает быть постоянным и выбирать его в качестве характеристики процессора не очень удобно.

Именно поэтому сейчас получила широкое распространение другая характеристика скорости работы процессора — его **тактовая частота**. Рассмотрим данную величину подробнее. Любая операция процессора (машинная команда) состоит из отдельных элементарных действий — **тактов**. Для организации последовательного выполнения требуемых тактов друг за другом в компьютере имеется специальный генератор импульсов, каждый из которых инициирует очередной такт машинной команды (какой именно, определяется устройством процессора и логикой выполняемой операции). Очевидно, что чем чаще следуют импульсы от генератора, тем быстрее будет выполнена операция, состоящая из фиксированного числа тактов. Из сказанного следует, что тактовая частота определяется количеством импульсов в секунду и измеряется в **мегагерцах** — т.е. миллионах импульсов за 1 секунду. Разумеется, тактовая частота не может быть произвольно высокой, поскольку в какой-то момент процессор может просто "не успеть" выполнить очередной такт до прихода следующего импульса. Однако инженеры делают все возможное для повышения значения этой характеристики процессора, и на данный момент тактовая частота самых современных процессоров уже превышает 1000 МГц, т.е. 1 ГГц (1 гигагерц).

Следует четко представлять, что сравнение тактовых частот позволяет надежно определить, какой из двух процессоров более быстродействующий только в том случае, если оба процессора устроены примерно одинаково. Если же попытаться сравнить процессоры, произведенные разными изготовителями и работающие по разным принципам, можно получить абсолютно неправильные выводы. В самом деле, если в одном из процессоров команда выполняется за 2 такта, а в другом — за 3, то при совершенно одинаковой частоте первый будет работать в полтора раза быстрее! Кроме того, не нужно забывать, что производительность современной компьютерной системы определяется не только быстродействием отдельно взятого процессора, но и скоростями работы *остальных* узлов компьютера и даже способами организации всей системы в целом: очевидно, что чрезмерно быстрый процессор будет вынужден постоянно простаивать, ожидая, например, медленно работающую память; или другой пример — очень часто простое увеличение объема ОЗУ дает гораздо больший эффект, чем замена процессора на более быстрый.

Косвенно скорость обработки информации зависит и еще от одного параметра процессора — его **разрядности**. Под разрядностью обычно понимают число одновременно обрабатываемых процессором битов. Формально эта величина есть количество двоичных разрядов в регистрах процессора, и для современных моделей она равна 32. Тем не менее все не так просто. Дело в том, что, помимо описанной "внутренней" разрядности процессора, существуют еще разрядность шины данных, которой он управляет, и разрядность шины адреса (о шине более подробно рассказано в предыдущем билете).

Эти характеристики далеко не всегда совпадают (данные для таблицы взяты из книги М.Гука "Процессоры Intel: от 8086 до Pentium II". СПб.: Питер, 1997):

Процессор	Разрядность			Объем памяти
	регистров	шины данных	шины адреса	
Intel 8086	16	16	20	до 1 Мб
Intel 80286	16	16	24	до 16 Мб
Intel 80386	32	16	24	до 16 Мб
Intel 80486	32	32	32	до 4 Гб
Pentium	32	64	32	до 4 Гб
Pentium II	32	64	36	до 64 Гб

Мы не будем обсуждать технические причины, по которым эти три разрядности могут различаться между собой, ибо причины эти сейчас представляют в основном исторический интерес. Отметим только, что разрядность регистров и разрядность шины данных влияют на длину обрабатываемых данных, а вот разрядность шины адреса R определяет максимальный объем памяти, который способен поддерживать процессор. Эту характеристику часто называют величиной **адресного пространства**, и она может быть вычислена по простой формуле 2^R. Действительно, R двоичных разрядов позволяют получить именно такое количество неповторяющихся чисел, т.е. в данном случае адресов памяти.

Перейдем теперь к описанию основных характеристик **памяти** компьютера.

Хотя память компьютера состоит из отдельных битов, непосредственно "общаться" с каждым из них невозможно: биты группируются в более крупные блоки информации, и именно они получают адреса, по которым происходит обращение к памяти. По сложившейся исторической традиции ***минимальная*** порция информации, которую современный компьютер способен записать в память, составляет 8 бит, или 1 байт. Отсюда становится очевидным, что общий **объем памяти** должен измеряться в байтах или в производных от него единицах. Размер памяти персональных компьютеров стремительно возрастает. Первые модели имели 16-разрядное адресное пространство, следовательно, объем памяти 2^{16} = 64 Кб. Затем, когда памяти под разрабатываемые программные системы перестало хватать, инженеры введением некоторых весьма специфических способов формирования адреса увеличили ее размер на порядок — в MS-DOS стандартная память была принята равной 640 Кб. Сейчас вы вряд ли сможете приобрести новый компьютер с ОЗУ менее 32—64 Мб, т.е. еще на два порядка больше (надеемся, читатели не забыли, что 1 Мб = 1024 Кб!).

Еще одной важной характеристикой памяти является **время доступа**, или **быстродействие памяти**. Этот параметр определяется временем выполнения операций записи или считывания данных; он зависит от принципа действия и технологии изготовления запоминающих элементов.

Оставляя в стороне целый ряд других технологических характеристик современных запоминающих устройств, нельзя тем не менее пройти мимо **статического** и **динамического** устройств микросхем памяти. Статическая ячейка памяти — это специальная полупроводниковая схема (инженеры называют ее триггером), обладающая двумя устойчивыми состояниями. Одно из них принимается за логический ноль, а другое — за единицу. Состояния эти действительно настолько устойчивы, что при отсутствии внешних воздействий (и, конечно, подключенном напряжении питания!) могут сохраняться сколь угодно долго. Динамические ячейки памяти, напротив, не обладают этим свойством. Такие ячейки фактически представляют собой конденсатор, образованный элементами полупроводниковых микросхем. С некоторым упрощением можно сказать, что логической единице соответствует заряженный конденсатор, а нулю — незаряженный. Существенным свойством динамической ячейки памяти является наличие постепенного самопроизвольного разряда конденсатора через внешние схемы, что ведет к потере информации. Чтобы этого не происходило, конденсаторы динамической памяти необходимо периодически подзаряжать (такой процесс принято называть **регенерацией ОЗУ**). Оба вида запоминающих микросхем успешно конкурируют между собой, поскольку ни одна из них не является идеальной. С одной стороны, статическая память значительно проще в эксплуатации, так как не требует регенерации и приближается по быстродействию к процессорным микросхемам. С другой стороны, она имеет меньший информационный объем и большую стоимость (в самом деле, изготовление конденсатора значительно проще изготовления триггерной схемы, кроме того, конденсатор требует на кремниевой пластине гораздо меньше места), сильнее нагревается при работе. На практике в данный момент *выбор микросхем для построения ОЗУ всегда решается в пользу динамической памяти.* И все же быстродействующая статическая память в современном компьютере тоже обязательно есть: она называется **кэш-памятью**.

Этот вид памяти заслуживает отдельного рассмотрения. Он появился относительно недавно, но, начиная с 486-го процессора, без кэш-памяти не обходится ни одна модель. Название "кэш" происходит от английского слова *cache*, которое обозначает тайник или замаскированный склад (в частности, этим словом называют провиант, оставленный экспедицией для обратного пути, или запас продуктов, например, зерна или меда, который животные создают на зиму). "Секретность" кэша заключается в том, что он невидим для пользователя и данные, хранящиеся там, недоступны для прикладного программного обеспечения. Процессор использует кэш, помещая туда извлеченные им из ОЗУ данные и команды программы и запоминая при этом в специальном каталоге адреса, откуда информация была извлечена. Если эти данные потребуются повторно, то уже не надо будет терять времени на обращение к ОЗУ — их можно получить из кэш-памяти значительно быстрее. Поскольку объем кэша существенно меньше объема оперативной памяти, его контроллер (управляющая схема) тщательно следит за тем, какие данные следует сохранять в кэше, а какие заме-

нять: удаляется та информация, которая используется реже или совсем не используется. Следует заметить, что кэш-память является очень эффективным средством повышения производительности компьютера, в чем легко убедиться на практике, если в вашем компьютере предусмотрена возможность отключения кэша.

В современных компьютерах кэш обычно строится по двухуровневой схеме. При этом первичный кэш встроен непосредственно внутрь процессора, а вторичный обычно устанавливается на системной плате. Как и для ОЗУ, увеличение **объема** кэша повышает эффективность работы компьютерной системы.

Литература

1. Информатика в понятиях и терминах: Книга для учащихся старших классов средней школы / Г.А. Бордовский, В.А. Извозчиков, Ю.В. Исаев, В.В. Морозов; Под ред. В.А. Извозчикова. М.: Просвещение, 1991, 208 с.

2. *Радченко Н.П., Козлов О.А.* Школьная информатика: экзаменационные вопросы и ответы. М.: Финансы и статистика, 1998, 160 с.

3. *Семакин И., Залогова Л., Русаков С., Шестакова Л.* Информатика. Учебник по базовому курсу (7—9-е классы). М.: Лаборатория базовых знаний, 1998, 464 с.

