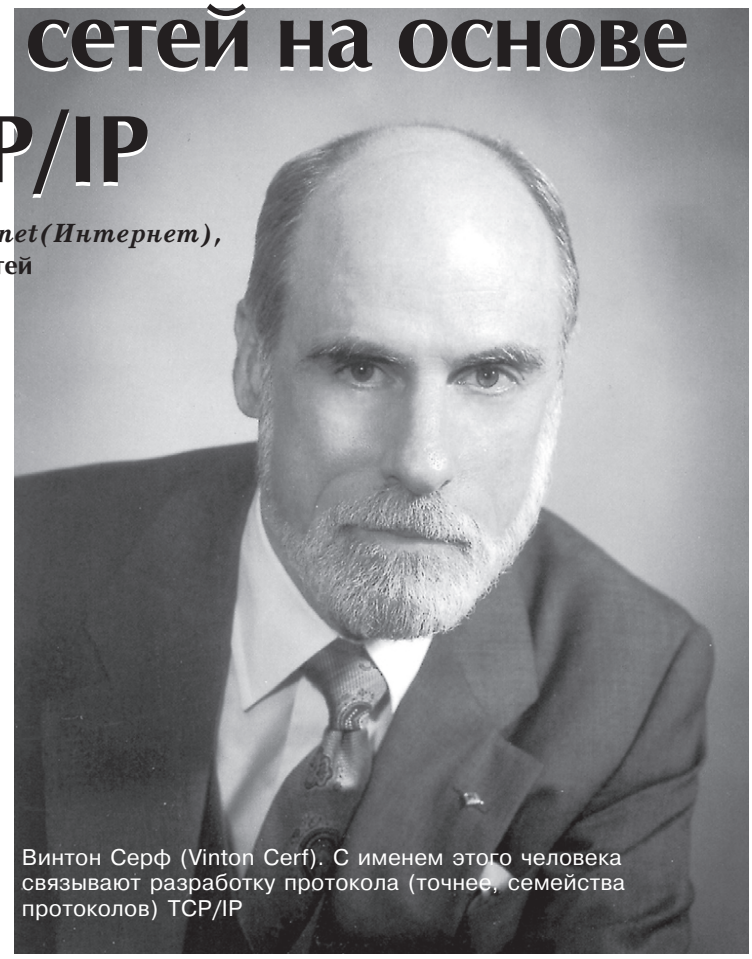
№ 6 (343) 8–15 февраля 2002

ИНФОРМАТИК

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

Объединение сетей на основе протокола TCP/IP

20 лет назад, в 1982 году, возник термин *Internet (Интернет)*, обозначающий объединение компьютерных сетей в соответствии с протоколом TCP/IP



Винтон Серф (Vinton Cerf). С именем этого человека связывают разработку протокола (точнее, семейства протоколов) TCP/IP

1982 год можно считать датой рождения сети Интернет, какой мы знаем ее сегодня (хотя надо принять во внимание, что многие сети, составляющие Интернет, и используемые в ней технологии были созданы раньше).

[Протокол TCP/IP (***T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol*) — совокупность стандартов (протоколов, соглашений) для передачи данных с одного компьютера, подключенного к Интернету, на другой. Создан в 1974 году. В 1988 году был рекомендован в качестве базового для компьютерных сетей.]

Каждый день Интернет "вырастает" примерно на миллион электронных страниц, которые добавляются к уже действующим сотням миллионов. Кто же управляет этим огромным хозяйством? Строго говоря, никто. У сети нет единого хозяина или единого управляющего центра. Тем не менее различные организации выполняют необходимую работу, связанную со стандартизацией и координацией функционирования сети, а также разрабатывают для нее документацию. Вот некоторые из них.

- Сообщество Интернета (***I**nternet **Soc**iety*, ISOC);
- Рабочая группа инженеров Интернета (***I**nternet **E**ngineering **T**ask **F**orce*, IETF);
- Исследовательская группа Интернета (***I**nternet **R**esearch **T**ask **F**orce*, IRTF);
- Группа управления инженерами Интернета (***I**nternet **E**ngineering **S**teering **G**roup*, IESG);
- Совет по архитектуре Интернета (***I**nternet **A**rchitecture **B**oard*, IAB);
- Центр сетевой информации Интернета (***Inter**net **N**etwork **I**nformation **C**enter*, InterNIC).

Они регулируют работу Интернета в целом, оказывают помощь организациям, занимающимся развитием сети, проводят обсуждение технических и организационных вопросов, координируют исследования в области технологий Интернета, решают проблемы, касающиеся безопасности данных, и некоторые другие вопросы.

ISOC занимается разработкой стандартов и протоколов, тесно сотрудничая с другими организациями, например, с IAB. Задача IAB — наблюдение за архитектурой и развитием протоколов, создание новых стандартов (и документов, описывающих протоколы и стандарты Интернета), публикация материалов, посвященных Интернету.

Основная задача IETF — разработка протоколов, связанных с управлением сетями, а также со служебными и пользовательскими прикладными программами.

InterNIC распределяет адреса IP и доменные имена, оказывает справочные и информационные услуги.

Существует также Российский научно-исследовательский институт развития общественных сетей — РОСНИИРОС (***R**ussian **I**nstitute for **P**ublic **N**etworks*, RIPN), который занимается координацией российских исследований и разработок в Интернете, регистрацией российских доменных имен, "сопровождением" документации и решением других задач подобного рода.

Олимпиады по информатике. Пути к вершине

Лекции читает Е.В. Андреева

Лекция 7. Динамическое программирование

Основные определения

Многие олимпиадные задачи, а также задачи практического программирования являются задачами на перебор вариантов и выбор среди этих вариантов допустимого или наилучшего по тому или иному критерию. Однако рассмотреть все варианты в силу чрезвычайно большого их количества зачастую не представляется возможным [1].

К счастью, для ряда задач, сходных по формулировке с проблемами, действительно требующими полного перебора вариантов, можно найти гораздо более эффективное решение. Чаще всего в таких случаях решение сводится к нахождению решений подзадач меньшей размерности, которые запоминаются в таблице и никогда более не пересчитываются, а подзадачи большей размерности используют эти уже найденные решения. Такой метод называется *динамическим программированием*, еще его называют табличным методом. В общей же форме под динамическим программированием понимают процесс пошагового решения задачи оптимизации, при котором на каждом шаге из множества допустимых решений выбирается одно, оптимизирующее заданную целевую или критериальную функцию. Иногда вместо оптимизационной тем же методом решается задача подсчета количества допустимых решений. В этом случае на каждом шаге вместо выбора оптимального решения производится суммирование решений подзадач меньшей размерности, причем они по формулировке не всегда полностью совпадают с исходной задачей (соответствующие примеры будут рассмотрены ниже). В обоих случаях найденное на текущем шаге решение обычно заносится в таблицу. Как правило, связь задач и подзадач формулируется в виде некоторого "принципа оптимальности" и выражается системой уравнений (рекуррентных соотношений).

Основы теории динамического программирования были заложены Р.Беллманом (см., например, [2]). Описание данного метода и примеры различных задач, которые можно им решать, приведены также в [3—6]. Заметим, что слово "программирование" в приведенном названии (*dynamic programming*), так же как и в "линейном программировании" (*linear programming*), **не** означает составление программ для компьютера.

Для решения задачи оптимизации, в которой требуется построить решение с максимальным или минимальным (оптимальным) значением некоторого параметра, алгоритм, основанный на динамическом программировании, можно сформулировать так:

1) выделить и описать подзадачи, через решение которых будет выражаться искомое решение,
2) выписать рекуррентные соотношения (уравнения), связывающие оптимальные значения параметра для подзадач,
3) вычислить оптимальное значение параметра для всех подзадач,
4) построить само оптимальное решение, используя полученную информацию.

Если нас интересует только значение параметра, то шаг 4 в алгоритме не нужен (такая ситуация характерна, например, для задач подсчета количеств допустимых вариантов или некоторых конфигураций, в том числе и комбинаторных). Однако в случае необходимости построения самого́ оптимального решения иногда приходится в процессе выполнения шага 2 алгоритма получать и хранить дополнительную информацию. Зачастую именно шаг 4 оказывается самым сложным при реализации подобных алгоритмов.

Условия применимости динамического программирования

Попробуем формализовать свойство задачи, которое позволяет применять указанный метод. Пусть мы нашли решение исходной задачи, тогда любая часть этого решения или любая подзадача, через которую выражается решение исходной задачи, также, в свою очередь, является решением аналогичной задачи меньшей размерности. То есть оптимальное решение задачи содержит оптимальные решения ее подзадач. Чтобы убедиться, что задача обладает этим свойством, надо показать, что, улучшая решение подзадачи, мы улучшим и решение исходной задачи.

Как только свойство оптимальности для подзадач установлено, обычно становится ясно, с каким именно множеством подзадач будет иметь дело алгоритм. А так как динамическое программирование, в сущности, вычисляет решение для всех подзадач, то количество различных подзадач, непосредст-

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

венно через которые выражается решение задач большей размерности, должно быть таким, чтобы было возможно все их (или по крайней мере ту часть, которая непосредственно необходима для решения задач большей размерности) запомнить в таблице. Это и есть второе свойство задачи, существенное при использовании динамического программирования. Благодаря этому свойству при рекурсивном способе решения подобных проблем мы многократно выходим на одни и те же подзадачи. В таком случае говорят, что у переборной задачи (оптимизационной или подсчета количеств) имеются перекрывающиеся подзадачи. В типичных случаях (но не всегда!!!) количество различных подзадач полиномиально зависит от размера исходных данных. Алгоритм, основанный на динамическом программировании, решает каждую из подзадач единожды и заносит его в таблицу. Когда эта же подзадача встречается снова, программа не тратит время на ее решение, а берет готовый ответ из таблицы. Поэтому общее время решения пропорционально размеру заполняемой таблицы в целом, хотя зачастую на этапе реализации алгоритма хранению подлежит лишь ее часть, например, одна строка. В любом случае это время существенно меньше времени перебора всех возможных вариантов с целью выбора среди них лучшего или подсчета общего количества допустимых вариантов.

Классические задачи динамического программирования

Задача 1. В таблице $N \times N$ клетки заполнены случайным образом цифрами от 0 до 9. Найти маршрут из клетки $A(1, 1)$ в клетку $A(N, N)$ такой, что:

1) он будет состоять из отрезков, соединяющих центры клеток, имеющих общую сторону;
2) длина маршрута минимально возможная;
3) из всех маршрутов, удовлетворяющих условиям (1) и (2), искомый маршрут тот, сумма цифр в клетках которого максимальна.

Решение. Пусть клетка (1, 1) — это левый верхний угол таблицы, а (N, N) — соответственно, правый нижний угол. Из условия (2) задачи следует, что за каждый шаг мы будем продвигаться по таблице либо на шаг вправо, либо на шаг вниз, что сразу нам гарантирует минимальность в длине пути и избавляет от анализа вариантов по данному критерию. Рассмотрим произвольную клетку таблицы (i, j). В нее мы можем прийти или из клетки $(i-1, j)$ или из $(i, j-1)$. Тогда, если мы уже знаем оптимальные маршруты из клетки (1, 1) в каждую из этих двух клеток, оптимальным маршрутом в клетку (i, j) будет подмаршрут с максимальной из двух сумм суммой плюс отрезок, соединяющий (i, j) с концом выбранного подмаршрута. Оптимальные маршруты из (1, 1) в (1, 2) и (2, 1) определены однозначно. Зная их, по указанному выше способу мы найдем оптимальные маршруты в (1, 3), (2, 2), (3, 1) и запишем их в соответствующих клетках таблицы (записывать нужно только сумму цифр маршрута и направление его последнего отрезка). Этот процесс можно продолжить, пока вся таблица не будет заполнена, причем заполнять ее можно по строчкам слева направо. В клетке (N, N) мы в итоге получим значение суммы цифр искомого маршрута и последний его отрезок. По такой таблице легко восстановить и весь маршрут, начиная с клетки (N, N).

Рассмотрим любую часть оптимального маршрута, например, между клетками $(i1, j1)$ и $(i2, j2)$. Докажем, что эта часть маршрута является решением исходной задачи для указанных клеток. Пусть это не так и существует маршрут с большей суммой, соединяющий эти клетки и имеющий такую же длину. Тогда и для клеток (1, 1) и (N, N) мы можем построить лучший маршрут, используя отрезки, соединяющие (1, 1) и $(i1, j1)$, а также $(i2, j2)$ и (N, N) из старого маршрута плюс улучшенный маршрут из $(i1, j1)$ в $(i2, j2)$, а это противоречит тому, что мы изначально рассматривали часть из уже оптимального маршрута. Значит, любая часть оптимального маршрута, в свою очередь, является оптимальной.

Программа решения данной задачи приведена, например, в [1].

Задача 2. Наибольшая общая подпоследовательность (НОП).

Подпоследовательность можно получить из некоторой конечной последовательности, если удалить из последней некоторое множество ее элементов (возможно, пустое). Например, *BCDB* является подпоследовательностью последовательности *ABCBDAB*. Будем говорить, что последовательность Z является общей подпоследовательностью последовательностей X и Y, если Z является подпоследовательностью как X, так и Y. Требуется для двух последовательностей X и Y найти общую подпоследовательность наибольшей длины. Заметим, что НОП может быть несколько.

Решение. Задача о НОП обладает свойством оптимальности для подзадач. Здесь подходящее множество подзадач — множество пар префиксов (начальных частей) двух данных последовательностей. Покажем это. Пусть $Z = \{z_1, z_2, ..., z_k\}$ — одна из НОП для $X = \{x_1, x_2, ..., x_m\}$ и $Y = \{y_1, y_2, ..., y_n\}$. Если $x_m = y_n$, то $z_k = x_m = y_n$ (в противном случае мы могли бы дописать $x_m = y_n$ в конец последовательности Z и получить НОП длины $k + 1$). Кроме того, Z_{k-1} является НОП для X_{m-1} и Y_{n-1} (если это не так, то мы можем к НОП X_{m-1} и Y_{n-1} дописать $x_m = y_n$ и получить НОП для X и Y более длинную, чем Z. Если $z_k \neq x_m$, то, так как Z — НОП для X и Y, она тем более является НОП для X_{m-1} и Y. Аналогично если $z_k \neq y_n$, то Z — НОП для X и Y_{n-1}. Максимальное количество подзадач, которое нам может понадобиться решить, равно $m \cdot n$ (для каждой пары префиксов X и Y). Это позволяет использовать при решении динамическое программирование. Выпишем рекуррентное соотношение между длинами НОП в подзадачах, обозначив за $a[i, j]$ длину НОП для X_i и Y_j:

$$a[i,\ j] = \begin{cases} 0, & i = 0 \text{ или } j = 0, \\ a[i-1,\ j-1] + 1, & i, j > 0,\ x_i = y_j, \\ \max(a[i-1,\ j],\ a[i,\ j-1]), & i, j > 0,\ x_i \neq y_j. \end{cases}$$

Приведем программу решения этой задачи для последовательностей, состоящих из не более чем 250 символов:

```
var x, y, z : string;
    a : array[0..250,0..250] of byte;
    i, j : byte;
begin
  readln(x);
  readln(y);
  fillchar(a,sizeof(a),0);
  for i := 1 to length(x) do
  for j := 1 to length(y) do
    if x[i] = y[j] then
       a[i, j] := a[i - 1, j - 1] + 1
    else if a[i - 1, j] >= a[i, j - 1] then
            a[i, j] := a[i - 1, j]
         else a[i, j] := a[i, j - 1];
  {длина НОП найдена в a[length(x),length(y)]}
  z := '';{строим саму НОП}
  i := length(x);
  j := length(y);
  while (i > 0) and (j > 0) do
    if x[i] = y[j] then
      begin
        z := x[i] + z;
        i := i - 1; j := j - 1
      end
    else if a[i - 1, j] >= a[i, j - 1] then
            i := i - 1 else j := j - 1;
  writeln(z)
end.
```

Задача 3. Задача об оптимальной расстановке скобок. Рассмотрим ее, например, в такой формулировке. В арифметическом выражении, операндами которого являются целые числа, а операциями — бинарные арифметические операции "+" и "×", расставить скобки так, чтобы результат оказался максимальным (исключение других арифметических операций не снижает в данном случае общности рассмотрения проблемы).