4. *Кушниренко А.Г. и др.* Основы информатики и вычислительной техники: Пробный учебник для средних учебных заведений / А.Г. Кушниренко, Г.В. Лебедев, Р.А. Сворень. М.: Просвещение, 1990, 224 с.

5. *Гук М.* Аппаратные средства IBM PC. Энциклопедия. СПб.: Изд-во "Питер", 2000, 816 с.

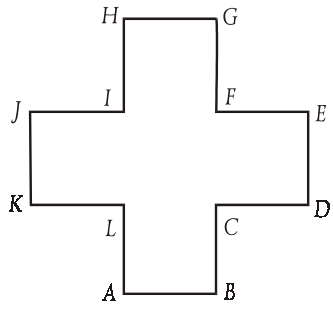
* * *

2. Разработка алгоритма перемещений и действий для исполнителей типа "Робот" или "Черепашка".

Просмотрев упражнения к нескольким школьным учебникам, мы остановили свой выбор на задаче построения изображения, которая может решаться для самых разнообразных исполнителей. Сразу подчеркнем, что, помимо приведенных ниже решений, содержащих процедуры и циклы, ученики могут реализовать и более простые линейные решения (просто написать длинный список необходимых команд, не обращая внимания на то, что они порой повторяются). По нашему мнению, это хорошо, ибо дает возможность слабым ученикам хоть как-то справиться с заданием; разумеется, оценка в последнем случае должна быть ниже.

Условие задачи

Составить алгоритм, в результате выполнения которого на экране будет вычерчено следующее изображение:



Рисование начать с точки *A* и далее перемещаться согласно указанным на рисунке буквам латинского алфавита. Считать, что все стороны изображаемой фигуры равны.

Примечание. Буквы нанесены на рисунок исключительно ради удобства обсуждения в тексте статьи; их рисование на дисплее ***в задачу не входит***!

Решения для различных исполнителей

Конкретный вид решения, разумеется, сильно зависит от системы команд исполнителя. Тем не менее для некоторых из них алгоритмы выглядят очень похоже. Ниже показаны решения задачи для исполнителей "**Черепашка**" (язык Лого) и "**Кенгуренок**". В обоих из них после некоторых подготовительных действий четыре раза выполняется процедура `p1`, которая рисует по три стороны фигуры (первый раз это *AB*, *BC* и *CD*, затем по той же программе изображаются стороны *DE*, *EF* и *FG* и т.д.).

Программа для LogoWriter	Программа для исполнителя "Кенгуренок"

`To main :X`
`Home`
`Clean`
`Right 90`
`Repeat 4 [p1 :X]`
`End`
 	
`To p1 :X`
`Forward :X`
`Left 90`
`Forward :X`
`Right 90`
`Forward :X`
`Left 90`
`End`
	прыжок
	прыжок
	прыжок
	прыжок
	сделай p1
	сделай p1
	сделай p1
	сделай p1
	
	процедура p1
	сделай line
	поворот
	сделай line
	поворот
	поворот
	поворот
	сделай line
	поворот
	конец процедуры
	
	процедура line
	шаг
	шаг
	шаг
	шаг
	конец процедуры

Поскольку исполнитель "Кенгуренок" обладает меньшими возможностями (например, поворачивается только налево и не умеет подсчитывать количество повторений), программа для него заметно длиннее.

А вот для исполнителя "**Чертежник**", описанного в учебнике А.Г. Кушниренко с соавторами, алгоритм будет несколько другим. Причина состоит в том, что в отличие от двух предыдущих исполнителей "Чертежник" не имеет понятий "вперед" и "назад", а пользуется системой координат, связанной с листом бумаги. С другой стороны, благодаря такому принципу работы он не нуждается в поворотах. Поэтому для "Чертежника" фигура разделяется не на четыре, а на две повторяющиеся части: их границей служит точка *G* на рисунке. Команды рисова-

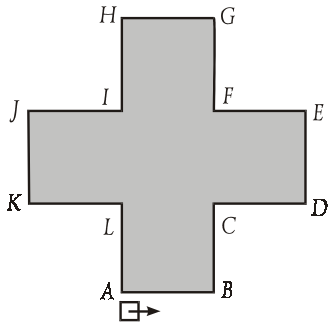
ния обеих частей рисунка совпадают с точностью до знака аргумента X, именно поэтому повторный вызов вспомогательного алгоритма p1 содержит знак "минус" перед аргументом. Обратите также внимание на то, что алгоритмы рисования линий *AB*—*BC* и *CD*—*DE* для данного исполнителя одинаковы, что позволяет во вспомогательном алгоритме p1 использовать цикл повторения необходимых действий два раза (разумеется, если ученик вместо такого цикла дважды напишет повторяющуюся пару команд, это не будет ошибкой!).

Исполнитель "Чертежник"

```
алг плюс (арг вещ X)
нач
| поднять перо
| сместиться на вектор(X, 0)
| опустить перо
| p1(X)
| p1(–X)
кон

алг p1(арг вещ X)
нач
| нц 2 раза
| | сместиться на вектор(X, 0)
| | сместиться на вектор(0, X)
| кц
| сместиться на вектор(–X, 0)
| сместиться на вектор(0, X)
кон
```

Наконец, рассмотрим еще одного исполнителя, который также описывается в этом же учебнике, — это "**Робот**". "Робот" не умеет рисовать, поэтому для него придется несколько переформулировать задачу. Двигаясь вдоль стены, он должен будет обойти вокруг всей фигуры из начального положения, показанного на рисунке около буквы *A*.



Кстати говоря, данное небольшое изменение задачи приводит к существенному изменению ее решения. Дело в том, что размеры стены, вдоль которой движется "Робот" (если хотите, для наглядности ее можно называть забором), теперь заданы, и ему не надо заботиться о длине пути — достаточно контролировать свое положение относительно стены. Именно поэтому в решении отсутствуют какие-либо геометрические параметры.

И еще об одной особенности решения для данного исполнителя. Из-за довольно "жесткой" системы команд — только на одну клетку строго в указанном направлении — решение получается очень громоздким. Может быть, стоит даже упростить задачу, убрав, например, выступ стены *FGHI*. Нам удалось написать алгоритм, используя 8 однотипных вспомогательных алгоритмов, т.е. по два на каждое направление (напомню, что всего у фигуры 12 сторон). Три первых алгоритма (для сторон *AB*, *BC* и *CD*) приведены в правой колонке, остальные легко могут быть восстановлены читателями по аналогии. Полное решение в виде текстового файла, как обычно, доступно на страницах сайта **http://info-bilet.narod.ru.**

Исполнитель "Робот"

```
алг обход
нач
| право1
| верх2
| право1
| верх1
| лево2
| верх1
| лево1
| низ2
| лево1
| низ1
| право2
| низ1
кон
алг право1
нач
| нц пока сверху стена
| | вправо
| кц
кон
алг верх2
нач
| нц пока сверху свободно
| | вверх
| кц
кон
алг верх1
нач
| нц пока слева стена
| | вверх
| кц
кон
...
```

Так выглядят решения предложенной задачи для четырех различных исполнителей. Разумеется, учитель может использовать и любого другого исполнителя.

Билет № 4

1. Внешняя память компьютера. Носители информации (гибкие диски, жесткие диски, диски CD-ROM, магнитооптические диски и пр.) и их основные характеристики.

2. Определение результата выполнения алгоритма по его блок-схеме, записи на естественном языке или записи на языке программирования.

* * *

1. Внешняя память компьютера. Носители информации (гибкие диски, жесткие диски, диски CD-ROM, магнитооптические диски и пр.) и их основные характеристики.

Внешняя (долговременная) *память* — это место длительного хранения данных (программ, результатов расчетов, текстов и т.д.), не используемых в данный момент в оперативной памяти компьютера. Внешняя память в отличие от оперативной является энергонезависимой. Носители внешней памяти, кроме того, обеспечивают транспортировку данных в тех случаях, когда компьютеры не объединены в сети (локальные или глобальные).

Для работы с внешней памятью необходимо наличие *накопителя* (устройства, обеспечивающего запись и (или) считывание информации) и устройства хранения — *носителя*.

Основные виды накопителей:
- накопители на гибких магнитных дисках (НГМД);
- накопители на жестких магнитных дисках (НЖМД);
- накопители на магнитной ленте (НМЛ);
- накопители CD-ROM, CD-RW, DVD.

Им соответствуют основные виды носителей:
- гибкие магнитные диски (*Floppy Disk*) (диаметром 3,5" и емкостью 1,44 Мб; диаметром 5,25" и емкостью 1,2 Мб (в настоящее время устарели и практически не используются; выпуск накопителей, предназначенных для дисков диаметром 5,25", тоже прекращен)), диски для сменных носителей;
- жесткие магнитные диски (*Hard Disk*);
- кассеты для стримеров и других НМЛ;
- диски CD-ROM, CD-R, CD-RW, DVD.

Запоминающие устройства принято делить на виды и категории в связи с их принципами функционирования, эксплуатационно-техническими, физическими, программными и др. характеристиками. Так, например, по принципам функционирования различают следующие виды устройств: электронные, магнитные, оптические и смешанные — магнитооптические. Каждый тип устройств организован на основе соответствующей технологии хранения/воспроизведения/записи цифровой информации. Поэтому в связи с видом и техническим исполнением носителя информации различают электронные, дисковые и ленточные устройства.