Решение. Прежде чем применять динамическое программирование к этой задаче, стоит убедиться, что простой перебор всех расстановок скобок не даст эффективного алгоритма. Из комбинаторики известно, что количество различных расстановок скобок между n операндами P_n равно числу Каталана с номером $n - 1$ (см., например, [7]), а именно:

$$P_n = \frac{C_{2(n-1)}^{n-1}}{n} > c\frac{4^n}{n^{3/2}},$$

— где c — некоторая константа, не зависящая от n (такую нижнюю оценку обозначают $\Omega(4^n/n^{3/2})$ [4]). То есть число расстановок скобок экспоненциально зависит от n, так что полный перебор неэффективен.

Подзадачами, через которые мы будем выражать оптимальное решение, будут задачи об оптимальной расстановке скобок в последовательностях наших операндов, начиная с i-го и заканчивая j-м. Запоминать результат оптимального решения соответствующей подзадачи мы будем в элементе $a[i, j]$ матрицы размером $n \times n$, диагональные элементы которой ($a[i, i]$) равны операндам, а для $i > j$ все элементы равны 0. Пусть мы хотим подсчитать значение арифметического выражения для операндов, начиная с i-го и заканчивая j-м для $i < j$. Если мы предположим, что последней будет выполняться арифметическая операция, расположенная после операнда с номером k ($i \le k < j$), и эта операция — сложение, то результат подсчета будет равен сумме элементов $a[i, k]$ и $a[k + 1, j]$ нашей матрицы. С умножением ситуация несколько более сложная. Так как операндами могут быть и отрицательные числа, то для нахождения максимума из произведений нужно знать не только максимальные, но и минимальные значения для арифметических операций над операндами с i-го по k-й и с $k + 1$-го по j-й соответственно. Значит, для любой последовательности операндов нужно хранить значение как максимально, так и минимально возможного результата выполнения арифметических операций над ними. Однако дополнительная матрица для хранения минимальных значений не нужна. Так как для $i > j$ матрица a не заполнена, мы можем отвести эти элементы для хранения минимумов, только запоминать результат решения этой подзадачи для операндов, начиная с i-го и заканчивая j-м, мы будем в элементе $a[j, i]$. Тогда в общем случае, если после k-го операнда стоит операция умножения ($i \le k < j$), то максимальный результат будет равен $max(a[i, k] \times a[k + 1, j], a[k, i] \times a[j, k + 1], a[i, k] \times a[j, k + 1], a[k, i] \times a[k + 1, j])$. Минимальный результат вычисляется аналогично. Оптимальное значение k нам не известно, но число k может принимать всего $i - j$ различных значений. Поскольку одно из них оптимально, то следует перебрать их все и выбрать наилучшее. Таким образом, любой элемент таблицы вычисляется за не более чем $4|i - j|$ операций, что соответствует асимптотической оценке $O(n)$. Единственным неудобством рассмотренного подхода является необходимость заполнять матрицу по линиям, параллельным главной диагонали: сначала вычисляются элементы, для которых $|i - j| = 1$, потом — $|i - j| = 2$ и т.д. Искомое значение арифметического выражения будет получено в элементе $a[1, n]$, который заполняется в последнюю очередь. Общее время работы алгоритма есть $O(n^3)$, тем самым он значительно эффективнее перебора всех возможных порядков выполнения арифметических операций. Объем памяти, необходимый для хранения таблицы, есть $O(n^2)$.

Приведем фрагмент программы для решения этой задачи, в котором и заполняется матрица a. Для хранения значений арифметических операций после каждого операнда используется одномерный массив `op`:

```
for m := 1 to n — 1 do {m = |i — j|}
  for i := 1 to n — m do
    begin
     j := i + m;
     r_max := —maxlongint — 1;
     r_min := maxlongint;
     for k := i to j — 1 do
       case op[k] of
       '+':begin {используем определенные выше
                  функции нахождения максимума
                  и минимума из двух элементов}
           r_max := max(r_max, a[i, k] + a[k + 1, j]);
           r_min := min(r_min, a[k, i] + a[j, k + 1])
           end;
           '*':begin
           r_max := max(r_max,
                    max(a[i, k] * a[k + 1, j],
                    max(a[k, i] * a[j, k + 1],
                    max(a[i, k] * a[j, k + 1],
                        a[k, i] * a[k + 1, j]))));
           r_min := min(r_min,
                    min(a[i, k] * a[k + 1, j],
                    min(a[k, i] * a[j, k + 1],
                    min(a[i, k] * a[j, k + 1],
                        a[k, i] * a[k + 1, j]))))
         end;
     end;{case}
    a[i, j] := r_max;
    a[j, i] := r_min
  end;
writeln(a[1, n]);
```

Если, помимо нахождения максимально возможного значения арифметического выражения, необходимо напечатать и сам оптимальный вариант расстановки скобок, то потребуется хранить еще и значения k, на которых достигается максимум или минимум. Попробуйте самостоятельно написать рекурсивный алгоритм, печатающий результат расстановки скобок требуемым образом.

Задача 4. Оптимальная триангуляция выпуклого многоугольника.

Требуется разрезать выпуклый n-угольник $n—2$ непересекающимися диагоналями на треугольники так, чтобы сумма длин диагоналей (стоимость разрезания) была минимальной.

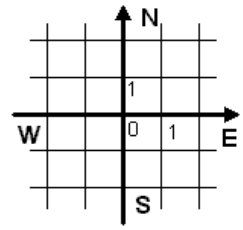
Решение. Несмотря на геометрический характер, эта задача легко сводится к предыдущей. Если мы проведем в таком многоугольнике одну диагональ, то стоимость разрезания, в котором эта диагональ обязательно присутствует, можно выразить через стоимость разрезания образовавшихся при ее проведении двух многоугольников. Более подробно о решении этой задачи можно прочитать в [3, 4, 6].

Волновые алгоритмы

Обычно под волновыми алгоритмами понимают некоторые способы поиска кратчайшего пути в графе. Рассмотрим их с точки зрения динамического программирования (ведь они относятся именно к этому методу решения задач), обобщив на более широкий класс проблем. Заметим, что при реализации волновых алгоритмов зачастую часть таблицы, отведенная для запоминания решений подзадач, остается незаполненной, а часть действий алгоритма является лишней, что отличает эти алгоритмы от решений задач, рассмотренных выше. Однако именно возможность использовать избыточную память и увеличить количество производимых операций, например в n раз, делает программы, реализующие эти алгоритмы, чрезвычайно простыми. Давайте в этом убедимся.

Задача 5. Робот. (*Командная олимпиада Санкт-Петербурга по программированию, 1999 г.*)

В исследовательской лаборатории фирмы *Robots&Co* разработали новую модель робота. Он работает по заранее заданной программе, в которой могут присутствовать команды: сделать шаг на Юг, на Север, на Восток или на Запад. Робот исполняет программу строго последовательно и, дойдя до конца программы, останавливается. Специалисты из *Robots&Co* заинтересовались вопросом, сколько существует различных программ, состоящих из K инструкций, таких, что робот, выйдя из начала координат, придет в точку с координатами (X, Y) $(0 \le K \le 16, \quad |X|, |Y| \le 16)$. Оси координат располагаются параллельно сторонам света, и единица измерения соответствует одному шагу робота. Напишите программу, которая дает ответ на этот вопрос.



Решение. Решать задачу будем волновым алгоритмом. А именно: сначала посчитаем, сколько существует программ, состоящих из одной инструкции, причем заканчивающихся во всех возможных точках (вернее, в тех, до которых успеет "докатиться волна" из начала координат за один шаг). Затем, опираясь на результаты первого шага, рассмотрим все программы, состоящие из двух инструкций, и т.д. Результаты будем запоминать в матрице a. Тогда, если после L шагов в каждом элементе $a[i, j]$ матрицы $(—16 \le i, j \le 16)$ будет храниться количество программ, состоящих из L инструкций и заканчивающихся в точке с координатами (i, j), то для каждой из точек (i, j) легко подсчитать количество программ, заканчивающихся в ней и состоящих из $L + 1$ инструкции, по формуле $a[i + 1, j] + a[i, j + 1] + a[i — 1, j] + a[i, j — 1]$. Теперь несложно написать программу для решения этой задачи. Для того чтобы избежать проверок на выход за

границу массива, как обычно, используем барьерный метод (увеличиваем размеры таблицы на единицу во всех направлениях):

```
var x, y, k : integer;
  a, b : array[−17..17, −17..17] of longint;
  i, p, q : integer;
begin
  read(k, x, y);
  fillchar(a,sizeof(a),0);
  b := a;
  a[0, 0] := 1;
  for i := 1 to k do
    begin
      for p := −i to i do
         for q := −i to i do
       {остальные точки заведомо недостижимы}
         b[p, q] := a[p + 1, q] + a[p, q + 1] +
                    a[p − 1, q] + a[p, q − 1];
      a := b
    end;
  writeln(a[x,y])
end.
```

Количество операций, которые выполняет данный алгоритм, есть $O(Kn^2)$, где n — размер просматриваемой на каждом шаге таблицы. Если, как в нашем случае, K не превосходит n, то количество операций растет как $O(n^3)$, хотя для данной задачи возможно построить алгоритм с вычислительной сложностью $O(n^2)$. Такой алгоритм для решения подобных задач будет рассмотрен в одной из следующих лекций. Вообще говоря, с точки зрения предложенного решения K может быть почти как угодно большим. Для динамической схемы в данном случае важно лишь то, что робот может перемещаться в ограниченной области, для описания которой мы в состоянии завести соответствующую таблицу. Хотя часть ее так и останется незаполненной (в нашей таблице всегда найдутся клетки, до которых робот по условию дойти за K шагов не сможет), но в просмотрах на каждом шаге алгоритма такие клетки участвуют, несмотря на то что мы попытались ограничить для каждого шага просматриваемую часть таблицы. Дальнейшая же оптимизация алгоритма приведет к потере прозрачности решения.

Задача 6. Лабиринт.

Класс задач о лабиринтах достаточно широк, и возможны различные формулировки подобных задач. Однако их решения мало отличаются друг от друга. Поэтому рассмотрим задачу о лабиринте в следующей формулировке. (*Московская олимпиада по программированию, 1983 г.*)

Лабиринт задан массивом a размером $n \times n$, в котором $a[i, j] = 1$, если клетка "проходима", и $a[i, j] = 0$ — если "непроходима". Путник изначально размещается в "проходимой" клетке $[i0, j0]$. Он может перемещаться в любую из соседних "проходимых клеток", если у нее есть общая сторона с той, в которой он находится. Определить, может ли путник выйти из лабиринта. Если может, то напечатать путь от выхода до начального положения путника. Выходом считается любая граничная точка массива a.

Решение. Запишем в клетку $[i0, j0]$ число 2. Просмотрим все клетки лабиринта (или исключим из рассмотрения заведомо недостижимые за один ход). Если у рассматриваемой "проходимой" клетки, помеченной единицей, есть сосед, помеченный двойкой (в общем случае числом k), то мы запишем в нее 3 (в общем случае $k + 1$). Таким образом, на каждом шаге алгоритма числом k будут помечены все те клетки, до которых путник может добраться ровно за k — 2 единичных перемещения (до которых за k — 2 шага "докатилась волна"). Этот процесс закончится, когда очередное число будет вписано в граничную клетку либо когда за весь просмотр массива ни одна из клеток не будет помечена (выхода в этом случае нет). Печать решения также произвести несложно. Если последняя клетка помечена числом k, то предпоследней в пути может быть любая соседняя с ней клетка, помеченная числом k — 1, и т.д. Полученный путь по построению является кратчайшим. Заметим, что для решения этой задачи дополнительные массивы можно не заводить, однако просмотр ряда клеток массива, как и в предыдущей задаче, будет избыточным. Приведем основную часть программы для решения этой задачи, в которой, как и ранее, используется нулевой "барьер":

```
k = 2; a[i0, j0] := k;
flag := true;{были помечены новые клетки}
while flag do
begin
  flag := false;
  for i := i0 − (k − 1) to i0 + k − 1 do
    for j := j0 − (k − 1) to j0 + k − 1 do
      if (a[i, j] = 1) and ((a[i − 1, j] = k) or
          (a[i + 1, j] = k) or (a[i, j − 1] = k) or
          (a[i, j + 1] = k)) then
      begin
      a[i, j] := k + 1; flag := true;
       if (i in [1, n]) or (j in [1,n]) then goto 1
      end;
  k := k + 1
end;
1 : if flag then {выход найден}
    repeat {печатаем путь}
      writeln(i,' ',j);
      if a[i − 1, j] = k then i := i − 1 else
      if a[i, j − 1] = k then j := j − 1 else
      if a[i + 1, j] = k then i := i + 1 else
         j := j + 1;
      k := k − 1
    until k = 0;{[i0, j0] напечатана}
```

В данной программе использование оператора `goto` иллюстрирует единственный случай его допустимого применения в структурированной программе. А именно: необходимость досрочного выхода сразу из нескольких вложенных циклов (в нашем примере — трех).

Для того чтобы исключить лишние просмотры клеток массива, можно завести список (динамический или

организованный с помощью массива) и хранить в нем координаты только тех клеток, рассмотрение которых следует произвести на очередном шаге.

Задача 7. В некоторой игре одно двузначное число можно заменить на другое по следующему правилу: любая из двух цифр исходного числа заменяется на сумму или разность его цифр (в случае разности из большей цифры вычитается меньшая). Для двузначных чисел a и b построить последовательность чисел минимальной длины, начинающуюся с числа a, заканчивающуюся b, а каждое следующее число в цепочке можно получить из предыдущего по указанному выше правилу или указать, что это сделать невозможно. Например, для чисел 12 и 31 последовательность будет выглядеть так: 12 32 31.

Решение. Отнести решение этой задачи к методу динамического программирования позволяет следующее свойство. Если допустимая последовательность минимальной длины построена, то любая ее часть решает аналогичную задачу для начального и конечного числа в выделенной подпоследовательности. Для реализации решения следует завести одномерный массив (таблицу), состоящий из 99 элементов, каждый элемент массива соответствует одному из двузначных чисел (здесь под двузначными понимаются все натуральные числа, не превосходящие 99). Первоначально массив следует обнулить, а в элемент массива с индексом a занести 1. Далее, получив из числа a все возможные вторые элементы последовательности, следует занести в соответствующие элементы массива двойки. Произвольный же (i-й) шаг алгоритма будет выглядеть так. Просматриваем массив, и если какой-либо его элемент равен i—1, то из индекса этого элемента получаем все допустимые варианты преобразованных чисел и записываем в соответствующие им элементы массива число i, но только если ранее значение элемента было равно 0. Алгоритм заканчивается, если будет заполнен элемент с индексом b или на i-м шаге ни один ранее нулевой элемент массива не будет заполнен числом i (решение в таком случае отсутствует). Если решение существует, то восстановить его по описанному массиву несложно. Динамическое программирование в данном случае удалось применить лишь потому, что количество различных чисел, которые могут возникнуть по ходу игры, ограниченно (их всего 100). Размер таблицы поэтому очевиден, однако на каждом шаге большая часть элементов, в данном случае одномерного массива, будет просматриваться зря. Изменить это можно так же, как и в предыдущей задаче.

Задача 8. Арифметическим показателем натурального числа M по цифре N (записывается $Ar(M, N)$) назовем минимальное количество цифр N, которые необходимо использовать в некотором арифметическом выражении, содержащем только цифры N, знаки операций $+$, $-$, $*$, $/$ и скобки, дающем результат M. Например, $Ar(17, 3) = 5$, а соответствующее выражение может быть, например, следующим: $3 \cdot (3 + 3) - 3/3$, а $Ar(24, 1) = 5$, поскольку существует выражение $(11 + 1) \cdot (1 + 1) = 24$. Вычислите арифметический показатель для заданной пары чисел. Результат любой промежуточной операции должен быть натуральным и **меньше 1000**. (*Московская командная олимпиада по программированию, 1999 г.*)

Решение. Как и в предыдущей задаче, применение динамического программирования возможно благодаря ограничению не только на результат, но и на промежуточные операции. Более того, решение данной задачи вообще не отличалось бы от предыдущей (за исключением правила, по которому формируются клетки для заполнения на очередном шаге алгоритма, в данном случае k-й шаг соответствует k цифрам в арифметическом выражении), если бы цифры не разрешалось объединять в многозначные числа. При имеющихся ограничениях цифры могут образовывать лишь двузначные и трехзначные числа. Поэтому, для того чтобы получить все допустимые числа из k цифр, следует рассмотреть все элементы вспомогательного массива (размерностью 1000), помеченные числом $k - 1$, и по очереди произвести все арифметические операции между их индексами и заданной в условии цифрой (для вычитания и деления в ряде случаев следует рассмотреть оба варианта взаимного расположения операндов). Потом рассмотреть все элементы, помеченные числом $k - 2$, и произвести все арифметические операции между их индексами и двузначным числом, образованным при слиянии двух заданных цифр. Наконец, следует рассмотреть элементы, помеченные числом $k - 3$, и произвести арифметические операции между их индексами и трехзначным числом, образованным при слиянии трех заданных цифр. Если при этом в результате арифметической операции получается натуральное число, меньшее 1000, и соответствующий элемент вспомогательного массива равен 0, то он помечается числом k. Таким образом, данный алгоритм также относится к "волновым".

Рассмотрение задач динамического программирования будет завершено в следующей лекции.

Литература

1. *Андреева Е.В.* Еще раз о задачах на полный перебор вариантов. "Информатика" № 45/2000.

2. *Беллман Р.* Динамическое программирование. М.: ИЛ, 1960.

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

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

5. *Гордеев Э.Н.* Задачи выбора и их решение. В кн.: Компьютер и задачи выбора. М.: Наука, 1989.

6. *Окулов С.М.* 100 задач по информатике. Киров: Изд-во ВГПУ, 2000.

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

Введение в профессию "учитель информатики"

Лекции читает А.Г. Гейн

Вниманию читателя предлагаются несколько лекций, читавшихся учителям на тех или иных курсах повышения квалификации. Они, конечно, подверглись определенной обработке. Кроме того, после каждой лекции приведены вопросы и задания, над которыми, на мой взгляд, интересно подумать каждому заинтересованному педагогу. Авторские ответы на них в той или иной степени будут обсуждаться в последующих лекциях.

Лекция 1. Задачи курса информатики в общеобразовательной школе

— Ты кто? — спросила Кошка, которая гуляла сама по себе.

— Я — Коза рогатая, иду за малыми ребятами.

— А зачем? — спросила Кошка. Коза задумалась и поняла, что всю свою жизнь она прожила зря.

Учебный предмет, появившийся чуть более 15 лет назад в школьном расписании, не только прочно вошел в жизнь сотен тысяч учащихся, но и проявляет весьма настойчивый характер. Словно росток цветка, он упорно пробивает не всегда благодатную почву, набирает силу и захватывает все большую территорию, оттесняя своих более старших собратьев. Сегодня базовый курс информатики из классов, завершающих школьное образование, нередко оказывается переместившимся в 7—9-е классы. Но и в старших классах информатика не исчезает, а становится профильно-ориентированным курсом. Многочисленные энтузиасты ведут информатику в начальной школе. Наконец, под разными названиями появились разработки сквозного курса информатики, охватывающего все 10 лет школьного образования.

Указанное разнообразие существующих и проектируемых курсов информатики не имеет аналога ни в каком другом школьном предмете. Убедиться в этом можно, заглянув, например, в сборник программ по информатике [7], где приведены 3 программы базового курса для 7—9-х классов и 10 программ для 10—11-х классов. Реально преподаваемая информатика еще более многолика. И дело здесь не столько в техническом "зоопарке" — от УКНЦ до пентиумов и макинтошей — или установке нынешнего Министерства образования на вариативность, сколько в характере самой научной дисциплины, отражением которой и стал соответствующий школьный предмет.

И поэтому вопрос "Зачем изучать информатику в школе?" не то чтобы стоит острее, чем аналогичный вопрос для других предметов, но ответ на него во многом определяет, чему учить школьников в данном курсе. Отмечу, кстати, что в своей практике я неоднократно слышал от учащихся вопросы: "Зачем мне учить математику (физику, химию, литературу и т.д.)? Ведь я не буду тем-то и тем-то", — но никогда не слышал этот вопрос по отношению к информатике. И это вселяет надежду, что декларируемые мной целевые установки находят позитивный отклик у учеников. Впрочем, надо понимать, что декларируемое учащимся далеко не всегда совпадает с тем, что имеет место на самом деле. И происходит это вовсе не по злому умыслу. А почему — мы поговорим в следующей лекции.

Вот и сейчас мы будем обсуждать цели и задачи курса информатики не с точки зрения "низов", т.е. учащихся, а с позиций "верхов", или, как говорят, с точки зрения социального заказа к системе образования.

Когда мы говорим слова "социальный заказ", то под этим вовсе не понимаются волюнтаристские решения руководства системой образования. Социальный заказ определяется образом жизни самой страны. И он меняется с изменением жизни в этой стране. Скажем, долгое время социальный заказ системе образования на фундаментальное обучение математике, физике, химии и т.п. определялся востребованностью специалистов военно-промышленным комплексом (ВПК), который доминировал в нашей стране. Изменения в жизни нашей страны, в частности, уменьшение (иногда на грани риска) ВПК и увеличение роли гуманитарно-экономической составляющей привели и к уменьшению школьных часов на математику и естественно-научные дисциплины, и к "размножению" в рамках школы (опять-таки, не всегда оправданному) гуманитарных и экономических предметов. Информатика же как наука и как школьный предмет имеет в нашей стране совсем особую историю. О становлении науки "Информатика" в России вы можете узнать из замечательной книги [6]; история школьной информатики еще не написана.

Социальный заказ на школьную информатику, а следовательно, и целевые установки этой дисциплины менялись за прошедшие 15 лет неоднократно. Знание истории позволяет говорить о будущем (здесь я позволил себе перефразировать извест-

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

ное высказывание: "Страна, не знающая своей истории, не имеет будущего"). Так вот, начнем с истории.

В марте 1985 года появилось совместное постановление ЦК КПСС и Совета Министров СССР о введении в общеобразовательной школе курса "Основы информатики и вычислительной техники". Ввести этот курс указанным постановлением требовалось с сентября того же года. А вот компьютеров в школе к сентябрю не обещали — внедрялся так называемый "безмашинный" курс информатики.

Тем не менее **основной целью** вводимого курса объявлялось обеспечение компьютерной грамотности всех выпускников общеобразовательной школы. Компьютерная грамотность при этом понималась как умение эффективно использовать вычислительную технику во всех областях деятельности человека. Структура и содержание компьютерной грамотности школьников расшифровывались как "подготовка пользователя, знающего возможности и сферу применения ЭВМ, овладевшего основными принципами программирования и имеющего навыки общения с компьютером" (см. [2], с. 7). Явный диссонанс между декларацией о приобретении навыков общения с компьютером и отсутствием таковых в школе не сильно смущал руководство системой образования. Во-первых, Минрадиопром заверял, что через пару лет оснастит все школы компьютерами (типа БК-0010, без дисководов, принтеров и тому подобных "излишеств") и школьная информатика станет компьютерной. А во-вторых... Бескомпьютерный вариант вполне отвечал социальному заказу того времени. По существовавшим в то время прогнозам стране предстояло в несколько раз увеличить штаты программистов, которые обеспечивали бы работу вычислительной техники в военных и гражданских отраслях народного хозяйства. Разумеется, такую подготовку должны были бы осуществлять все равно вузы и техникумы. Но начинать с нуля — слишком долго и дорого, гораздо эффективнее начальную подготовку перенести в общеобразовательную школу, даже если потом это и не будет востребовано большей частью выпускников. И для решения такой задачи в школе вовсе не требуются высокопроизводительные компьютеры с хорошей периферией, графикой и т.д.

Бескомпьютерный вариант был привычен и удобен школе. И не только потому, что вычислительная техника требовала значительных финансовых средств для своего приобретения и поддержания в рабочем порядке. Прежде всего не изменялся характер педагогического процесса. Как и во всех других предметах, учитель оставался единственной инстанцией, решавшей, правильно или неправильно выполнено задание учеником. Преподавание информатики в машинном варианте — это совсем иная педагогическая обстановка. Окончательный вердикт алгоритму о его правильности выносит не учитель, чье знание учеником должно восприниматься как истина в последней инстанции, а какая-то машинюшка, которую почему-то никак не удается заставить исполнить то, что написано в задании. Эту стрессовую ситуацию выдержали далеко не все учителя, пришедшие в школьную информатику в первые дни ее существования. И примерно два года спустя — по мере появления в школе компьютерной техники — им на смену преподавать информатику приходят профессиональные программисты. Они начинают учить школьников какому-нибудь языку программирования и относиться к компьютеру как большому и чуть более "умному" калькулятору. Впрочем, компьютеры, имевшиеся в то время в школе, и программное обеспечение к ним вполне оправдывали такое к себе отношение.

Целевые установки представляются в этот момент выполненными на все 100, а информатика фактически отождествляется с программированием. В статье [4] Г.В. Лебедев, соавтор А.Г. Кушниренко по одному из наиболее распространенных в то время учебников информатики, прямо ставит знак равенства между информатикой и программированием. В другой статье (см. [1]), излагающей целевые установки того же авторского коллектива, говорится: "Для нас важно то, что существует специфический стиль мышления, который принято называть алгоритмическим. Развитие этого стиля мышления и есть *основная цель* курса информатики" (курсив мой. — **А. Г.**). Здесь, как мы видим, речь идет уже не только об обучении программированию, но и о формировании определенного стиля мышления. Это претензии на гораздо большее, чем просто обучение программированию. Человек с алгоритмическим стилем мышления сам уподобляется программируемому устройству. Этакий винтик общества, который вполне алгоритмизуем и потому хорошо управляем. Весьма заманчивая перспектива для руководства обществом. Но наша страна делает крутой поворот.

Производство, торговля и другие сферы деятельности перестают быть государственной монополией. Соответственно уже не государство является единственным заказчиком и потребителем специалистов в области использования вычислительной техники. И самое главное, что применение этой техники в многочисленных возникающих фирмах и фирмочках ограничивается однотипными задачами, легко решаемыми с помощью развитых к этому времени информационных технологий. Эти фирмы и фирмочки вовсе не намерены ждать, пока появятся отечественные персональные компьютеры, способные обеспечить необходимый уровень информационного обслуживания. В страну хлынули компьютеры зарубежного производства, а вместе с ними и программные продукты для этих компьютеров — ведь компьютер без программного обеспечения не более чем груда железа и пластмассы.

Поэтому в начале 90-х годов в массовом сознании происходит смещение центра тяжести в понимании того, что есть информатика, от алгоритмизации в сторону так называемых "информационных технологий". Речь идет именно о массовом сознании — широкое офисное внедрение персональных компьютеров сорвало покров таинственности и элитарности с вычислительной техники. Каждый родитель обрел уверенность: "Вот то, что нужно знать моему ребенку, чтобы прожить в современном мире". И школа (как и другие

сферы образования, в том числе негосударственного) откликнулась на этот социальный заказ. Появились и адепты нового курса, где вся информатика сводится к изучению офисного программного обеспечения, созданного фирмой Microsoft.

Иногда можно услышать мнение, что о компьютерных технологиях обработки информации мы узнали только в связи с экспансией в нашу страну западных производителей компьютерной техники. Вообще говоря, это не так. Такие технологии были известны и применялись в нашей стране с тех же времен, что и на западе. Другое дело, что они не были доступны широким массам. Но это совсем другой поворот темы. На мой взгляд, политический. Те, кому сейчас около пятидесяти и более, еще помнят времена, когда пишущие машинки стояли на учете в компетентных органах — как-никак множительная техника! О каком массовом владении компьютерными информационными технологиями могла идти речь в таком обществе? Это сейчас любой текст можно набрать и распечатать на принтере в любом количестве экземпляров. Это сейчас можно записать информацию с одного компьютера и перенести на другой. Это сейчас можно выйти в Интернет или отправить электронное письмо хоть в процветающие США, хоть в самую что ни на есть Тмутаракань. Величайшее политическое искусство отцов-основателей школьной информатики состояло в том, что им удалось убедить руководство страны смотреть на компьютер как на средство программирования, а не информационное средство. Главные учебные цели — воспитание алгоритмического мышления, главное понятие — формальный исполнитель, не задумывающийся о сути и цели исполняемого алгоритма. Это ли не идеал руководства нашей страной в то время, желающего видеть руководимое общество комплектом винтиков-исполнителей? Ведь не удалось же ввести в школу курс кибернетики даже после того, как она перестала именоваться "продажной девкой капитализма". Была разработана программа [5] (в ряде пунктов весьма близкая к некоторым современным программам по информатике), были экспериментальные курсы (см. [3]), но все кончилось ничем. Оно и понятно: разве можно всех школьников учить основам управленческой науки — они же тогда начнут задумываться, правильно ли управляется страна, область, город, район...