Основные характеристики накопителей и носителей:
- информационная емкость;
- скорость обмена информацией;
- надежность хранения информации;
- стоимость.

Остановимся подробнее на рассмотрении вышеперечисленных накопителей и носителей.

Принцип работы *магнитных запоминающих устройств* основан на способах хранения информации с использованием магнитных свойств материалов. Как правило, магнитные запоминающие устройства состоят из собственно *устройств чтения/записи информации* и *магнитного носителя*, на который непосредственно осуществляется запись и с которого считывается информация. Магнитные запоминающие устройства принято делить на виды в связи с исполнением, физико-техническими характеристиками носителя информации и т.д. Наиболее часто различают дисковые и ленточные устройства. Общая технология магнитных запоминающих устройств состоит в намагничивании переменным магнитным полем участков носителя и считывания информации, закодированной как области переменной намагниченности. Дисковые носители, как правило, намагничиваются вдоль концентрических полей — дорожек, расположенных по всей плоскости дискоидального вращающегося носителя. Запись производится в цифровом коде. Намагничивание достигается за счет создания переменного магнитного поля при помощи головок чтения/записи. Головки представляют собой два или более магнитных управляемых контура с сердечниками, на обмотки которых подается переменное напряжение. Изменение величины напряжения вызывает изменение направления линий магнитной индукции магнитного поля и при намагничивании носителя означает смену значения бита информации с 1 на 0 или с 0 на 1.

Дисковые устройства делят на гибкие (*Floppy Disk*) и жесткие (*Hard Disk*) накопители и носители. Основным свойством дисковых магнитных устройств является запись информации на носитель на концентрические замкнутые дорожки с использованием физического и логического цифрового кодирования информации. Плоский дисковый носитель вращается в процессе чтения/записи, чем и обеспечивается обслуживание всей концентрической дорожки, чтение и запись осуществляются при помощи магнитных головок чтения/записи, которые позиционируют по радиусу носителя с одной дорожки на другую.

Для операционной системы данные на дисках организованы в *дорожки* и *секторы*. *Дорожки* (40 или 80) представляют собой узкие концентрические кольца на диске. Каждая дорожка разделена на части, называемые *секторами*. При чтении или записи устройство всегда считывает или записывает целое число секторов независимо от объема запрашиваемой информации. Размер сектора на дискете равен 512 байтам. *Цилиндр* — это общее количество дорожек, с которых можно считать информацию, не перемещая головок. Поскольку гибкий диск имеет только две стороны, а дисковод для гибких дисков — только две головки, в гибком диске на один цилиндр приходится две дорожки. В жестком диске может быть много дисковых пластин, каждая из которых имеет две (или больше) головки, поэтому одному цилиндру соответствует множество дорожек. *Кластер* (или *ячейка размещения данных*) — наименьшая область диска, которую операционная система использует при записи файла. Обычно кластер — один или несколько секторов.

Перед использованием дискета должна быть форматирована, т.е. должна быть создана ее логическая и физическая структура.

Дискеты требуют аккуратного обращения. *Они могут быть повреждены, если*

- дотрагиваться до записывающей поверхности;
- писать на этикетке дискеты карандашом или шариковой ручкой;
- сгибать дискету;
- перегревать дискету (оставлять на солнце или около батареи отопления);
- подвергать дискету воздействию магнитных полей.

Накопители на жестких дисках объединяют в одном корпусе *носитель (носители)* и *устройство чтения/записи*, а также нередко и *интерфейсную часть*, называемую *контроллером жесткого диска*. Типичной конструкцией жесткого диска является исполнение в виде одного устройства — камеры, внутри которой находится один или более дисковых носителей, помещенных на одну ось, и блок головок чтения/записи с их общим приводящим механизмом. Обычно рядом с камерой носителей и головок располагаются схемы управления головками, дисками и часто интерфейсная часть и (или) контроллер. На интерфейсной карте устройства располагается собственно интерфейс дискового устройства, а контроллер с его интерфейсом располагается на самом устройстве. С интерфейсным адаптером схемы накопителя соединяются при помощи комплекта шлейфов.

Принцип функционирования жестких дисков аналогичен этому принципу для ГМД.

Основные физические и логические параметры ЖД.

- *Диаметр дисков*. Наиболее распространены накопители с диаметром дисков 2,2, 2,3, 3,14 и 5,25 дюйма.
- *Число поверхностей* — определяет количество физических дисков, нанизанных на ось.
- *Число цилиндров* — определяет, сколько дорожек будет располагаться на одной поверхности.
- *Число секторов* — общее число секторов на всех дорожках всех поверхностей накопителя.
- *Число секторов на дорожке* — общее число секторов на одной дорожке. Для современных накопителей показатель условный, так как они имеют неравное число секторов на внешних и внутренних дорожках, скрытое от системы и пользователя интерфейсом устройства.
- *Время перехода от одной дорожки к другой* обычно составляет от 3,5 до 5 миллисекунд, а у самых быстрых моделей может быть от 0,6 до 1 миллисекунды. Этот показатель является одним из определяющих быстродействие накопителя, так как именно переход с дорожки на дорожку является самым длительным процессом в серии процессов произвольного чтения/записи на дисковом устройстве.
- *Время установки или время поиска* — время, затрачиваемое устройством на перемещение головок чтения/записи к нужному цилиндру из произвольного положения.
- *Скорость передачи данных*, называемая также *пропускной способностью*, определяет скорость, с которой данные считываются или записываются на диск после того, как головки займут необходимое положение. Измеряется в мегабайтах в секунду (Мб/с) или мегабитах в секунду (Мбит/с) и является характеристикой контроллера и интерфейса.

В настоящее время используются в основном жесткие диски емкостью от 10 Гб до 80 Гб. Наиболее популярными являются диски емкостью 20, 30, 40 Гб.

Кроме НГМД и НЖМД, довольно часто используют сменные носители. Довольно популярным накопителем является Zip. Эти накопители могут хранить 100 или 250 Мб данных на картриджах, напоминающих дискету формата 3,5", обеспечивают время доступа, равное 29 мс, и скорость передачи данных до 1 Мб/с. Если устройство подключается к системе через параллельный порт, то скорость передачи данных ограничена скоростью параллельного порта.

К типу накопителей на сменных жестких дисках относится накопитель Jaz. Емкость используемого картриджа — 1 или 2 Гб. Недостаток — высокая стоимость картриджа. Основное применение — резервное копирование данных.

В накопителях на магнитных лентах (чаще всего в качестве таких устройств выступают *стримеры*) запись производится на мини-кассеты. Емкость таких кассет — от 40 Мб до 13 Гб, скорость передачи данных — от 2 до 9 Мб/мин., длина ленты — от 63,5 до 230 м, количество дорожек — от 20 до 144.

CD-ROM — это оптический носитель информации, предназначенный только для чтения, на котором может храниться до 650 Мб данных. Доступ к данным на CD-ROM осуществляется быстрее, чем к данным на дискетах, но медленнее, чем на жестких дисках.

Компакт-диск диаметром 120 мм (около 4,75") изготовлен из полимера и покрыт металлической пленкой. Информация считывается именно с этой металлической пленки, которая покрывается полимером, защищающим данные от повреждения. CD-ROM является односторонним носителем информации.

Считывание информации с диска происходит за счет регистрации изменений интенсивности отраженного от алюминиевого слоя излучения маломощного лазера. Приемник или фотодатчик определяет, отразился ли луч от гладкой поверхности, был рассеян или поглощен. Рассеивание или поглощение луча происходит в местах, где в процессе записи были нанесены углубления. Фотодатчик воспринимает рассеянный луч, и эта информация в виде электрических сигналов поступает на микропроцессор, который преобразует эти сигналы в двоичные данные или звук.

Скорость считывания информации с CD-ROM сравнивают со скоростью считывания информации с музыкального диска (150 Кб/с), которую принимают за единицу. На сегодняшний день наиболее распространенными являются 52-скоростные накопители CD-ROM (скорость считывания — 7500 Кб/с).

Накопители CD-R (CD-Recordable) позволяют записывать собственные компакт-диски.

Более популярными являются накопители CD-RW, которые позволяют записывать и перезаписывать диски CD-RW, записывать диски CD-R, читать диски CD-ROM, т.е. являются в определенном смысле универсальными.

Аббревиатура DVD расшифровывается как ***D**igital **V**ersatile **D**isk*, т.е. *универсальный цифровой диск*. Имея те же габариты, что обычный компакт-диск, и весьма похожий принцип работы, он вмещает чрезвычайно много информации — от 4,7 до 17 Гб. Возможно, именно из-за большой емкости он и называется универсальным. Правда, на сегодня реально применяется DVD-диск лишь в двух областях: для хранения видеофильмов (DVD-Video или просто DVD) и сверхбольших баз данных (DVD-ROM, DVD-R).

Разброс емкостей возникает так: в отличие от CD-ROM диски DVD записываются с обеих сторон. Более того, с каждой стороны могут быть нанесены один или два слоя

информации. Таким образом, односторонние однослойные диски имеют объем 4,7 Гб (их часто называют DVD-5, т.е. диски емкостью около 5 Гб), двусторонние однослойные — 9,4 Гб (DVD-10), односторонние двухслойные — 8,5 Гб (DVD-9), а двусторонние двухслойные — 17 Гб (DVD-18). В зависимости от объема требующих хранения данных и выбирается тип DVD-диска. Если речь идет о фильмах, то на двусторонних дисках часто хранят две версии одной картины — одна широкоэкранная, вторая в классическом телевизионном формате.