Примечательно, как отреагировало Министерство образования на изменение ситуации. В принятом в то время новом базисном учебном плане (который, кстати, действует и поныне) записано, что курс информатики разрешается преподавать только в том случае, если для этого в школе есть условия (т.е. класс компьютерной техники, позволяющий осваивать информационные технологии). При этом органы народного образования снимают с себя какие-либо обязательства по поставке компьютерной техники в школы и финансированию обслуживания той, которая к этому моменту в школах имеется. Казалось бы, информатика в этих условиях должна была бы исчезнуть из школьного расписания. Но этого не происходит. Социальный заказ оказался настолько сильно выраженным, что информатика в школах осталась, пусть во многих из них в форме изучения информационных технологий.

Обучение чисто технологии — это подготовка рабочего. С точки зрения образования нет разницы между токарем, каждодневно вытачивающим одну и ту же деталь, и человеком, ежедневно готовящим на компьютере одни и те же отчетные формы или заполняющим одну и ту же базу данных наименованиями поступивших в магазин товаров. И постепенно рынок труда в сфере примитивной информационной деятельности заполняется, хотя потребность в рабочих информационной сферы сегодня все еще намного выше, чем потребность в токарях.

Мы стоим на пороге очередного изменения социального заказа в области информатики. Не пытаясь предугадать, каким он будет, обсудим, что может дать информатика образованию школьников в случае благоприятного для нее развития ситуации.

Прежде всего курс информатики позволяет учащимся освоить одно из фундаментальных понятий, активно проявивших себя в XX веке, — понятие **информации**. Принципиально важно, чтобы учащиеся осознали, что решение любой задачи, возникающей в деятельности человека, начинается с решения этой задачи на информационном уровне. Такое решение обязательно проходит три этапа:

— на первом этапе определяется, какую исходную информацию необходимо иметь для решения задачи и как ее получить;

— на втором этапе привлекаются знания, позволяющие связать исходную информацию с результатами и определяющие, как именно получить требуемый результат;

— на третьем этапе исходная информация извлекается и преобразуется в результирующую информацию.

На каждом из этих этапов нужно оценивать качество информации: ее полезность, достоверность, существенность и т.д. Все это приучает учащегося подходить к информации осмысленно, а не просто запоминать ее или бездумно ей следовать.

Не менее важным компонентом является понимание того, какие и как использовать для указанной выше информационной деятельности существующие компьютерные информационные технологии, в том числе Интернет.

В своей информационной деятельности человек должен уметь защищать необходимую ему информацию и уметь защищаться от информационной агрессии других. К сожалению, мир наш еще далеко не совершенен.

Указанный комплекс знаний и умений можно назвать **информационной культурой** человека. Овладение информационной культурой и следует считать **основной целью** современного общеобразовательного курса информатики.

Но перед курсом информатики стоят и другие задачи.

Одной из них является осознание учащимися единства информационных процессов, протекающих в самых разнообразных системах — живых, технических,

социальных. Такое единство позволяет говорить о законах, которым подчиняются все эти процессы. А знание объективных законов позволяет избежать многих ошибок в деятельности человека, в данном случае в его информационной деятельности.

Наконец, отметим, что информация — это основа для грамотной организации процессов управления. При демонополизации производства понимание основ управления становится необходимым практически каждому взрослому жителю страны. Это вовсе не значит, что каждая кухарка должна управлять государством, но понимать суть управленческих решений хотя бы в рамках той фирмы, где человек является акционером, необходимо.

А как же алгоритмизация? Неужели она должна покинуть информатику? Вовсе нет. Алгоритмическое управление техническими (в том числе компьютерными) системами — важный раздел человеческого знания. Но, кроме того, сам человек весьма часто в своей деятельности руководствуется различными алгоритмами. И понимание сути своей собственной алгоритмической деятельности весьма важно для каждого человека. Поэтому в вопросе о месте алгоритмизации в курсе информатики мы полностью присоединяемся к тезису, высказанному в "Концепции содержания образовательной области "Информатика и информационные технологии" в двенадцатилетней школе" (см.: "Информатика" № 15/2001), что одной из основных тенденций развития указанной образовательной области является "отказ от обязательного освоения школьниками сред и языков профессионального программирования как составной части общеобразовательной подготовки школьников". Тем не менее изучение алгоритмов, средств их описания и методов построения остается обязательным пунктом раздела "Теоретическая информатика", который предусмотрен в курсе информатики данной концепцией.

К сожалению, трудно указать литературу для учителя, где вопрос о целях и задачах курса информатики рассматривался бы достаточно объективно, а не с точки зрения вкусовых пристрастий автора. Поэтому приведенный ниже список литературы — это лишь указание на источники цитат, использовавшихся в данной публикации.

Вопросы и задания

1. Каковы причины изменения целей изучения курса информатики в общеобразовательной школе? Ниже указаны некоторые из них. Проранжируйте их по степени значимости (1 — наиболее значимая, 2 — следующая по уровню значимости и т.д.):
- расширение возможностей компьютерной техники — _____;
- политические изменения в обществе — _____;
- рост числа работников, занятых в сфере применения компьютерных технологий, — _____;
- стремление идти в ногу со странами, имеющими развитое информационное общество, — _____;
- необходимость включения в международное информационное сотрудничество — _____;
- иная причина (укажите, какая, на ваш взгляд) _____ — _____.

2. 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. Однако при объяснении учащимся того, зачем они будут изучать информатику, весьма редко звучат те цели и задачи, которые сформулированы в данной лекции. Правильно ли это? Постарайтесь аргументировать свою точку зрения.

3. В учебниках созданию мотивации к изучению курса обычно отводится хотя бы несколько абзацев предисловия. Ниже приведены соответствующие фрагменты из предисловий нескольких реально применяемых учебников (мы специально не называем авторов — это не имеет значения для целей данного задания, но в скобках указываем, для какого класса предназначается цитируемый учебник). Оцените, в какой мере в каждом из этих фрагментов представлены цели и задачи изучения курса информатики. Можете ли вы предложить свой вариант введения, который бы более адекватно отражал эти цели и задачи?

а) "Мы начинаем изучать новый учебный предмет — "Основы информатики". **Информатика** — наука о методах и средствах представления, преобразования, накопления, передачи и использования информации. Как самостоятельная научная дисциплина информатика выделилась недавно — во второй половине XX века, однако многие ее задачи решались людьми на интуитивном уровне с незапамятных времен. Социальное и культурное развитие человечества было бы невозможно без накопления, обобщения и распространения информации. В современном мире роль информатики, средств обработки, передачи и накопления информации неизмеримо возросла. Средства информатики и вычислительной техники сейчас во многом определяют научно-технический потенциал страны, уровень развития ее народного хозяйства, образ жизни и деятельности человека" (8-й класс).

б) "Мир, окружающий каждого из нас, удивительно разнообразен. Природа и творения человеческих рук — куда ни взгляни — предъявляют нам вещественность объектов и энергетику процессов. Эти два понятия — **вещество** и **энергия** — уже несколько столетий освоены человечеством и в различных своих проявлениях служат объектом изучения таких наук, как физика, химия, биология, геология, и вообще, можно сказать, любой естественной науки.

Но XX век выдвинул на передний край еще одно понятие — понятие **информации**. Наука, изучающая свойства информации, законо-

мерности информационных процессов, называется **информатикой**. А теперь информатика появилась у вас и как учебная дисциплина. Вы будете ее изучать, а наш учебник, как мы надеемся, поможет вам в этом" (10-й класс).

в) "Мы начинаем изучать новый предмет — информатику. ***Информатика*** изучает методы представления, накопления, передачи и обработки информации с помощью электронно-вычислительных машин (ЭВМ). Что же такое информация, что такое ЭВМ и как ЭВМ обрабатывает информацию?" (полный текст предисловия к одному из учебников 10-го класса).

г) "О чудесных возможностях компьютеров наслышаны все. Да и сами они уже давно не в диковинку. Конечно, вам хочется познакомиться с ними поближе. И в первую очередь научиться применять компьютер для решения разнообразных задач" (7-й класс).

д) "Вы начинаете знакомство с новым для вас предметом — это о том, как работать на компьютере. Да, конечно, и об этом тоже, но только на первый взгляд, если не пытаться разобраться, зачем надо уметь работать на компьютере и почему компьютер стал предметом чуть ли не первой необходимости для ваших родителей, старших братьев и сестер, многих знакомых вам людей. На самом деле дисциплина эта настолько многогранна, что четко определить ее границы практически невозможно.

Информатика — область человеческой деятельности, связанной с процессами хранения, преобразования и передачи информации.

Ваша задача в процессе изучения информатики — не только научиться пользоваться компьютером, но и осознать его возможности, определить роль, которую он играет в нашей жизни, понять, какую помощь он может оказать вам при исследовании окружающего мира" (6-й класс).

4. В лекции цели и задачи школьного курса информатики анализировались исключительно с позиций социального заказа, т.е. как бы извне системы образования. Известно, однако, что на использование компьютерных информационных технологий в самом процессе обучения возлагаются большие надежды.

а) Ниже приведены варианты использования компьютерных технологий в обучении. Проранжируйте их по степени значимости (1 — наиболее значимая, 2 — следующая по уровню значимости и т.д.):

- организация компьютерного тестирования — ___;
- моделирование процессов, которые нельзя осуществить в реальных условиях класса,— _____;
- оперативный доступ к значительным объемам справочной информации (электронные энциклопедии и т.п.) —_____;
- использование сетевых ресурсов для организации работы в коллективных проектах — _____;
- доступ в Интернет — _____;
- иной вариант (укажите, какой, на ваш взгляд) _____ — ____.

б) Для каждого из перечисленных в а) вариантов укажите, какой раздел информатики и с какого класса должен изучаться учащимися, чтобы этот вариант мог использоваться.

Литература

1. *Зайдельман Я.Н., Лебедев Г.В., Самовольнова Л.Е.* Три кита школьной информатики. ИнфО № 3, 1993. С. 19—23.

2. Изучение основ информатики и вычислительной техники: Методическое пособие для учителей и преподавателей средних учебных заведений. Ч. 1 / Ершов А.П., Монахов В.М., Кузнецов А.А. и др. М.: Просвещение, 1985, 191 с.

3. *Кузнецов А.А.* Изучение факультативного курса "Основы кибернетики" // Сб. "Факультативные занятия в школе". М.: Педагогика, 1978.

4. *Лебедев Г.В.* О новом учебнике информатики // Информатика и образование № 5, 1990. С. 24—30.

5. *Леднев В.С., Кузнецов А.А.* Программа факультативного курса "Основы кибернетики" // Математика в школе № 1, 1975.

6. *Фет Я.И.* Очерки истории информатики в России. Н.: ОИГМ РАН, 1998.

7. Программы для общеобразовательных учреждений. Информатика. М.: Просвещение, 1998, 143 с.

Калейдоскоп

В поисках пиратов

Одна из английских организаций, борющихся с пиратским распространением компьютерных программ, предприняла компанию по запугиванию владельцев нелегальной продукции. На оживленной площади города Глазго она поставила автофургон с огромной антенной на крыше. На вопросы прохожих сотрудники организации пояснили, что это новейший специальный локатор, который, разъезжая по городам, будет выявлять компьютеры с пиратскими программами. Стоит отметить, что некоторые обладатели действительно испугались, так как это отразилось на продажах лицензионной продукции.

По материалам журнала "Наука и жизнь"

Программист программисту

Калифорнийский суд снял все обвинения с молодого программиста, которому удалось взломать систему защиты дисков DVD и который поделился полученными сведениями с сетевой общественностью. Эндрю Баннер сумел отстоять свои интересы и после первого обвинительного приговора подал апелляцию в вышестоящую судебную инстанцию, которая была удовлетворена, а предыдущее решение отменено.

В качестве истца выступала DVD Copy Control Association, усомнившаяся в праве программиста делиться своими наблюдениями с единомышленниками, однако американская судебная система доказала, что свобода слова в США еще существует и каждый человек вправе делиться своими сведениями со всеми, с кем захочет.

По материалам сайта www.ananova.com

Материалы Роботландского университета

Тропинка конструктора

А.А. Дуванов,
Ю.А. Первин

Продолжение. См. № 4/2002

Книга для ученика

ГЛАВА 2. МУДРЫЙ КРОТ

1. Знакомство с Кротом

Сидевший, как всегда, у компьютера дедушка Фёрстов пообещал детям освободиться через несколько минут. (“Надо закончить срочное письмо”, — объяснил он и попросил внуков посидеть в своем кабинете на диване, подождать немного.) Дина и Тим забрались с ногами на мягкий диван и вполголоса принялись обсуждать роботландский “Монитор”.

— На занятиях про “Монитор” мы с дедушкой несколько раз перетаскивали со страницы на страницу пиктограмму *Мудрого Крота*, да и на странице перемещали его туда-сюда. А мне так хотелось остановить мышку и войти в эту программу, — сказал Тим.

— Да и я тоже люблю *Мудрого Крота*, — понимающе ответила Дина, — у него такие интересные задачи.

Дедушка оторвал взгляд от экрана и спросил ребят:

— О чем вы там беседуете?

— О *Мудром Кроте*. Нам очень хочется в него поиграть. Ты ведь знаешь *Мудрого Крота*, дедушка? — ответила Дина.

Конечно, дедушка Фёрстов хорошо знал эту программу. Ему даже доставляло удовольствие иной раз погонять *Крота* по лабиринтам (“Чтобы встряхнуться”, — объяснял он сам себе, когда, устав от долгих размышлений у экрана, он щелкал по пиктограмме с маленьким кротом в лабиринте). Но он не стал признаваться в своей слабости, а решил воспользоваться вопросом Дины, чтобы заставить детей поговорить о программе.

— Расскажите-ка мне вкратце, кого называют *Мудрым Кротом* в этой программе, почему он мудрый и что представляют собой его задачи.

А дети словно ждали этого приглашения и принялись наперебой рассказывать о любимой программе.

— *Мудрый Крот* живет под землей, где он прорыл себе многочисленные ходы, превратившие его жилище в настоящий лабиринт. *Крот* очень трудолюбив и запаслив. А еще он любит порядок. Все собранные им мешки с зерном лежат на складе ровно и аккуратно. Но вот однажды, когда *Крот* был в гостях у знакомой жабы (“Это Дина вспомнила сказку про Дюймовочку”, — по ходу рассказа сестры прокомментировал Тим), злоумышленники утащили мешки со склада и шутки ради разбросали по всему лабиринту. Что делать? Приходится собирать мешки снова. А лабиринт большой и запутанный, с узкими проходами, как тут не быть мудрым? И еще беда: *Кроту* хоть и нравится побегать по коридорам лабиринта, но он уже стал стар и слабоват. Он не может тащить мешок, а способен только толкать его перед собой. К тому же *Крот* — не слон: силенок у него маловато, даже два мешка ему не сдвинуть, а три — тем более.

Наша задача — помочь *Кроту* собрать все мешки с зерном на складе (в одной или нескольких комнатах лабиринта, где место для каждого мешка помечено светлой точкой).

— Да, красивую сказку вы придумали. На самом деле, я думаю, мешки собирает не *Крот*, а человек, управляя исполнителем на экране при помощи стрелок клавиатуры.

Тим вмешался, словно не расслышав дедушкино замечание:

— Дина не сказала про еще одну неприятность, которая может случиться с нашим *Кротом*. Загнав мешок в угол, он делает свою задачу неразрешимой: такой мешок навсегда останется в углу и на кротовый

склад уже не попадет — задачу приходится решать заново.

Дина согласно кивнула, но уточнила:

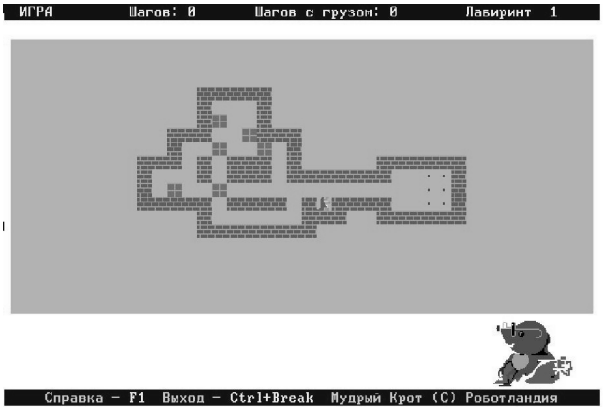
— Ну, не совсем заново, ведь можно воспользоваться откаткой.

Дедушка, по-прежнему разыгрывая изумление, предложил:

— Так, может быть, Тим, ты вызовешь эту программу на экран нашего компьютера? И вы мне покажете этого самого *Крота*.

Тима об этом можно было и не просить:

— С удовольствием! Перехожу в "Мониторе" на страницу *Мудрый Крот* и нажимаю там пиктограмму запуска:



Дина продолжила свои объяснения:

— На экране этой программы есть две служебные полоски и рабочее поле, на котором изображен лабиринт с мешками, складом, в котором можно увидеть полочки-точки, и самим *Кротом*, ожидающим наших команд.

А Тим добавил:

— В нижней полоске написано, как вызывать подсказку (клавиша F1) и как выходить из программы (аккорд Ctrl + Break). Там же расположены названия программы и фирмы, которая эту программу сделала.

Тут дедушка не выдержал и тоже включился в обсуждение программы о кроте:

— Верхняя полоска — тоже информационная. В ней показаны текущий режим работы (сейчас он установлен в положение *Игра*), номер лабиринта (сейчас на экране лабиринт с номером 1) и табло, описывающее перемещения исполнителя: в нем располагаются счетчики шагов *Крота* по лабиринту. *Крот* не сделал еще ни одного шага, поэтому сейчас счетчики показывают нули.

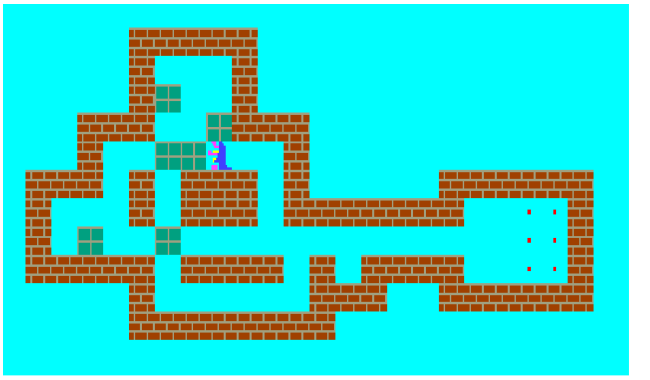
Такие познания дедушки в программе, о которой он с интересом только что просил рассказать, могли бы заставить ребят заподозрить какой-то подвох, но они были настолько увлечены предвкушением игры, что дедушкино дополнение было воспринято спокойно, без эмоций и подозрений. Только Тим успел предупредить, прежде чем приступить к работе с программой:

— В этой программе мышка не участвует. Придется действовать только клавишными стрелками.

2. Игра

Тим сказал:

— Первый лабиринт — очень простой, я запросто решаю эту задачу. Нужно только догадаться, как правильно переместить на склад первый мешок. Вот смотри, я делаю первое перемещение мешка:

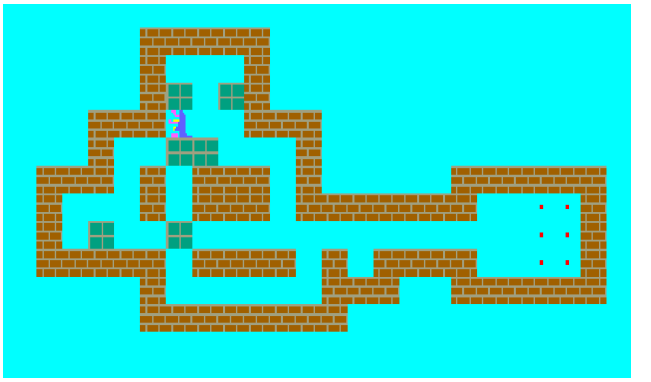


— А тебя не пугает, что два мешка оказались рядом? Сможет ли *Крот* потом их толкнуть?

Но Тима такая трудность не испугала:

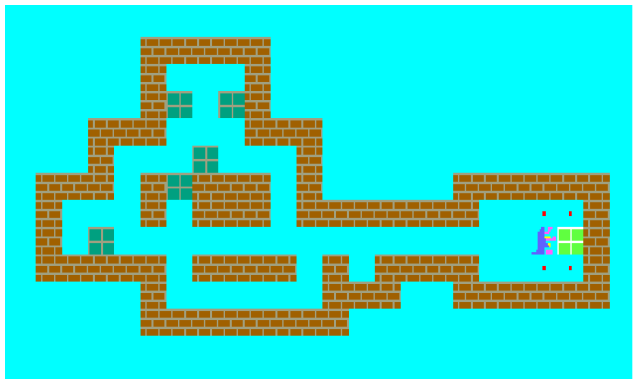
— Справа толкнуть не удастся. Это верно. Но ведь к мешкам (точнее, к одному из этих двух мешков) можно будет подойти не справа, а сверху. Для этого надо сначала отодвинуть вверх тот мешок, под которым стоит сейчас *Крот*.

И Тим уверенно сделал следующий ход — толкнул мешок вверх и сдвинулся на два шага влево:



— Ну а теперь ты и сам, наверное, видишь, что нужно делать дальше: достаточно толкнуть мешок, расположенный прямо под *Кротом*, вниз, и тогда откроется путь в левый вертикальный коридор, по которому легко подобраться слева к мешку, стоящему на перекрестке коридоров. Дальше совсем просто: толкай этот мешок стрелкой →, и очень скоро он окажется на складе. Ясно?

— Все ясно. Спасибо. Ты очень убедительно показал мне, как найти в лабиринте самое главное, или, как часто говорят, **ключевое место** лабиринтной задачи. Здесь это перекресток двух коридоров, занятый мешком. Сложность лабиринтной задачи и состоит прежде всего в отыскании ключевых мест. Как вы понимаете, таких мест в задаче может оказаться несколько. Но когда ключевое место обнаружено, то решение становится совсем простым.



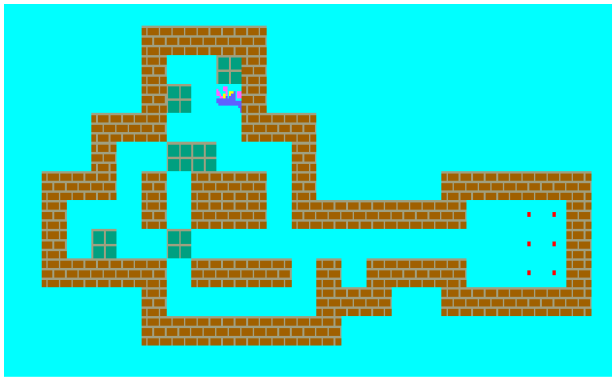
— Но ты видишь, дедушка, что с такой хитрой задачей мы разобрались сами! — не удержалась Дина, чтобы не похвалиться перед дедушкой.

— Молодцы, молодцы! — сказал дедушка, действительно порадовавшийся самостоятельности своих внуков. — Теперь мне остается только посмотреть, в чем вы уже успели разобраться, а что еще предстоит вам рассказать.

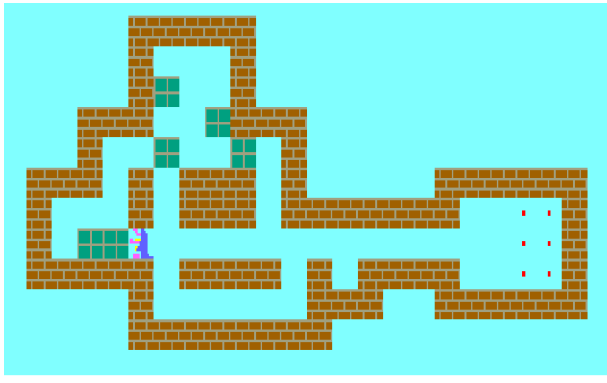
— Так, значит, ты все-таки знаком с *Мудрым Кротом*? — огорчилась Дина тем, что ее увлеченный рассказ про *Крота* не стал для дедушки откровением.

Пришлось дедушке во всем признаваться. Но зато дальше беседа пошла без ненужных осложнений. Тим, поняв, что дедушка — хороший специалист в лабиринтных задачах, задал мучивший его вопрос:

— Досадно, что иногда случайно задержишь палец на клавише со стрелкой — и *Крот* попадает в "аварию" — решить задачу становится невозможно.



Самый верхний мешок в таком положении уже не удастся больше никуда переместить.



Два мешка перед *Кротом* "навсегда" останутся на своих местах, да и остальные на склад не попадут никогда.

— Так иногда случается у новичков, которые еще не совсем освоились с клавиатурой компьютера. Но беды никакой нет: как только обнаружилось, что *Крот* сделал неверный шаг, можно легко вернуть его назад и тем самым исправить ошибку. Такой возврат назад, в предыдущее состояние, как верно сказала Дина, называется **откаткой**. Откатка выполняется аккордом из двух клавиш, расположенных в разных местах клавиатуры. Одна из них — клавиша Ctrl (на клавиатуре она расположена в левом нижнем углу клавиатуры), а другая — клавиша обратного пробела BackSpace (она находится в противоположном правом верхнем углу).

Если нажать аккорд Ctrl + BackSpace два раза, *Крот* с мешком "откатится" (то есть вернется назад) на два шага. Так его можно откатывать несколько раз подряд. Если же захочется начать решение с самого начала, достаточно всего один раз нажать аккорд Ctrl + Delete.

Очень важно запомнить, что откатка означает не один шаг *Крота* задом наперед, а возврат в предыдущее состояние всей окружающей среды, в которой живет *Крот* на экране компьютера, включая положение мешков в коридорах и на складе, показания счетчиков шагов и, конечно, положение *Крота*.

Имея такую операцию, как откатка, программа предлагает и обратную операцию — **накатку**. Ею пользуются тогда, когда откатка "завела" *Крота* слишком далеко назад и нужно снова пройти вперед. Чтобы сделать накатку, надо набрать аккорд Ctrl + Enter. Точно так же, как и откатка, накатка может выполняться несколько раз подряд.

Тим по этому поводу сделал обобщающее замечание, которое дедушка не мог не оценить:

— Я заметил, что во многих программах *Роботландии* есть откатка и накатка. И всегда она выполняется этими аккордами: откатка — Ctrl + BackSpace, а накатка — Ctrl + Enter.

После этого у дедушки были все основания уверенно продолжать:

— Итак, первую задачу *Крота* вы решили (а вы ее решили вместе? — прервал он сам себя вопросом, на который дети дружно закивали головами), но ведь сначала вы включали демонстрационный режим и смотрели, как за вас ее решал компьютер.

— Да, это правда! — признался Тим. — В справке, которая вызывается на экран клавишей F1, написано, что демо-режим включается для первого лабиринта клавишей F2. Но зато мы сами решили вторую задачу, а для нее демо-режима нет.

Дедушка отнесся к этому одобрительно:

— Это уже кое-что! Хотя скажу вам по секрету, что демонстрационный режим у *Мудрого Крота* предусмотрен для всех 27 задач, включенных в эту программу.

— Вот это да! — восхитился Тим незнакомой ему возможностью любимой программы. — А как его

Продолжение см. на с. 18

Музыкальный компакт-диск

В 1980 году, когда персональные компьютеры переживали пору младенчества, имели 64 Кб оперативной памяти и флоппи-дисководы на 160 Кб, корпорация Sony совершила переворот в звукозаписи, выпустив цифровой носитель информации емкостью 650 Мб. Правда, несмотря на свою цифровую природу, вначале он предназначался вовсе не для компьютеров. На него записывалась музыка, причем с невиданным доселе качеством. Никакие грампластинки, катушечные и (тем более) кассетные магнитофоны не могли обеспечить настолько качественное звучание. Разумеется, первые проигрыватели компакт-дисков (представляющие, по сути дела, специализированные компьютеры) стоили раза в два дороже персональных компьютеров того времени (около 2 тыс. долларов). Стоимость самих компакт-дисков тоже была “на высоте”.

Для того чтобы понять принцип записи музыки на компакт-диск, давайте разберемся с кодированием звуковой информации.

В отличие от графической информации, кодирование звука оказалось довольно-таки простым. С XIX века известны устройства, которые улавливают звуковые волны и преобразуют их в электрический ток переменного напряжения. Устройства эти называются микрофонами. Точно такой же непрерывный (или *аналоговый*) сигнал подается и в звуковые усилители, а затем на динамики — для воспроизведения звука. Задача, стало быть, состоит в том, чтобы придумать способ кодирования аналогового сигнала, изображенного, например, на *рис*. 1, с помощью чисел.

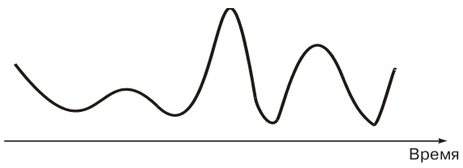


Рис. 1. Аналоговый сигнал
(например, напряжение, получаемое от микрофона)

Для этого достаточно несколько раз (несколько десятков, сотен, тысяч раз) в секунду измерять величину аналогового сигнала и кодировать ее... ну, скажем, с помощью 256 различных значений.

Фактически мы разбиваем плоскость вертикальными и горизонтальными линиями и приближенно считаем, что график проходит в точности через узлы получившейся сетки, заменяя плавную линию ломаной (*рис*. 2).

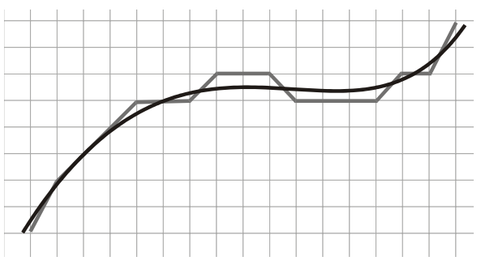


Рис. 2. Оцифровка аналогового сигнала

Вертикальное разбиение называется *дискретизацией* (по времени) аналогового сигнала и характеризуется *частотой дискретизации*. Обычно для кодирования звука выбирают одну из трех частот — 44,1 КГц, 22,05 КГц или 11,025 КГц. Самая “качественная” частота как раз и соответствует музыке, записанной на компакт-диск. Величина аналогового сигнала меряется в этом случае 44 100 раз в секунду. Естественно, отдельно кодируются левый и правый каналы в случае стереозвучания. Иными словами, расстояние между клеточками по горизонтали соответствует 1/44100 доле секунды.