Таким образом, здесь приведен обзор основных устройств внешней памяти с указанием их характеристик.

Основная литература

1. *Гейн А.Г., Сенокосов А.И., Шолохович В.Ф.* Информатика. 7—9-е классы: Учебник для общеобразовательных учебных заведений. М.: Дрофа, 1998.

2. *Каймин В.А., Щеголев А.Г., Ерохина Е.А., Федюшин Д.П.* Основы информатики и вычислительной техники: Пробный учебник для 10—11-х классов средней школы. М.: Просвещение, 1989.

3. *Кушниренко А.Г., Лебедев Г.В., Сворень Р.А.* Основы информатики и вычислительной техники: Учебник для средних учебных заведений. М.: Просвещение, 1993.

4. *Семакин И., Залогова Л., Русаков С., Шестакова Л.* Информатика: учебник по базовому курсу. М.: Лаборатория базовых знаний, 1998.

5. *Угринович Н.* Информатика и информационные технологии. Учебное пособие для общеобразовательных учреждений. М.: БИНОМ, 2001, 464 с. (§ 2.14. Хранение информации, с. 91—98).

Дополнительная литература

1. **http://citforum.ru/pp/pc03.shtml** — Накопители на магнитных дисках.

2. *Мюллер Скотт.* Модернизация и ремонт ПК, 11-е издание: Пер. с англ.: Учебное пособие. М.: Издательский дом "Вильямс", 2000, 1136 с. (главы 9—14).

* * *

2. Определение результата выполнения алгоритма по его блок-схеме, записи на естественном языке или записи на языке программирования.

Решение подобного рода задач основано на пошаговом исполнении алгоритма, в итоге делается вывод о том, какую задачу выполняет этот алгоритм и что является конечным результатом.

Предлагается задача: в приведенном алгоритме при $k = 4$ каким будет выведенное значение P?

Это задание можно сформулировать как тестовое, где нужно выбрать правильный ответ из нескольких предложенных и обосновать его.

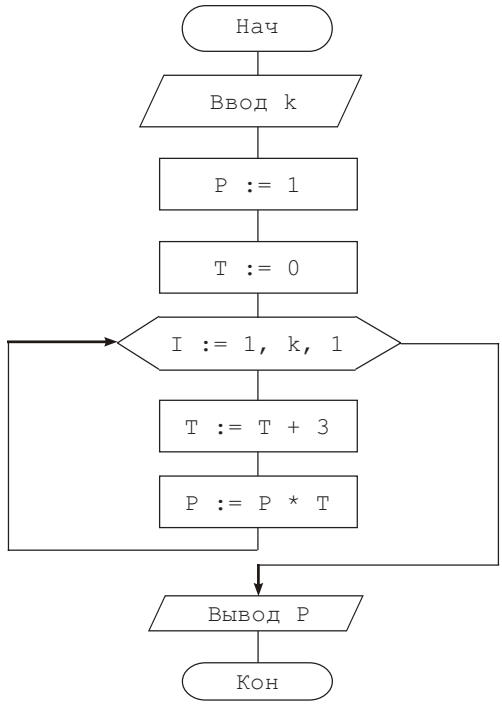
В нашем случае варианты ответов (правильный выделен жирным шрифтом).

1) 1; 3) 12; **5) 1944**.
2) 4; 4) 81;

Кроме того, можно предложить просто определить ответ и сформулировать условие задачи, решение которой приведено.

В нашем примере задача формулируется так: **найти произведение первых k натуральных чисел, кратных 3; $P = 1944$.**

Этот же алгоритм, в зависимости от того, как преподавался курс алгоритмизации и программирования, можно предложить для исследования, записав его на одном из алгоритмических языков.



Язык Паскаль

```
program Ex_1;
var k, P, T, I : word;
begin
 write('Введите натуральное число');
 readln(k);
 P := 1; T := 0;
 for I := 1 to k do
 begin T := T + 3; P := P * T end;
 writeln('Результат:', P)
end.
```

Язык Бейсик

```
input "Введите натуральное число: ", k
p = 1: t = 0
for i = 1 to k
    t = t + 3: p = p * t
next i
print "Результат: ", p
end
```

Язык Си

```
#include <stdio.h>
void main()
{int k, p, i, t;
 printf("Введите натуральное число: ");
 scanf("%d", &k);
 p = 1; t = 0;
 for (i = 1; i <= k; i++)
 {t += 3; p *= t;}
 printf("Результат: %d", p);
}
```

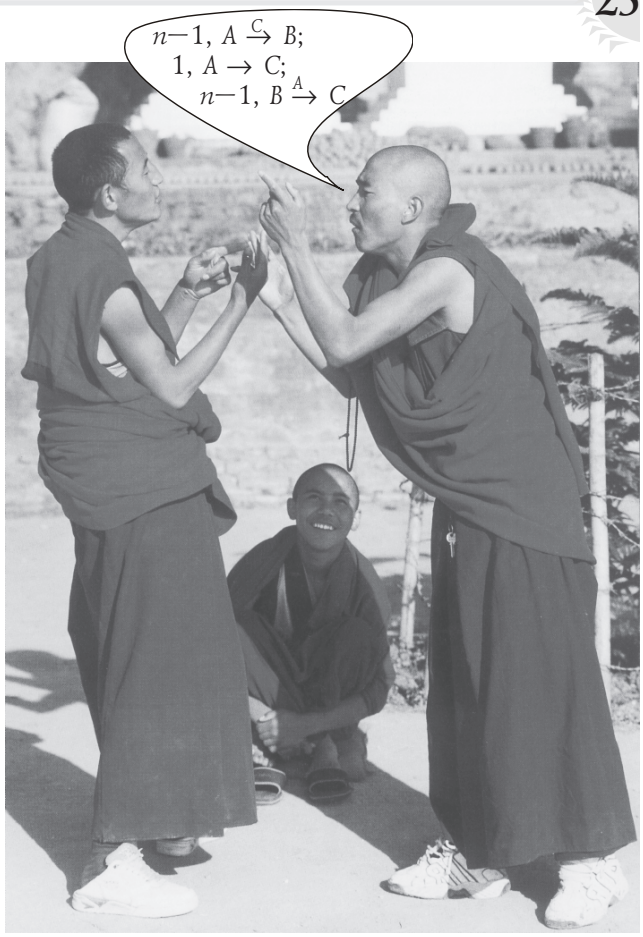
Школьный алгоритмический язык

```
алг вычисление(цел k, цел p)
   арг k
   рез p
нач цел i, t
|      p:= 1; t := 0;
| для i от 1 до k
| нц
| |  t := t + 3; p := p * t;
| кц
| Вывод "Результат", p
кон
```

Продолжение следует

Дышите... не дышите

Эрик Фрэнк Рассел



Когда Тейлора вводили в комнату, кандалы на ногах лязгали, а на руках позвякивала цепь. Из-за оков он шел неуклюже, шаркая ногами. И охранники забавлялись тем, что заставляли его идти быстрее, чем он мог. Ему указали на кресло, стоявшее перед длинным столом. Кто-то толкнул пленника с такой силой, что он потерял равновесие и шлепнулся на жесткое сиденье.

Черные волосы землянина колыхнулись — единственная видимая реакция. Он огляделся. Его глаза были светло-серыми, такими светлыми, что казались ледяными. Взгляд ни дружелюбный, ни враждебный, в нем не читалось ни покорности, ни гнева.

Просто бесстрастный и безразлично-холодный взгляд.

Семеро гомбариан, сидевших с другой стороны стола, рассматривали пленника. Кто с любопытством, кто со скукой, а некоторые — с триумфом и высокомерием. Гомбариане принадлежали к гуманоидам, но ведь и гориллы — гуманоиды. На этом сходство кончалось.

— Итак, — начал гомбарианин, сидевший в центре; он завершал каждое третье слово урчанием, — твое имя Уэйн Тейлор?

Пленник молчал.

— Ты прибыл с планеты, которая называется “Земля”?

Пленник по-прежнему молчал.

— Давай не будем тратить время, Паламин, — предложил сидевший слева. — Если он не хочет говорить добровольно, будет говорить по принуждению.

— Ты прав, Экстер. — Паламин сунул руку под стол и вытащил молоток с грушевидной головкой. — Как тебе понравится, если все кости на твоей руке, палец за пальцем, сустав за суставом, будут раздроблены?

— Вовсе не понравится, — сознался Уэйн Тейлор.

— Очень разумный ответ, — одобрил Паламин и многозначительно положил молоток на середину стола. — Уже потрачено много дней на то, чтобы обучить тебя нашему языку. За это время даже ребенок научился бы понимать вопросы и отвечать на них. — Он удостоил заключенного тяжелым взглядом. — Ты старался быть чересчур тупым учеником. Но больше тебе не удастся вводить нас в заблуждение. Сейчас ты все расскажешь.

— Хочешь ты этого или нет, — вставил Экстер, облизав тонкие губы, — но ты нам все расскажешь.