Разумеется, важна и горизонтальная разбивка. Чем мельче сетка — тем качественнее мы приблизим аналоговый звук с помощью цифр. Горизонтальная разбивка называется *квантованием* (по уровню). Обычно применяются всего два его типа — 8-битное или 16-битное. В первом случае мы получаем 256 различных значений напряжения, во втором — 65 536 значений. Иными словами, интервал от нулевого до максимального напряжения разбивается либо на 256, либо на 65 536 уровней.

Восьмибитное квантование применяется для оцифровки обычной речи (в частности, телефонного разговора) и радиопередач на коротких волнах.

Шестнадцатибитное — для оцифровки музыки и УКВ радиопередач.

Давайте посчитаем, сколько будет занимать в килобайтах одна минута записи на музыкальном компакт-диске.

Итак, при шестнадцатибитном кодировании сигнала нам потребуются 2 байта на запись значения напряжения. Поскольку сигнал измеряется 44 100 раз в секунду, надо $2 \times 44\,100 = 88\,200$ байт каждую секунду, или примерно 86,1 Кб. За минуту это составит $86{,}1 \times 60 = 5166$ Кб, или примерно 5,05 Мб. Поскольку сигнал стереофонический, это значение нужно умножить на 2. Проведя еще несколько аналогичных вычислений, получим следующую таблицу, в которой указано, сколько мегабайт будет занимать закодированная таким образом одна минута звуковой информации:

Тип сигнала	Частота дискретизации в КГц		
	44,1	22,05	11,025
16 бит, стерео	**10,1 Мб**	5,05 Мб	2,52 Мб
16 бит, моно	5,05 Мб	2,52 Мб	1,26 Мб
8 бит, моно	2,52 Мб	1,26 Мб	*630 Кб*

Максимальное качество, соответствующее сигналу музыкального компакт-диска, выделено жирным шрифтом. Минимальное, соответствующее телефонному разговору, выделено курсивом. В частности, именно столько нужно будет автоответчику, программно установленному на вашем компьютере для записи входящих звонков.

Как видите, всего одна минута качественной музыки занимает столько же места, сколько и 10 толстенных 500-страничных томов (без иллюстраций!) или две качественные иллюстрации размером примерно в полстраницы того же тома.

Понятно, почему и звук, и музыка одинаково тяжело поддаются анализу с помощью компьютерных технологий.

Стандартный компакт-диск (CD) состоит из трех слоев: основы, отражающего и защитного. Основа выполнена из прозрачного поликарбоната, на котором методом прессования сформирован информационный рельеф. Поверх рельефа напыляется металлический отражающий слой (алюминий, золото, серебро, другие металлы и сплавы). Отражающий слой покрывается сверху защитным слоем поликарбоната или нейтрального лака — так, чтобы вся металлическая поверхность была защищена от контакта с внешней средой. Общая толщина диска — 1,2 мм.

Информационный рельеф диска представляет собой непрерывную спиральную дорожку, начинающуюся от центра и состоящую из последовательности углублений — питов (*pits*). Промежутки между питами носят название *lands*. Чередованием питов и промежутков различной длины на диске записывается закодированный цифровой сигнал: переход от промежутка к питу и наоборот обозначает единицу, а длина пита или промежутка — длину серии нулей. Расстояние между витками дорожки выбирается от 1,4 до 2 мкм, стандарт определяет расстояние в 1,6 мкм.

Запись на компакт-диске производится от центра к краю. Общая длительность записи на компакт-диске — 74 минуты, однако при уменьшении стандартного шага дорожки и расстояния между питами можно достичь увеличения времени записи — за счет снижения надежности считывания в стандартном дисковом приводе.

Основной способ изготовления дисков — прессование с матрицы. Оригинал формируется с исходной цифровой мастер-ленты, содержащей уже подготовленный и закодированный цифровой сигнал, специальным высокоточным станком на стеклянном диске, покрытом слоем фоторезиста — материала, изменяющего свою растворимость под воздействием лазерного луча. При обработке записанного оригинала растворителем на стекле возникает требуемый рельеф, который методом гальванопластики переносится на никелевый оригинал (негатив), который может служить матрицей при мелкосерийном производстве либо основой для снятия позитивных копий, с которых, в свою очередь, снимаются негативы для массового тиражирования.

Штамповка выполняется методом литья под давлением: с негативной матрицы прессуется поликарбонатная подложка с рельефом, сверху напыляется отражающий слой, который покрывается лаком.

С другой стороны диска, поверх защитного слоя обычно наносятся информационные надписи и изображения.

Необходимо избегать механического повреждения любой из поверхностей, попадания на диск органических растворителей и прямого яркого света, ударов и перегибов диска. Надписи на записываемых дисках допускается делать только карандашами или специальными фломастерами, исключая нажим и использование шариковых или перьевых ручек.

При извлечении диска из коробки следует остерегаться его перегиба. Не рекомендуется также касаться пальцами прозрачной защитной поверхности. Один из удобных и безопасных методов требует участия двух рук — большой палец левой руки слегка нажимает на фиксатор, ослабляя его, в то время как другая рука снимает диск с фиксатора. Метод с использованием одной руки, когда указательный палец ослабляет фиксатор, а большой и средний снимают диск, требует более точного согласования усилий, без которого легко перегнуть диск или сломать лепестки фиксатора.

Загрязненный диск можно мыть теплой водой с мылом или неагрессивным поверхностно-активным веществом (шампунь, стиральный порошок), либо специально выпускаемыми жидкостями. Неглубокие царапины на прозрачном слое можно заполировать — полирующими пастами, не содержащими органических растворителей и масел, или обычной зубной пастой.

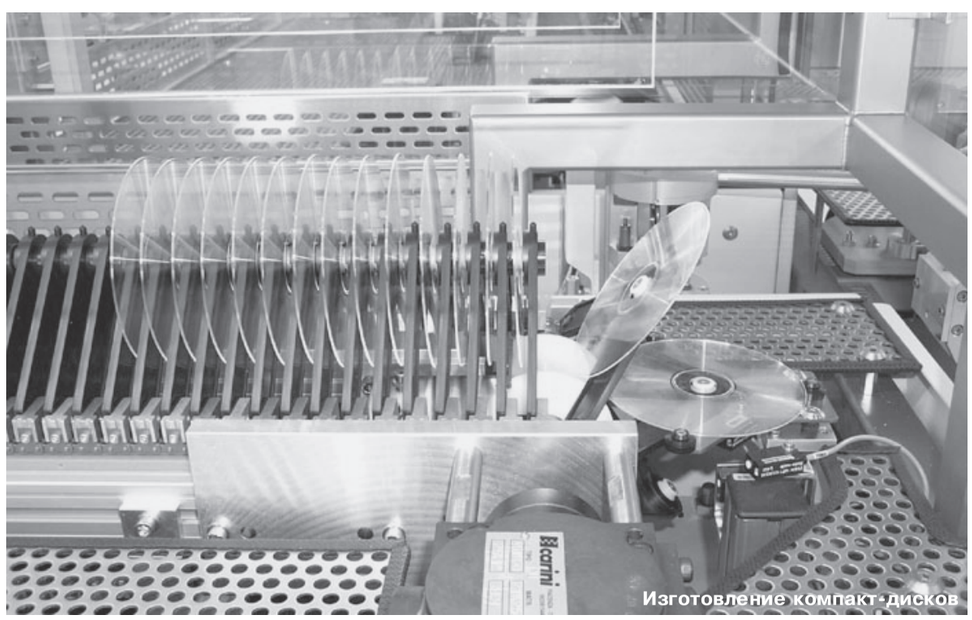
Протирать диск следует только в направлении от центра к краю и обратно, но не круговыми движениями. Для протирки необходимо использовать мягкую ткань типа фланели или мягкого хлопчатобумажного трикотажа.

Литература

1. *Евгений Музыченко* (**www.iXBIT.ru**).

2. *Шкритек П.* Справочное руководство по звуковой схемотехнике. М., “Мир”, 1991.

3. Оптические дисковые системы. М., “Радио и связь”, 1991.



Изготовление компакт-дисков

Продолжение. Начало см. на с. 13–15

запускать? Клавиша F2 на других лабиринтах не работает, я проверял.

Дедушка пояснил:

— Демо-режим для всех лабиринтов, кроме первого, вызывается аккордом Ctrl + F2. Разработчики программы — хитрые роботландцы специально не написали об этом в справке, чтобы у вас не было соблазна включать его каждый раз перед своим решением.

Тим заверил дедушку:

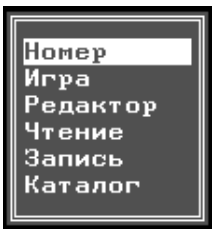
— Нет, я буду сам решать задачи. Подсматривать не интересно, если только уж совсем не будет получаться...

Вот только одно неудобство: сейчас я бы стал решать третью задачу, потому что две первых уже решены. Но, чтобы пройти третий лабиринт, снова придется проходить первые два, ведь к следующей задаче программа переходит только тогда, когда решена предыдущая. Это очень неудобно.

Дедушке пришлось утешать внука:

— Не торопись, а лучше нажми на клавишу перехода Esc.

— Нажал... На экране появилось меню:



— Первая позиция в этом меню — *Номер*. Она-то и предназначена для выбора произвольного лабиринта.

Тим, конечно, тут же решил попробовать.

— Нажимаю клавишу выполнения. На экран вывелось новое окошко:

Дина, до сих пор молча наблюдавшая за действиями брата, сказала:

— Я догадываюсь, как надо работать в этом окне: щелчок по верхней стрелке, номер увеличится на единицу, щелчок по нижней — уменьшается.

Тим не только согласился с Диной, но и понял, что должен подвинуть к ней клавиатуру. Правда, он не удержался от замечания:

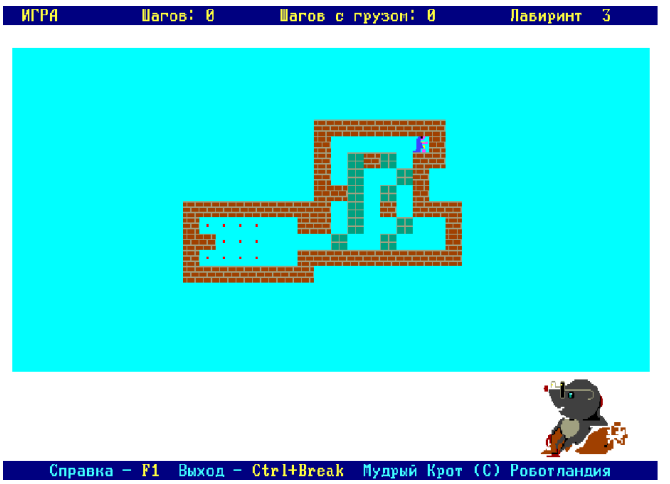
— Никаких щелчков тут не бывает, надо нажимать стрелки ↑ и ↓. Ведь *Крот* не понимает мышку.

Дина не стала спорить с Тимом (о том, что *Крот* управляется только клавишами — стрелками, она, конечно, знала):

— Поставлю-ка я четвертый лабиринт. Нажимаю стрелку ↑ — раз, два. В самом деле, в окошке появилось число 3. Нажимаю на Enter и снова попадаю в меню *Крота*. Ух, какой сложный лабиринт!

Теперь, конечно, в меню надо выбрать раздел *Игра*.

Дедушка кивнул в знак одобрения догадки. Дина сделала выбор. Меню исчезло с экрана, и лабиринт № 3 предстал перед ребятами во всей своей красоте и сложности.



3. Конструирование новых задач

— Вы обратили внимание на другие разделы кротового меню? — спросил дедушка.

Дина ответила:

— Да. Там есть раздел *Редактор*. Значит ли это, что, включив его, я смогу вызвать *Микрона*, *Художника* или, быть может, *Шарманщика*?

— Ну, конечно, нет! *Микрон* редактирует текстовую информацию, *Художник* — графическую, *Шарманщик* — музыкальную. У *Крота* есть свой, кротовый, редактор, и редактирует он лабиринты.

Ты можешь вывести на экран любой из хранящихся в памяти 27 лабиринтов *Крота*, но заняться не игрой, а совершенствованием задачи. Ты можешь добавить в лабиринт новые ходы, мешки, комнаты-склады, предусмотреть в них новые места для мешков, иными словами, ты сможешь заняться редактированием лабиринтных задач.

Дина тут же поинтересовалась:

— А смогу ли я поставить в лабиринт еще одного крота для помощи первому? Думаю, что в сложных лабиринтах это не помешало бы.

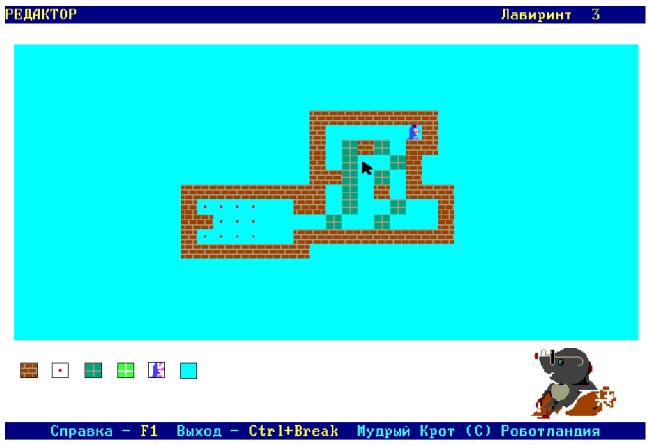
Но дедушка ее не утешил:

— Нет, редактор тебе этого не позволит. По правилам игры *Крот* в этой программе должен быть только один.

Дина миролюбиво согласилась:

— Ну что ж. Попробую для начала менять лабиринт с номером три — ведь он сейчас на экране. Вызываю меню, выбираю раздел *Редактор* и нажимаю клавишу выполнения. На экране появился курсор в виде наклон-

ной стрелочки, изменилась верхняя служебная полоска и появились "строительные" кубики в рабочем поле под лабиринтом:



— Я полагаю, что "строительный" материал выбирается курсором-стрелкой? — сделал предположение Тим, но на сей раз одобрительного подтверждения дедушки не последовало.

— Нет. Курсор-стрелка указывает то место в поле лабиринта, которое должно быть изменено. А меню выбора кубика имеет свой собственный курсор в виде контурного прямоугольника. Оба курсора управляются стрелками клавиатуры, но второй работает только тогда, когда вместе со стрелками (← или →) нажата клавиша переключения регистров Shift.

И Дину, и Тима волновал один вопрос:

— И как же редактировать лабиринт?

— Очень просто. Выбираешь в "строительном" меню нужный кубик, указываешь место на экране и нажимаешь пробел.