— Верно, — согласился Паламин. — Давай начнем все сначала и посмотрим, удастся ли нам избежать пыток. Твое имя Уэйн Тейлор и ты прибыл с планеты, которая называется “Земля”?

— Я сознался в этом сразу же, как только меня взяли в плен.

— Знаю. Но тогда ты еще плохо говорил на нашем языке, а мы стремимся избежать всяких неточностей. Зачем ты прибыл на Гомбар?

— Я говорил своему учителю об этом по меньшей мере двадцать раз. Мой корабль был поврежден, и я случайно...

— Тогда зачем ты его взорвал? Почему ты не вошел с нами в открытый контакт? Почему не попросил нас отремонтировать твой корабль?

— Ни один земной корабль не должен попасть в руки врагов, — решительно сказал Тейлор.

— Врагов? — Паламин попытался принять вид оскорбленной добродетели, но его лицо не было к этому приспособлено. — Раз вы, земляне, совсем ничего не знаете о нас, какое вы имеете право считать нас врагами?

— По прибытии меня не расцеловали, — резко ответил Тейлор. — Когда я садился, в меня стреляли. В меня стреляли и когда я убегал. За мной охотились, а когда поймали — избили.

— Наши солдаты выполняли свои обязанности, — с благородным негодованием заметил Паламин.

— Сейчас я был бы уже мертв, если бы они не оказались самыми отвратительными стрелками в районе Лебедя.

— А что такое Лебедь?

— Звезда.

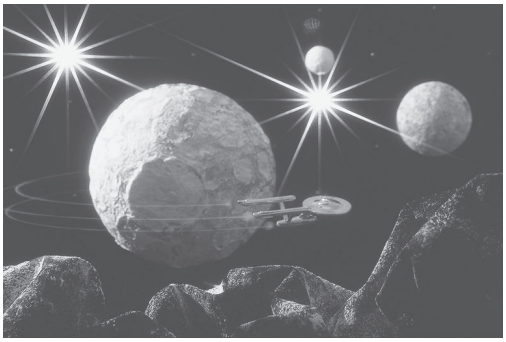
— Кто ты такой, чтобы критиковать наших солдат? — вмешался Экстер, бросив на него сердитый взгляд.

— Землянин, — проинформировал его Тейлор, будто этого более чем достаточно.

— Это для меня ничего не значит, — откликнулся Экстер с нескрываемым презрением.

— Не значит, так будет.

— Если бы вам был желателен дружественный контакт, земные



власти послали бы большой корабль с официальной делегацией на борту, разве не так? — вновь заговорил Паламин.

— Не думаю.

— Почему?

— Мы не рискуем большими кораблями и значительными людьми, если неизвестно, какой их ожидает прием.

— А кто добывает эту информацию?

— Космические разведчики.

— Ага! — Паламин огляделся с гордостью пигмея, который поймал слона. — Так что, в конце концов ты признаешь, что ты — шпион?

— Я — шпион только в глазах неприятеля.

— Наоборот, — вмешался субъект с тяжелой челюстью, сидевший справа. — Раз мы говорим, что ты шпион, то ты шпион и есть.

— Хорошо, будь по-твоему, — уступил Тейлор.

— Это уж нам судить.

— Можешь быть в этом уверен, мой дорогой Боркор, — успокоил его Паламин и снова повернулся к пленнику. — Сколько всего землян?

— Около двенадцати миллиардов.

— Он лжет, — заявил Боркор, жадно пожирая глазами молоток.

— На одной планете не может проживать такое количество разумных существ, — поддержал его Экстер.

— Они разбросаны на нескольких десятках планет, — сказал Тейлор.

— И снова лжет, — настаивал Боркор.

Отмахнувшись от него, Паламин спросил:

— А сколько у них космических кораблей?

— К сожалению, простым космическим разведчикам не доверяют тайн статистики флота, — холодно ответил Тейлор. — Могу лишь сказать, что я не имею ни малейшего представления.

— Какое-то представление ты должен иметь.

— Я могу высказать свои предположения, только ты сам решай, чего они стоят.

— Итак?

— Миллион.

— Чепуха! — заявил Паламин. — Полнейший абсурд.

— Ну и прекрасно! Тысяча... Или любое другое число, которое ты сочтешь разумным.

— Так мы ни к чему не придем, — выразил недовольство Боркор.

— А чего вы ожидали? — обратился к своим соотечественникам Паламин. — Если бы мы послали шпиона на Землю, стали бы мы набивать его секретнейшей информацией? Чтобы он выдал ее врагам, если его поймают? Или мы сказали бы ему ровно столько, чтобы хватило для выполнения поставленной перед ним задачи? Идеальный шпион — это проницательный невежда, способный все воспринять и не способный ничего выдать.

— Идеальный шпион не попался бы в ловушку, — злобно заметил Экстер.

— Спасибо тебе за приветливые слова, — вступил в разговор Тейлор. — Если бы я прибыл сюда как шпион, вы бы не увидели ни моего корабля, ни тем более меня.

— Ну, так куда же ты в таком случае направлялся, прежде чем вынужден был сделать посадку на Гомбар? — спросил Паламин.

— К другой звездной системе.

— Не обращая внимания на эту?

— Да.

— Почему?

— Я летел туда, куда мне было приказано.

— Твои россказни неубедительны. — Паламин откинулся назад и осуждающе посмотрел на него. — Совершенно неправдоподобно, чтобы космический исследователь миновал одну звездную систему ради другой, расположенной дальше.

— Я направлялся к двойной звезде, у которой по меньшей мере сорок планет, — объяснил Тейлор. — А в этой системе их только три, поэтому она считалась менее важной.

— Как, несмотря на то, что мы заселили все три планеты?

— Откуда нам было знать? Никто здесь раньше не пролетал.

— Теперь знаете, — зловеще пробормотал Экстер.

— Знает только он, — поправил Паламин. — И чем дольше не узнают, тем лучше для нас. Когда чуждая форма жизни сует свое рыло в нашу систему, нам нужно время, чтобы собраться с силами.

Все одобрительно закивали.

— Так вот каков ваш образ мышления, — задумчиво протянул Тейлор.

— Что ты имеешь в виду?

— Вы полагаете само собой разумеющимся, что встреча должна вести к столкновению, а затем и к войне.

— Мы были бы первостатейными дураками, если бы думали по-другому и позволили захватить себя врасплох, — подчеркнул Паламин.

Тейлор вздохнул:

— Ныне мы укрепились на сотне планет без единой битвы. Потому что мы не лезем туда, где нас не хотят видеть.

— Представляю себе, — саркастически откликнулся Паламин. — Вам говорят “убирайтесь”, и вы любезно сматываетесь. Это противоречит инстинкту.

— Вашему инстинкту, — возразил Тейлор. — Мы не видим смысла тратить время и деньги на межзвездную войну, когда можно потратить и то и другое на исследование и развитие.

— Так, значит, в вашем флоте нет боевых кораблей?

— Конечно же есть.

— И много?

— Хватит, чтобы справиться с вами.

— Пацифисты, вооруженные до зубов, — объявил Паламин остальным. Те понимающе улыбнулись.

— Лжецы всегда непоследовательны, — произнес Экстер с апломбом. Он уставился каменным взглядом на пленника.

— Если вы так заботливо избегаете неприятностей, то зачем вам нужны боевые корабли?

— Потому что у нас нет гарантий, что вся Вселенная разделяет нашу политику: живи сам и дай жить другим.

— Объясни подробнее.

— Мы никому не досаждаем. Но когда-нибудь кому-нибудь может прийти в голову досадить нам.

— Тогда вы начинаете войну?

— Нет. Начнут войну другие. Мы ее выиграем.

— Чистейшая уловка, — сказал Экстер Паламину и остальным. — Эта технология непонятна разве что идиоту. Они заселили сотню планет.

На большинстве планет никакого сопротивления не было, потому что некому было сопротивляться. На других — туземцы, слабые и отсталые, они знают, что их борьба обречена на поражение, потому и не борются. Но на любой планете, достаточно сильной и решившей сопротивляться — такой, например, как Гомбар, — земляне быстренько примут сопротивление за недозволенное вмешательство в свои дела. Они начнут утверждать, что им угрожают. И это будет их моральным оправданием войны.

Паламин взглянул на Тейлора.

— Что скажешь на это?

Выразительно пожав плечами, Тейлор сказал:

— Политический цинизм такого сорта давно устарел там, откуда я прилетел. Ничем не могу помочь, раз уж вы отстали от нас в умственном развитии на десять тысяч лет.

— Мы собираемся сидеть здесь и позволять пленнику в цепях оскорблять нас? — гневно обратился Экстер к Паламину. — Предлагаю казнить его и всем разойтись по домам. Я, например, по горло сыт пустой болтовней.

— Я тоже, — согласился один из гомбариан; он выглядел закоренелым подхалимом.

— Терпение, — посоветовал Паламин, а потом обратился к Тейлору: — Ты утверждаешь, что тебе было приказано исследовать двойную систему Хплора и Риди?

— Если ты имеешь в виду ближайшую двойную звезду, то да. Таково было мое предписанное направление.

— Предположим, вместо этого тебе бы сказали изучить нашу Гомбарианскую систему. Как бы ты поступил?

— Я подчиняюсь приказам.