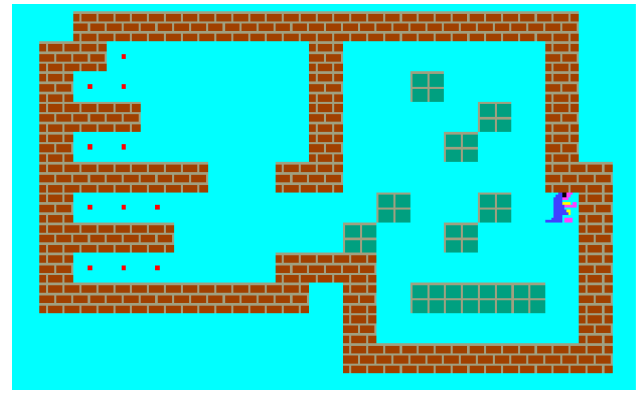
Дина уточнила свой вопрос:

— А что делать, если кубик поставлен неправильно и, значит, надо убрать его с экрана?

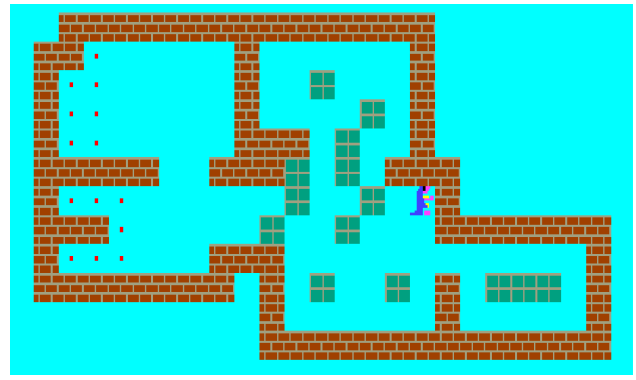
— Проблемы нет. Нажимаешь еще раз пробел, и поставленный кубик пропадает. Для удаления кубиков можно также выбрать в "строительном" меню последний "пустой" кирпич. И тогда курсор-стрелка вместе с пробелом будут работать как ластик. Редактор лабиринтов устроен очень просто и осваивается быстрее в деле, чем в пространных объяснениях. Начните работу, и вы увидите, что проблема не в технике рисования, а в том, чтобы придумать такой лабиринт, в который было бы интересно играть.

Очень скоро Дина и Тим поняли, насколько дедушка был прав, говоря о том, что построить красивую лабиринтную задачу непросто.

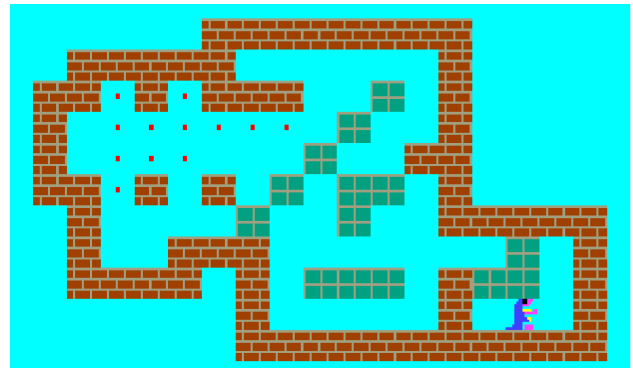
Первый лабиринт, наскоро построенный Диной, никуда не годился, несмотря на то, что в нем было много мешков. Перемещение их на склад было простым упражнением на работу со стрелками клавиатуры. Ни о каких ключевых местах в этом лабиринте и речи не могло быть.



Следующий лабиринт взялся делать Тим. Но и этот лабиринт тоже оказался плох: задача *Крота* в нем была неразрешима:



Тогда Дина и Тим решили, что такую работу надо делать вместе, и, наконец, после долгих усилий они гордо показали дедушке свое последнее творение:



Дедушка был доволен:

— Задача получилась неплохая, молодцы! Хотя, конечно, и она не из самых сложных.

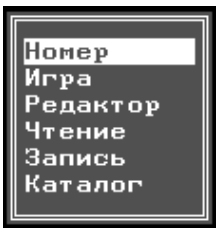
— Дедушка, — вдруг попросила Дина, — мне бы хотелось эту задачу, которую мы с Тимом составили вместе, показать моей подруге Наташе. Но ей-то я могу отнести наш лабиринт только на дискете. Ты не покажешь нам, как можно сохранить нашу работу?

— Это тот вопрос, которого я от вас ждал, ребята. Конечно, хорошую работу надо сохранить. Запомнили ли вы, глядя на меню *Мудрого Крота*, какие его позиции относятся к работе с диском?

По школьной привычке Тим поднял руку и, не дожидаясь приглашения к ответу, начал быстро перечислять: *Читать*, *Писать*, *Каталог*.

— А вот и не так, — вмешалась Дина. — Надо так: *Чтение*, *Запись*, *Каталог*.

— Это точнее, — согласился дедушка и, нажав Esc, снова вывел на экран основное меню *Мудрого Крота*:



Теперь надо войти в позицию *Запись*.

Дина отметила:

— Готово! Теперь программа просит ввести имя файла, то есть имя нашей задачи. Я правильно понимаю?

Опережая дедушку, Тим подтвердил:

— Ты правильно понимаешь, Дина. Предлагаю назвать файл именем FIRST. Наташа сразу догадается, кто его строил.

Все имена лабиринтов имеют в качестве расширения слово LAB. Но писать его не обязательно, потому что программа автоматически присоединяет его к вводимому имени. Поэтому ваш на диске получит имя FIRST.LAB.

— Я не возражаю, — согласилась Дина, но все же осторожно спросила дедушку: — Нашу задачу мы получили, редактируя третий лабиринт *Мудрого Крота*. Значит, теперь он безвозвратно утерян?

Дедушка ответил ободряюще:

— Не волнуйся, Диночка. Когда ты снова запустишь Крота, то увидишь, что третий лабиринт на своем месте. А свой лабиринт FIRST.LAB ты можешь загружать с диска из позиций меню *Чтение* или *Каталог*, независимо от того, какой лабиринт сейчас находится на экране. Твой файл его временно заменит, до следующего запуска Крота.

Тим спросил:

— А можно ли строить собственную задачу, не редактируя готовые лабиринты *Крота*?

Дедушка немного задумался, как точнее ответить на этот вопрос:

— Гм... Специального "чистого" листа для построения новых лабиринтов в программе нет. Но в редакторе *Мудрого Крота* всегда можно очистить лабиринтное поле при помощи тройного аккорда Ctrl + Shift + Delete.

Вполне удовлетворенный, Тим ответил дедушке:

— Ну, это почти что одно и то же. Согласитесь, что намного удобнее строить задачу на чистом поле, чем разгребать завалы чужих кирпичей, мешков и складов, в которых очень легко запутаться.

Дедушка подвел итоги сегодняшней встречи:

— Я рад, что *Мудрый Крот* вас заинтересовал. Поэтому вас не затруднит домашнее задание: каждому — сконструировать по два собственных лабиринта.

— Конечно, нет, — подтвердили дети. — Мы даже соревнование устроим — чей лабиринт лучше.

4. Чей лабиринт лучше?

— Ну-ка, покажите мне ваши лабиринты. — Так встретил внуков дедушка Фёрстов в своем кабинете на следующий вечер.

— Ты будешь нам ставить оценки? — высказала Дина гипотезу, понимая, однако, что она невероятная.

— Нет, нет! Правда, оценки сегодня будут, но ставить их будете вы сами.

Дети, постепенно привыкающие к неожиданным поворотам дедушкиных бесед, не испугались упоминания об оценках, они приготовились внимательно слушать.

— Во-первых, — начал дедушка, — коль скоро закончилась важная тема нашего курса, посвященная *Мудрому Кроту*, значит...

— Значит, будет новый турнир! Ура! — Торопыжка Тим не дал закончить дедушке ожидаемую фразу.

— Не перебивай дедушку, — назидательно заявила Дина брату, хотя и ей тоже нравилась идея предстоящего турнира *Мудрого Крота*. Команда Фёрстовых очень неплохо выступила в предыдущем турнире-конкурсе пиктограмм, и ребятам, конечно, хотелось закрепить успех.

Тим понял, что надо извиниться, и поддержанный внучкой дедушка продолжил:

— Верно. Турнир будет. И вашу работу в турнире надо будет оценивать. Вот тут-то и понадобятся оценки.

Дина решила уточнить:

— Но ты сказал, дедушка, про оценки, которые будем ставить мы. А ведь за нашу работу по конструированию лабиринтов оценку будешь ставить ты.

— Но я еще не сказал "во-вторых".

— И что же такое "во-вторых"?

— Это тоже знакомая вам работа — перекрестная проверка. Вам предстоит проверить лабиринтные задачи, составленные ребятами из других команд. Скоро эти задачи мы получим по электронной почте.

— Может, есть и "в-третьих"? — в шутку спросил Тим.

Но дедушка ответил серьезно:

— Есть и "в-третьих". Дело в том, что перед тем, как вы отправите свои лабиринты другим командам на перекрестную проверку, вам рекомендуется сделать самооценку.

— Сделал работу и сам за нее ставишь себе оценку? — недоверчиво спросил Тим. — Такого у нас еще никогда не было. Тогда каждый захочет поставить себе отличную оценку.

— Ты же понимаешь, что самому себе очень стыдно ставить хорошую оценку, если сам видишь свои недостатки. Получить плохую оценку, конечно, неприятно, но гораздо хуже (намного более стыдно!) показать всем ребятам (они же будут проверять твою работу вместе с твоей самооценкой), что ты себе за плохую работу ставишь хорошую оценку. Можешь быть уверен, что свои задачи ребята оценят честно.

— Ты убедил нас, дедушка, что мы должны знать, по каким правилам выставляются оценки за кротовые лабиринты, — признался Тим. — Мне кажется, что оценить лабиринтную задачу труднее, чем пиктограмму.

— Вот тут ты совершенно прав, — ответил дедушка. — Но главное в оценке лабиринтов мы возьмем от тех же способов, которыми оценивались пиктограммы. Хотя от лабиринта могут сложиться разные впечатления у различных людей, все же если мнений будет собрано мно-

го, то и усредненная оценка будет объективнее, то есть справедливей. В этом вы убедились. А кроме того, лабиринт, так же как и пиктограмму, мы будем оценивать не одной оценкой, а несколькими: по одной оценке за каждое из разных его качеств.

Дина и Тим уже сгорали от нетерпения скорее узнать, "что такое хорошо и что такое плохо" в лабиринтных задачах *Мудрого Крота*.

Дедушка начал "раскладывание по полочкам":

— Прежде чем назвать вам критерии оценки лабиринтов, я должен сказать о главном требовании к конструируемой задаче: она должна иметь решение.

— А как же иначе! — удивились дети такому простому требованию.

— Почему вы удивляетесь? — спокойно сказал дедушка. — Ведь в прошлый раз Тим старался-старался, а лабиринт у него получился неразрешимый. Помните?

И, не развивая далее эту мысль, дедушка выдал рекомендацию:

— Значит, закончив конструирование задачи, вы должны прежде всего убедиться, что она имеет решение. За неразрешимую задачу надо ставить даже не кол, а оценку, которой в школе не бывает, — ноль.

Ну а теперь главное. За каждый лабиринт вам будет поставлена не одна, а пять оценок. Первые две из них — *N*1 и *N*2 — оценивают ключевые места в лабиринте, третья, *N*3, — количественную простоту и четвертая, *N*4, — красоту лабиринта.

— Ничего себе, оценки! — снова удивился Тим. — Сразу даже и не поймешь, за что такие оценки ставить.

— Труднее всего оценивать, пожалуй, ключевые места, — признал дедушка. — *N*1 — это оценка сложности ключевого места в лабиринте. Она оценивается числом от 1 до 3: оценку 1 получают простые ключевые места, 2 — средней сложности, 3 — ключевые места повышенной сложности.

Дина усомнилась в справедливости такой оценки:

— Но ведь одному ключевое место может показаться простым, а другому — сложным. Разве можно так поразному оценивать?

Дедушка посмотрел на Тима — не подскажет ли он нужный ответ. И Тим не подкачал:

— Один может ошибиться в одну сторону, другой — в другую. Но если собрать оценки многих проверяющих, то и получится в среднем правильная оценка. Ты же сама говорила, Дина, про оценки перекрестных проверок в конкурсе пиктограмм, что они напоминают тебе оценки судей на фигурном катании. Здесь ведь очень похоже. Правда, дедушка?

Дедушка порадовался и тому, что внук умеет так четко выразить свои мысли, и тому, что в конечном итоге ему все-таки нужна поддержка со стороны дедушкиного авторитета.

— Для того чтобы поставить окончательную оценку *N*1, — продолжил он, — судья (или проверяющий) должен знать ключевые места если не всех, то по крайней мере нескольких встроенных лабиринтов *Мудрого Крота*.

— Встроенных? — не поняли ребята.

— Да, встроенных. Так называют лабиринты, которые доступны каждому пользователю программы. Файлы этих лабиринтов находятся в одном каталоге с программой *Мудрый Крот*. Приобретая *Мудрого Крота*, пользователь получает вместе с ними и эти 27 уже готовых лабиринтов. Поэтому они и называются встроенными (в программу). В отличие от лабиринтов, которые конструирует пользователь в редакторе *Мудрого Крота*, встроенные лабиринты различаются не по именам, а по специальным номерам.

Встроенные лабиринты очень полезны для обучения решению лабиринтных задач. Но если ученик, познакомившись с интересным встроенным лабиринтом, переносит в конструируемый лабиринт запомнившееся ему ключевое место, а не изобретает его сам, то, понятно, оценку за такой лабиринт надо чуть понизить. Если же проверяющий убежден, что ученик придумал новое, свое ключевое место в лабиринте, то *N*1 надо увеличить вдвое, умножить на 2. Таким образом, окончательная величина *N*1 оценивает не только сложность, но и новизну ключевого места.

Наконец, когда подсчитана величина сложности и новизны каждого ключевого места, все эти величины надо сложить и сумму умножить на 5. Полученное произведение и станет оценкой *N*1 для всего лабиринта.

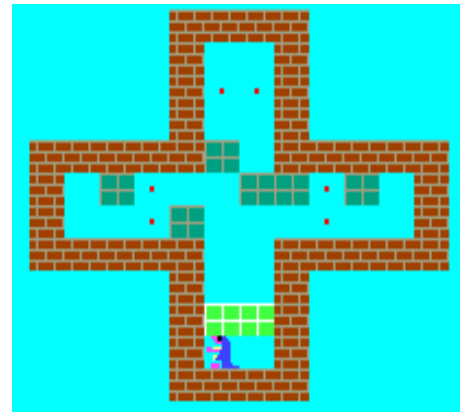
Теперь перейдем к оценке N2, которая определяет разнообразие ключевых мест. Она выражается числами 3, 6 и 9. Оценку 3 получают лабиринты, содержащие ключевое место только одного типа, оценку 6 — лабиринты, имеющие разные по типу ключевые места, оценку 9 — лабиринты, которые не содержат повторяющиеся типы ключевых мест. Для лабиринтов с единственным ключевым местом $N2 = 3$.

Третье оцениваемое качество лабиринта называется "количественная простота".

— Только что оценивали сложность, а теперь говорим о простоте! — подивилась Дина таким различным подходам к оценкам.