— Ты прилетел бы тайно и стал бы потихоньку везде высматривать и вынюхивать?

— Не обязательно. Если бы с первого взгляда вы мне показались дружелюбно настроенными, я бы появился перед вами открыто.

— Он увиливает от ответа, — настаивал Экстер, все так же полный гнева.

— Что бы ты делал, если бы был не уверен в нашей реакции?— продолжал Паламин.

— То же, что делал бы любой на моем месте, — резко ответил Тейлор. — Околачивался бы поблизости, пока не убедился в ваших намерениях.

— Тем временем старался бы избежать плена?

— Конечно.

— И, если тебе не понравилось наше поведение, ты доложил бы о нас как о неприятеле?

— Да.

— Вот и все, что мы хотели узнать, — подытожил Паламин. — Твои признания равносильны признанию в шпионской деятельности. И не имеет значения, приказывали ли тебе сунуть свой излишне любопытный нос в нашу или какую другую систему, все равно ты — шпион. — Он повернулся к остальным. — Все согласны?

Члены судилища хором ответили:

— Да.

— Такие, как ты, достойны лишь одной судьбы, — заключил Паламин. — Ты вернешься в свою камеру и будешь находиться там до времени официальной казни. Уведите его.

Охранники выбили из-под Тейлора кресло и пинками заставили подняться. Они тащили пленника быстрее, чем тот мог идти. Скованный землянин спотыкался и чуть не падал. Но он нашел время бросить от дверей один быстрый взгляд назад, и взгляд его бледных глаз казался ледяным.

Когда тот тюремщик, что постарше, принес ужин, Тейлор спросил:

— А как у вас здесь казнят?

— А как это делают там, откуда ты прилетел?

— Никак.

— Никак? — Тюремщик удивленно моргнул. Поставив поднос на пол, он сел на скамью возле Тейлора, оставив решетку из толстых железных прутьев широко открытой. Рукоятка его пистолета маячила в соблазнительной близости от руки пленника.

— Тогда как же вы справляетесь с опасными преступниками?

— Мы лечим их всеми доступными эффективными средствами, включая операции на мозге. Неизлечимых же мы отвозим на одиночную планету, оставленную исключительно для них. Пусть варятся там в собственном соку.

— Какой расточительный способ, — высказал свое отношение охранник. Будто случайно он вытащил свое оружие, направил на стену и нажал кнопку. Ничего не произошло.

— Не заряжен, — сказал он. Тейлор не откликнулся.

— И не пытайся выхватить его у меня, а тем более удрать. Тут бронированные двери и полным-полно охраны.

— Мне нужно избавиться от наручников, прежде чем затевать что-то, — осторожно проговорил Тейлор. — А как насчет подкупа?

— Это чем же? У тебя нет ничего, кроме одежды, да и ту сожгут, когда ты умрешь.

— Что ж, забудем об этом. — Тейлор громко звенел кандалами. Он выглядел раздраженным. — Ты еще не сказал мне, как я умру.

— Тебя публично задушат, — сообщил тюремщик. Он облизал губы без видимой причины. — Все казни производят в присутствии населения. Мало знать, что справедливость восторжествовала, нужно еще и видеть, как она торжествует. Так что смотрит каждый. К тому же это зрелище дисциплинирует. — Он снова облизнул губы. — Замечательный спектакль.

— Да уж, конечно.

— Тебя поставят на колени спиной к столбу, руки и ноги крепко свяжут, — объяснял тюремщик; со стороны могло показаться, что он читает важное наставление. — В столбе просверлена дырка на уровне твоей шеи. Шею захлестнут веревочной петлей, веревку пропустят сквозь дырку и обмотают вокруг палки другой стороны столба. Палач вращает палку и тем самым затягивает петлю, быстро или медленно, в зависимости от настроения.

— Полагаю, когда к нему приходит артистическое вдохновение, он продлевает агонию жертвы, несколько раз ослабляя и натягивая веревку, — догадался Тейлор.

— Нет, нет, ему запрещено так поступать, — уверил тюремщик, не заметив сарказма в словах землянина. — Не во время финальной казни. Такой метод используется только для того, чтобы добиться признания у упорствующего. Мы — справедливый и мягкосердечный народ, понимаешь?

— Ты меня утешил.

— Так что тебя удушат быстро и эффективно. Я видел множество казней, но пока еще не приходилось видеть непрофессиональных. Тело тяжелеет и провисает на веревках, глаза вылезают из орбит, язык вываливается, чернеет, и наступает смерть. Сделать смерть быстрой и максимально безболезненной — вот в чем заключается искусство палача. Тебе действительно не о чем беспокоиться.

— Похоже, беспокоиться действительно не о чем, судя по тому, как ты все описал, — сухо проговорил Тейлор. — Я воистину наверху блаженства. — Он немного подумал и спросил: — А когда меня поведут на казнь?

— Сразу же после того, как кончится твоя игра, — сообщил тюремщик.

Тейлор недоуменно посмотрел на него:

— Игра? Какая игра? Что ты имеешь в виду?

— У нас принято позволять осужденному сыграть свою последнюю игру против искусного игрока, выставленного нами. Когда игра кончается, осужденного уводят и душат.

— Не важно, выиграет он или проиграет?

— Результат не имеет значения. Его казнят, невзирая на то, победитель он или проигравший.

— По-моему, это безумие, — нахмурившись, сказал Тейлор.

— Это потому, что ты — чужак, — откликнулся тюремщик. — Но ты, конечно, согласишься с тем, что преступник, стоящий перед лицом неминуемой смерти, должен получить некую компенсацию или даже привилегию провести последние минуты в борьбе за жизнь.

— Совершенно бесполезной борьбе.

— Может быть. Но каждая минута отсрочки драгоценна для осужденного. — Тюремщик возбужденно потер руки. — Скажу я тебе, нет ничего более восхитительного, более захватывающего, чем чей-то предсмертный матч с искусным игроком.

— Вот как?

— Понимаешь, осужденный не способен полностью сосредоточиться на игре. С одной стороны, его голова занята мыслями о нависшем над ним роке, в то время как у противника нет такой ноши. С другой стороны, он не дает своему противнику ни выиграть, ни проиграть. Он концентрирует все свои способности на том, чтобы предотвратить окончательный результат, и тянет игру как можно дольше. Конечно же все это время его мысленно и морально подстегивает осознание того, что конец приближается.

— И от этого вы получаете огромное удовольствие, — подытожил Тейлор.

Тюремщик прикусил губу.

— Много раз я видел, как преступник, обливаясь холодным потом, играет с искусством, порожденным безнадежностью. Потом наконец наступает финал. Он падает в обморок и скатывается с кресла. Мы выносим его на казнь вялым, как пустой мешок. Когда он приходит в себя, то уже стоит на коленях лицом к толпе и ждет первого поворота палки.

— Не стоит и беспокоиться, — решил Тейлор. — Никакой игрок долго не протянет.

— Обычно — нет, но я знаю и исключения, хороших и умелых игроков, которые умудрялись оттянуть смерть на четыре-пять дней. Был один такой парень, профессиональный игрок в ализик, так он ухитрился шестнадцать дней избегать конца. Он был так хорош. Какая жалость, что ему пришлось умереть! Многие видеозрители очень жалели, когда наступил конец.

— Так вы передаете эти предсмертные матчи по видео?

— Это — самое популярное шоу. Всех пригвождает к стульям.

— Хм-м-м. — Тейлор немного подумал и спросил: — Предположим, осужденный был бы способен тянуть игру год или больше, позволили бы ему это?

— Конечно. Никого не пошлют на смерть до того, как завершится его последняя игра. Полагаю, это можно назвать суеверием. Более того, правила таковы, что во время игры осужденного хорошо кормят. Если он хочет, то может есть, как король. Все равно они редко едят.

— Почему?

— Они так нервничают, что их желудки отказываются принимать пищу. Некоторые буквально болеют во время игры. Когда я вижу такого, я знаю, что до конца следующего дня он не протянет.

— Ты здесь часто развлекаешься, — предположил Тейлор.

— Очень часто, — согласился тюремщик. — Но не всегда. На плохих игроков поступают жалобы от видеозрителей. Они начинают игру, сразу же все портят, их уводят к позорному столбу, и с ними покончено. Для всех нас огромное удовольствие, когда осужденный ведет настоящую битву.

— Так, значит, у меня никаких шансов. Я не знаю гомбарианских игр, а вы не знаете земных.

— Любой игре можно научиться за короткое время, а выбор принадлежит тебе. Естественно, тебе не позволят выбрать такую игру, ради которой тебя придется выпустить. Нужно выбрать что-то такое, во что можно играть, находясь в этой камере. Хочешь добрый совет?

— Давай.

— Сегодня вечером прибудет официальное лицо, чтобы обговорить условия состязания, после чего он подберет тебе подходящего партнера. Не проси, чтобы тебя обучили одной из наших игр. И, каким бы хитроумным ты ни старался быть, твой противник будет лучше, потому что ему все будет знакомо, а тебе — чуждо. Выбери одну из игр своей планеты и, таким образом, получишь преимущество.

— Спасибо за совет. Это бы помогло, если бы поражение означало смерть, а победа — жизнь.

— Я тебе уже сказал, что результат не имеет значения.

— Вот я и говорю. Какой смысл выбирать?

— Ты выбираешь между смертью утром и смертью следующим утром. — Тюремщик встал со скамьи, вышел, запер решетку. — В любом случае я принесу тебе описание наших комнатных игр. Тебе хватит времени прочитать его до прихода официального лица.

— Очень мило с твоей стороны, — вздохнул Тейлор. — Но, мне кажется, ты зря тратишь свое время.

***

Уэйн Тейлор остался один и задумался. Нельзя сказать, чтобы мысли его были приятными. Профессия космических разведчиков является одной из самых рискованных, и никто не знает этого лучше, чем сами разведчики. Каждый охотно относится к опасностям по древнему принципу, считая, что они всегда случаются с кем-то другим. На этот раз он сам попал в опасное положение. Тейлор провел пальцем по внутренней части воротничка, который показался ему чересчур тугим...

Когда он вынырнул из облаков и наткнулся на два самолета, те тут же открыли огонь по иллюминаторам и люкам, Тейлор мгновенно нажал кнопку сигнала тревоги “Д”. Тут же его радиопередатчик начал передавать короткое сообщение, в котором указывались его координаты, и эта планета была определена как вражеская территория.

Чуть раньше, за тысячу миль от этой планеты в космосе, он сообщил о своем намерении совершить вынужденную посадку и передал те же самые координаты. Таким образом, это подтвердит его первое сообщение. По прикидкам Тейлора, за это время сигнал тревоги должен был быть отправлен по меньшей мере сорок раз.

Сразу же после посадки он включил взрыватель замедленного действия и пустился наутек. Самолеты все еще жужжали где-то поблизости. Один из них спикировал на приземлившийся корабль как раз тогда, когда произошел взрыв. Самолет исчез во вспышке. Другой набрал высоту и кружил наверху, руководя поисками. Судя по скорости, с которой появились военные отряды, Тейлора угораздило шлепнуться в милитаризованный район, набитый болванами в униформе. Тем не менее он заставил их побегать шесть часов и покрыл расстояние в двадцать миль, прежде чем его взяли. Тогда он выразил свое неудовольствие кулаками и ногами.

Так что теперь он не знал, получили ли земные станции, работающие на прием, его повторяющийся “Д”-сигнал. Скорее всего получили, потому что это был канал высшего приоритета, на котором поддерживалась круглосуточная вахта. И Тейлор ни на мгновение не сомневался, что, получив сообщение, его не оставят на произвол судьбы.

Но что бы они ни предприняли, будет слишком поздно. В этом самом секторе патрулировал “Маклин”, новейший, самый большой и наилучшим образом вооруженный линкор. Если “Маклин” окажется неподалеку, то и для него, чтобы долететь до Гомбара с максимальной скоростью, потребуется десять месяцев. А если он вернулся в порт (его время от времени заменяют более старыми и медлительными кораблями), то задержка может достигнуть двух лет.

Два года — слишком долго. Впрочем, десять месяцев тоже долго. Он не может ждать и десяти недель. На самом деле, весьма вероятно, у него нет и десяти дней. Ох, время, время, почему это человек не может растянуть тебя, как резину?

Вернулся охранник, просунул книгу сквозь прутья.

— Вот, возьми. Ты достаточно хорошо знаешь наш язык, чтобы понять, о чем тут написано.

— Спасибо.

Тейлор растянулся во весь рост на скамье и принялся быстро, но внимательно читать. Некоторые страницы он пролистывал, бегло просмотрев, потому что игры описывались слишком коротко, упрощенно, по-детски, и на них не стоило обращать внимания. Он не удивился, обнаружив, что некоторые игры были чуждыми вариациями широко известных земных игр. Гомбариане играли в карты, например, но их было восемьдесят в колоде, с десятью мастями.

Ализик оказался большей и усложненной версией шахмат, с четырьмястами полями на доске и сорока фигурками у каждого игрока. Эту игру некто растянул на шестнадцать дней. И она была единственной в книге, которую, по-видимому, можно было так долго тянуть. Некоторое время Тейлор обдумывал ализик, прикидывая, смогут ли власти и видеоаудитория вынести игру, в которой делают один ход за десять часов. Он сомневался. В любом случае он не может запретить своему искусному партнеру сделать ответный ход за пять секунд.

Да, вот что ему нужно на самом деле. Игра, которая замедляет его партнера, несмотря на его попытки ускорить игру. Игра, которая со всей очевидностью игра, а не трюк, чтобы каждый дурак мог с первого взгляда видеть, что она определенно имеет конец. Однако это должна быть такая игра, какую его противник не мог бы закончить, выиграть или проиграть, как бы сильно он ни старался.

Не было такой игры на всех трех планетах Гомбара на сотню миров от Земли или на мириадах еще не открытых планет. Такой игры не могло быть, потому что, если бы она и была, в нее бы никто не играл. Людям нужен результат. Нет таких дураков, которые станут тратить время, скакать на игрушечной лошадке, которая никуда не привезет, потворствовать пустому вздору, который может надоесть самым отъявленным занудам.

Нет таких дураков!

Нет?

Он встал со скамьи и принялся мерить шагами камеру, словно встревоженный тигр.

Окончание следует

ДОРОГИЕ УЧИТЕЛЯ, ДОРОГИЕ КОЛЛЕГИ, ДОБРО ПОЖАЛОВАТЬ НА ПРАЗДНИК!

Двадцать два весенних профессиональных праздника!

С 1 по 30 апреля 2002 года в Московском городском доме учителя пройдет фестиваль “Московский педагогический марафон учебных предметов”, организованный Московским комитетом образования совместно с Издательским домом “Первое сентября”.

Каждый рабочий день апреля будет посвящен одному из школьных предметов.

РАСПИСАНИЕ ПЕДАГОГИЧЕСКОГО МАРАФОНА

1 апреля	День учителя русского языка
2 апреля	День учителя литературы
3 апреля	День учителей мировой художественной культуры, музыки и ИЗО
4 апреля	День учителя истории
5 апреля	День школьного библиотекаря
8 апреля	День учителя географии
9 апреля	День учителя биологии
10 апреля	День учителя химии
11 апреля	День учителя физики
12 апреля	День учителя математики
15 апреля	День учителя информатики
16 апреля	День учителя английского языка
17 апреля	День учителя немецкого языка
18 апреля	День учителя французского языка
19 апреля	День учителей технологии, профориентации и ОБЖ
22 апреля	День учителя физкультуры
23 апреля	День здоровья детей
24 апреля	День дошкольного образования
25 апреля	День учителя начальной школы
26 апреля	День логопедов и коррекционных педагогов
29 апреля	День школьного психолога
30 апреля	День школьной администрации

В рамках каждого Дня педагогического марафона с 10 утра до 18 вечера одновременно на нескольких площадках уютного, красивого, вновь отреставрированного Московского городского дома учителя пройдут встречи с известными педагогами, авторами популярных и новых учебников, учеными, методистами, состоятся «круглые столы» по вопросам интеграции учебных предметов, аттестации, тестирования (ЕГЭ), презентации новых подходов в преподавании, включая актуальные сегодня проектные и дистанционные формы обучения. В завершение каждого дня — добрые сюрпризы: музыка, авторские песни, капустники, удивительные фильмы, — то есть настоящий праздник.

Цель марафона — дать возможность встретиться в **своем** московском Доме представителям каждой педагогической профессии (включая работников дошкольного воспитания), чтобы в профессиональном кругу обсудить насущные проблемы, поделиться умениями и успехами, окунуться в атмосферу праздника.

Опыт Издательского дома “Первое сентября” подсказывает, что интересы учителей-предметников различаются: разные стили, разные подходы, разные взгляды на общие проблемы. Можно сказать, что у каждого предмета, у каждого учителя-предметника своя гордость. Но эта гордость за свой предмет, за свою профессию как раз и объединяет всех учителей. В этой общности и разнообразии — красота школьной жизни, залог ее полноты. Дни предметов будут разными — под стать каждому предмету, но суть марафона — красивая, умная, добрая, обращенная на ребенка, успешная, профессиональная школа. Каждый из Дней марафона — серьезная, интересная работа в атмосфере профессионального праздника учителей-предметников и их коллег.

Программы каждого дня публикуются в соответствующих изданиях Издательского дома “Первое сентября” и других профессиональных изданиях. Программа всех Дней марафона одновременно будет опубликована в газете “Первое сентября”. Также ближе к апрелю по школам Москвы и Подмосковья будет разослана брошюра с программой.

Участие в марафоне бесплатное. Мы постараемся сделать так, чтобы интересно было в любой момент, когда бы и на сколько бы вы ни пришли — на два часа или на весь день. Параллельные мероприятия позволят вам выбрать что-то по душе. А если кому-то захочется просто пообщаться с коллегами за чашкой кофе или чая — кафе встреч будет открыто весь день.

Добро пожаловать в Дом учителя на ваш профессиональный праздник!

Адрес Дома учителя: **ул. Пушечная, д. 4, стр. 2, станция метро “Кузнецкий мост”**.

До встречи!
С искренним уважением,
руководители фестиваля
“Московский педагогический марафон”

По любым вопросам и с предложениями, касающимися проведения марафона, обращайтесь, пожалуйста, в оргкомитет по телефону/факсу: **249-31-38, 249-31-84**; по адресу: **Москва, Киевская, 24, редакция “Первое сентября”** (с пометкой на конверте “Марафон”), или воспользуйтесь электронным адресом (e-mail) **alpha@1september.ru**. Подробную информацию можно найти также в Интернете на сайте Издательского дома: www.1september.ru.

План проведения Дня учителя информатики в Московском городском доме учителя

15 апреля, ул. Пушечная, дом 4, строение 2,
станции метро "Лубянка" или "Кузнецкий мост"

Время проведения мероприятия	БОЛЬШОЙ ЗАЛ Секция **"Преподавание информатики в начальной, средней и старшей школе"**	МАЛЫЙ ЗАЛ Секция **"Страницы повышения квалификации"**	ПАРКЕТНЫЙ ЗАЛ Секция **"Мастер-классы"**
10^{00} — 10^{30}	**Открытие.** **Пленарное заседание.** Тема пленарного заседания: **"Место курса информатики в действующем и экспериментальном учебных планах"**.		
10^{30} — 10^{50}	Перерыв		
10^{50} — 12^{10}	**Семинар "Информатика в начальной школе"**. Семинар проводит *Ю.А. Первин*	**Семинар "Олимпиады по информатике. Пути к вершине"**. Семинар проводит *Е.В. Андреева*	**Семинар "Виртуальная школа — всепроникающие информационные технологии"**. Семинар проводит *А.И. Сенокосов*
12^{10} — 12^{30}	Перерыв		
12^{30} — 13^{50}	**Семинар "Базовый курс информатики"**. Семинар проводит *Н.Д. Угринович*	**Семинар "Современные педагогические технологии и частные методики обучения информатике"**. Семинар проводит *И.Н. Фалина*	**Семинар "Роботландский сетевой университет"**. Семинар проводит *В.А. Козлова*
13^{50} — 14^{10}	Перерыв		
14^{10} — 15^{30}	**Семинар "Профильные курсы информатики"**. Семинар проводит *Е.К. Хеннер*	**Семинар "Введение в специальность "учитель информатики"**. Семинар проводит *А.Г. Гейн*	**Семинар "Олимпиады по базовому курсу информатики"**. Семинар проводит *С.В. Русаков*
15^{30} — 15^{50}	Перерыв		
15^{50} — 16^{50}	**Пленарное заседание**. Тема пленарного заседания: **"Обновленное содержание курса информатики, стандарты, итоговая аттестация"**. **Закрытие**		
16^{50} — 17^{10}	Перерыв		
17^{10} — 18^{00}	**Концерт**		

*В течение всего дня в **Педагогической гостиной** Дома учителя работает музей истории вычислительной техники, а в **Голубой гостиной** – выставка-продажа учебной литературы.*

LZ и LZW

Окончание. Начало на с. 1

Потоки информации, с которыми приходится иметь дело компьютеру, содержат, как правило, повторяющиеся цепочки символов. Наглядный пример — текст, где неоднократно употребляются те или иные слова. В алгоритме RLE (***R**un **L**ength **E**ncoding* — "групповое" кодирование), который первоначально разрабатывался специально для хранения графической информации, предусматривается замена одинаковых символов [1—5]. Так, строка из *N* одинаковых чисел *m* (*m*, ..., *m*) заменяется двумя числами: *N*, *m*. Данный метод применяется, в частности, для сжатия файлов графического формата PCX, а также как составная часть алгоритма JPEG (являющегося уже более 10 лет основным алгоритмом сжатия графической информации с потерями). Усовершенствованный алгоритм RLE используется в одном из вариантов формата TIFF и в формате TGA.

В 1977 году два израильских математика, А.Лемпел и Я.Зив, представили новый класс алгоритмов сжатия без потерь. В них предусматривалась замена уже не только цепочек одинаковых символов, но и повторяющихся последовательностей. Одновременно с описанием общих идей был представлен и основанный на этих идеях алгоритм LZ77. Через год Лемпел и Зив предложили усовершенствованный вариант, который стал известен как LZ78. На основе этих разработок впоследствии было создано множество методов сжатия информации (LZ-алгоритмы). Алгоритмы такого типа используются, например, в утилитах PKZIP, WinZIP, ARJ, LHARC и некоторых других программах-упаковщиках, а также в протоколе V.42bis (расширение стандарта V.42, разработанного для обмена данными между модемами) [1, 2, 6, 7].

Недостатком классического варианта алгоритма LZ является то, что он эффективен лишь для повторяющихся цепочек длиной не менее пяти байт. В 1983 году сотрудник компании Unisys Т.Уэлч нашел способ усовершенствовать классический LZ-алгоритм таким образом, чтобы было выгодно работать с повторяющимися цепочками начиная с двух байт. Этот его вариант стал называться "алгоритмом Лемпела — Зива — Уэлча" (LZW).

Предположим, что найдена цепочка символов *A*, которая содержит цепочку *B* и больше нее всего на один символ *C*. В классическом алгоритме LZ цепочки *A* и *B* записываются и обрабатываются отдельно или же берется большая из них, т.е. *A*. В алгоритме LZW последовательность *A* будет записана как "*C*, ссылка на *B*" [1]. Таким образом, не только уменьшается объем сохраняемой в файле информации, но и значительно повышается быстродействие. В отличие от классического LZ-алгоритма, который стал общедоступным, алгоритм LZW был запатентован компанией Unisys.

Алгоритм LZW стал употребляться в самых разных целях, но особенно удачным оказалось его использование для сжатия изображений, передаваемых по компьютерным сетям. На основе этого алгоритма в компании CompuServe, которая владела одной из крупнейших компьютерных сетей в США, в 1987 году был создан формат GIF (***G**raphics **I**nterchange **F**ormat* — формат обмена графическими файлами) [1, 3—5].

Литература

1. *Васильев А.* Сжатие изображений: вчера, сегодня, завтра // Hard'n'Soft № 4/2001.

2. *Борзенко А.Е., Федоров А.Г.* Мультимедиа для всех. Изд. 2-е. М.: КомпьютерПресс, 1996.

3. *Мостицкий И.Л.* Новейший англо-русский толковый словарь по современной электронной технике. М.: ЛУЧШИЕ КНИГИ, 2000.

4. *Пройдаков Э.М., Теплицкий Л.А.* Англо-русский толковый словарь по вычислительной технике, Интернету и программированию. Изд. 2-е, испр. и доп. М.: Издательско-торговый дом "Русская редакция", 2000.

5. *Шниер М.* Толковый словарь компьютерных технологий: Пер. с англ. Киев: ДиаСофт, 2000.

6. PKZIP и PKUNZIP // Информатика № 46/99.

7. Попробуйте упаковать // Информатика № 26/2001.

Гл. редактор
С.Л. Островский
Зам. гл. редактора
А.И. Сенокосов
Редакция:
Е.В. Андреева
Н.Л. Беленькая
Л.Н. Картвелишвили
Н.П. Медведева
Дизайн и верстка:
Н.И. Пронская
Корректоры:
Е.Л. Володина,
С.М. Подберезина

Адрес редакции и издателя:
121165, Киевская, 24
тел. 249-48-96
Отдел рекламы
тел. 249-98-70

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта изданий 32744

Учредитель: ООО "Чистые пруды"

Зарегистрировано в Министерстве РФ по делам печати. ПИ № 77-7230 от 12.04.2001.
Отпечатано в ОИД "Медиа-Пресса",
125993, ГСП-3, Москва, А-40, ул. "Правды", 24.
Тираж 7000 экз.
Срок подписания в печать по графику 28.02.2002.
Номер подписан 28.02.2002.
Заказ №
Цена свободная

Тел.: (095)249-31-38, 249-33-86. Факс (095)249-31-84

Internet: inf@1september.ru
WWW: http://www.1september.ru

ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ», ГЛАВНЫЙ РЕДАКТОР — А.СОЛОВЕЙЧИК

Газеты ИЗДАТЕЛЬСКОГО ДОМА: **Первое сентября** — гл. ред. Е.Бирюкова, **Английский язык** — гл. ред. А.Громушкина, **Библиотека в школе** — гл. ред. О.Громова, **Биология** — гл. ред. Н.Иванова, **Воскресная школа** — гл. ред. монах Киприан (Ященко), **География** — гл. ред. О.Коротова, **Дошкольное образование** — гл. ред. М.Аромштам, **Здоровье детей** — гл. ред. А.Лекманов, **Информатика** — гл. ред. С.Островский, **Искусство** — гл. ред. Н.Исмаилова, **История** — гл. ред. А.Головатенко, **Литература** — гл. ред. Г.Красухин, **Математика** — гл. ред. И.Соловейчик, **Начальная школа** — гл. ред. М.Соловейчик, **Немецкий язык** — гл. ред. М.Бузоева, **Русский язык** — гл. ред. Л.Гончар, **Спорт в школе** — гл. ред. Н.Школьникова, **Управление школой** — гл. ред. А.Адамский, **Физика** — гл. ред. Н.Козлова, **Французский язык** — гл. ред. Г.Чесновицкая, **Химия** — гл. ред. О.Блохина, **Чудесная газета** — гл. ред. М.Аромштам, **Школьный психолог** — гл. ред. М.Сартан.