— Нетрудно сделать несколько разных ключевых мест, если выстроен огромный лабиринт из большого числа кирпичей и мешков. Труднее построить сложную и интересную лабиринтную задачу, использовав для этого по возможности меньше кирпичей. Для оценки *N*3 предлагается такая формула: $N3 = 700/n$, где n — это число кирпичей в лабиринте.

Вот взгляните на хитрый лабиринт, где совсем не много кирпичей и мешков.



И Дина, и Тим с интересом разглядывали этот лабиринт и даже попытались его решить. Однако "на ходу", быстро задача не поддавалась, и дедушка рекомендовал отложить это интересное упражнение на следующий день, "на свежую голову", как он выразился.

Необычный лабиринт не только вызывал интерес, но и показал им неожиданные дополнительные заботы:

— Получается, что при проверке каждой работы нам придется считать кирпичи? Ну и работка! Скучнее не придумаешь, — недовольно пробурчал Тим.

И на этот раз дедушка нашел утешение:

— Ну зачем же так огорчаться! У *Мудрого Крота* есть такая вспомогательная программа, утилита, с помощью которой легко посчитать количество кирпичей и мешков. Называется она LABTOTXT.EXE. Для того чтобы ее запустить, надо написать имя этой программы в командной строке (программу можно использовать только в программной оболочке DOS), а через пробел написать имя того лабиринта, для которого ты хочешь посчитать количество кирпичей. Например, для подсчета кирпичей и мешков лабиринта FIRST.LAB в командной строке записывают LABTOTXT.EXE FIRST.LAB

Тим тут же написал такую строку и запустил эту программу. Дедушка объяснил:

— В качестве своего результата программа создает текстовый файл FIRST.TXT.

— А-а, понятно: основа имени текстового файла берется от имени лабиринта, а расширение — обычное для текстовых файлов — TXT.

— Правильное наблюдение, — похвалил дедушка. — А если у вас есть текстовый файл, то вы можете его посмотреть.

— Нажав клавишу F3, — успел со своей подсказкой Тим.

Пришлось дедушке и Тима тоже похвалить.

— Верно. Попробуй-ка!

Тим установил курсор-подсветку на имя файла FIRST.TXT и нажал F3.

— Ой, как здорово! — восхитилась Дина. — Тут не только подсчитано все, что нужно знать про лабиринт, — число кирпичей, мешков и мест на складе, но даже нарисован сам лабиринт. Правда, нарисован он схематично, но все обозначения понятны, легко догадаться, как выглядит лабиринт на рабочем поле *Крота*. Полезная программа!

— Наконец, последняя оценка ставится за красоту лабиринта.

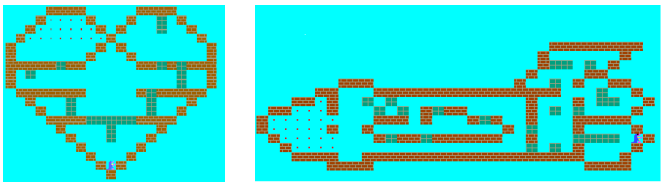
Пока дедушка подбирал слова для рассказа внукам про красоту кротовых лабиринтов, Тим в очередной раз высказал предположение о дальнейших действиях:

— Сейчас мы начнем разукрашивать лабиринты разными красками! — на что Дина рассудительно заметила:

— "Красивый" — это не значит, что разукрашенный.

Их спор не разгорелся, потому что дедушка вновь взял беседу в свои руки:

— Внешний вид лабиринта (расположение комнат, коридоров, мешков, полок на складе и начальное положение самого *Крота*) должен быть по возможности привлекательным. Как этого добиться? Готовых рецептов, конечно, нет. Можно, например, строить лабиринты симметричными, хотя иной раз нарушение симметрии тоже может выглядеть красиво. Привлекательным может показаться лабиринт, если внешне он похож на симпатичного зверя, насекомое или какую-нибудь машину. Иными словами, надо не забывать о красоте представления, проявляя выдумку и вкус. Давайте, я вам покажу пару примеров красивых, на мой взгляд, лабиринтных задач.



Дедушка взял в руки мышку и, очень ловко щелкая ею, стал прыгать из каталога в каталог, пока, наконец, не вывел на экран два таких лабиринта (по очереди):

Ребята залюбовались лабиринтами.

— Мне пока трудно оценить, насколько они сложны. Но уж точно — красивы! Особенно вот это сердечко! Это ты придумал такие лабиринты, дедушка?

— Нет, не я, — признался дедушка. — Это творчество ваших ровесников и однокурсников. Лабиринт-самолет построил Миша Неустроев из Иркутска, сердечко придумала Диана Сергеева из Красноярска, а тот лабиринт *Крест*, который я вам показал раньше, — дело рук (и головы!) Виталия Белова из Барнаула.

— Так как же ставится оценка за красоту? — задал вопрос дотошный Тим.

Дедушка объяснил:

— Красоту лабиринта договариваются оценивать одним из трех чисел — 3, 6 и 9. Оценку 3 получают некрасивые лабиринты, оценку 6 — привлекательные, оценку 9 — очень красивые.

— Дедушка, ты всегда говоришь очень убедительно. Но как хорошо бы увидеть эти способы проставления оценок на каком-нибудь примере.

— Согласен, Тим. Пример необходим. Я сейчас на ваших глазах заполню оценочную таблицу оценками трех знакомых вам лабиринтов — это первые два встроенных лабиринта *Мудрого Крота* (вы в них играли) и лабиринт FIRST.LAB (его вы сами построили). А вы попробуйте, глядя на эту таблицу, сказать, согласны ли вы с такими оценками. В любом случае такой взгляд на

Лабиринт	Число ключевых мест	N1	N2	Число кирпичей	N3	N4	Сводная оценка N1 + N2 +N3 + N4
Встроенный лабиринт 1	1	1 * 5 = 5	3	79	700/79 = 9	6	5 + 3 + 9 + 6 = 23
Встроенный лабиринт 2	2	(2 + 1) * 5 = 15	3	62	700/62 = 11	6	15 + 3 + 11 + 6 = 35
Встроенный лабиринт 3	3	(2 + 2) * 5 = 20	3	58	700/58 = 12	6	20 + 3 + 12 + 6 = 41
FIRST.LAB	1	1 * 5 = 5	3	66	700/66 = 11	3	5 + 3 + 11 + 3 = 22

таблицу поможет вам и при самооценке, и при перекрестных проверках.

— Что ж, попробуем разобраться, — не очень уверенно произнес Тим. — Но ведь, если будет какая-нибудь трудность, мы все равно тебя будем спрашивать. Ладно?

— Договорились! — ответил дедушка Фёрстов и приподнялся из своего кресла, чтобы попрощаться с внуками после вечера, наполненного задачами, лабиринтами, компьютером и... *Мудрым Кротом*.

Книга для учителя

ГЛАВА 2. МУДРЫЙ КРОТ

Задачи Мудрого Крота привлекают детей, особенно тогда, когда решение строится на соревновательной основе.

Каждый лабиринт содержит свою "изюминку", **ключевое место**, разгадав которое можно считать задачу принципиально решенной.

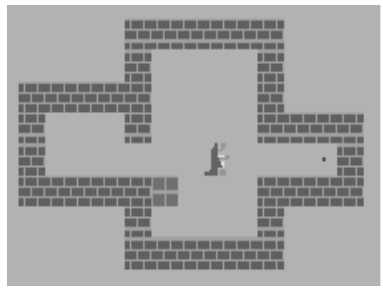
Число "лабиринтных" приемов не так велико.

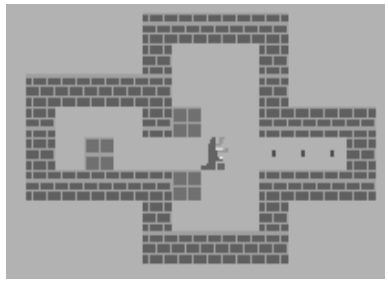
• Типична ситуация с выбором единственного мешка из всех возможных для выполнения очередного хода (неверный выбор приводит к неразрешимости задачи или является непродуктивным).

• Часто очень важной оказывается очередность заполнения "полок" на складе (неверный порядок складирования приводит к невозможности заполнения всех свободных мест).

• Еще один типичный пример — "отклеивание" мешка от стенки.

Мешки в приведенных примерах "отклеиваются" проталкиванием их в свободную комнату. Во втором лабиринте комнату надо предварительно освободить.





Существуют и другие приемы, превращающие задачи *Крота* в красивые головоломки.

Разработка новых ключевых позиций в дополнение к тем, которые присутствуют во встроенных лабиринтах *Крота*, — это интересная задача.

Понятно, что первый цикл занятий строится на решении готовых, встроенных лабиринтов. Сначала дети просто осваиваются с новой средой, привыкают к интерфейсу программы и решают задачи на интуитивном уровне.

На втором этапе дети продолжают решение готовых задач, но руководитель обращает внимание на ключевые ходы, решающие задачу в принципе. На этом этапе можно выдвинуть тезис о выделении таких ходов, их описании и классификации. С одной стороны, это поможет детям быстрее решать готовые лабиринты, а с другой, послужит основой для осмысленного построения своих собственных задач.

Третий цикл занятий — конструкторский. Перед ребятами ставится задача о построении хорошего лабиринта. Обсуждаются критерии, по которым будет оцениваться работа:

• **Разрешимость** лабиринта. Построенная задача должна иметь решение.

• **Ключевые места лабиринта**.

а) Наличие ключевых мест. Решение не должно сводиться просто к задаче управления курсором-*Кротом* по извилистой траектории, оно обязано содержать ключевые ходы, обдумывание которых и представляет собой цель построения.

б) Сложность ключевых мест. Ключевые места могут быть разного уровня сложности. Разрешение одного ключевого места может быть слабо связано с разрешением последующих. Иногда такая связь очень тесная, и, решая одну ключевую проблему, приходится задумываться на несколько ходов вперед. Такие ключевые места, конечно, сложнее и оцениваются выше.

в) Разнообразие ключевых мест. Можно построить лабиринт, в котором одно и то же ключевое место повторяется много раз. В качестве учебного задания, своеобразного лабиринтного этюда, это может быть оправданным, но хороший лабиринт-головоломка должен содержать разнообразные ключевые приемы.

г) Наличие динамических ключевых мест. В простых лабиринтах ключевые места могут быть изолированы друг от друга и задаваться только начальным расположением мешков и Крота. Назовем такие ключевые места статическими. Более сложной задачей для конструктора является построение динамического ключевого места, то есть такого, которого нет в начальной конфигурации и которое возникает в процессе решения задачи. Лабиринты с динамическими ключевыми местами более интересны своей неожиданностью.

Задача *Крота* может иметь несколько решений, в частности, и не запланированных проектировщиком. В разных решениях может возникать разное количество динамических ключевых мест, в том числе созданных самим учеником в процессе решения. В силу этого динамические ключевые места лабиринта плохо поддаются учету.

д) Новизна ключевого места. Лабиринты, построенные с придуманными ключевыми местами в отличие от тех, в которых используются приемы встроенных лабиринтов, оцениваются выше.

• **Количественная простота**. Лабиринт должен содержать минимум "кирпичей" при максимуме идей.

• **Красота лабиринта**. Внешний вид лабиринта (расположение комнат, коридоров, мешков, полок на складе и начальное положение самого *Крота*) должен быть эстетически интересным, красивым, привлекающим внимание своею формой.

Задание: "хитрые" приемы Крота

Это очень серьезное задание для детей и учителя — руководителя команды. Предлагается разработать классификацию "ключевых" мест лабиринта, проиллюстрировать ее соответствующими лабиринтными этюдами и предложить методы решения.

Решение этой задачи может оказаться для детей первым серьезным научным исследованием. Учитывая сложность и неформальность той работы, организаторы считают предложенное задание факультативным. Если руководителя оно не вдохновит или у него не окажется необходимого времени, то задание можно опустить и не предлагать его детям.

Работы детей надо оформить в виде текстового файла, в начале которого учитель записывает следующие пометки:

- Руководитель команды
- Регистрационный номер команды
- E-mail команды
- Задание: хитрые приемы *Крота*
- Автор (авторы)

Задание: конкурс лабиринтов

Для сборника лабиринтных задач и участия в конкурсе команда высылает в Роботландский университет четыре лучших лабиринта, построенных в редакторе *Мудрого Крота*. Руководитель команды должен убедиться в том, что все лабиринты, посылаемые на конкурс, по крайней мере разрешимы. (Разрешимость лабиринта должен продемонстрировать его автор.)

Набор лабиринтов должен сопровождаться текстовым файлом с описанием каждого из них по следующей схеме:

Заголовок файла
- руководитель команды
- регистрационный номер команды
- E-mail команды

Описание лабиринтов
- название лабиринта
- файловое имя лабиринта
- автор (авторы)
- описание ключевых мест лабиринта
- авторская оценка лабиринта

Критерии оценок

Предлагаемая числовая характеристика лабиринта не является безусловно объективной, но содержит в себе рациональное начало, проверенное практикой нескольких конкурсов-турниров *Мудрого Крота*.

Предлагается оценивать лабиринт в виде суммы взвешенных оценок за отдельные его качества, кратко обозначенные, как

- наличие ключевых мест,
- сложность ключевых мест,
- новизна ключевых мест,
- разнообразие ключевых мест,
- количественная простота,
- красота лабиринта.

Такое "измерение" лабиринта дает в руки проверяющему некоторый формализм, который позволит более объективно оценивать задачу. Кроме того, участвуя в перекрестной проверке работ, дети продолжают свое обучение. Они будут анализировать лабиринты, выделяя в них ключевые позиции, оценивать их сложность. Они будут задумываться над дизайном изображения, упражняться с утилитой LABTOTXT.EXE для подсчета числа кирпичей. Есть основания полагать, что такой подход к проверке оправдывает ее трудоемкость. Ниже описана методика построения числовой оценки.

Неразрешимые лабиринты получают оценку 0. Таких лабиринтов не должно быть в коллекции, присылаемой на конкурс.

Остальные лабиринты оцениваются по формуле:

$$N = N1 + N2 + N3 + N4,$$

— где $N1$ — оценка за наличие ключевых мест, их сложность и новизну. Она вычисляется как сумма оценок за сложность и новизну каждого ключевого места, умноженная на 5. Сложность каждого ключевого места оценивается числом от 1 до 3. Оценку 1 получают простые ключевые места, 2 — средней сложности, 3 — ключевые места повышенной сложности. Если ключевое место придумано школьником и не встречается во встроенных лабиринтах Крота, оценка за сложность такого места умножается на два.

$N2$ — оценка за разнообразие ключевых мест выражается числами 3, 6 или 9. Оценку 3 получают лабиринты, содержащие ключевые места только одного типа, оценку 6 — лабиринты, имеющие разные по типу ключевые места, оценку 9 — лабиринты, не содержащие повторяющихся типов ключевых мест.

Для лабиринта только с одним ключевым местом $N2 = 3$.

$N3$ — оценка за "количественную простоту", вычисляется по формуле: $700/n$, где n — число кирпичей в лабиринте.

Замечание. Для подсчета числа кирпичей в лабиринте рекомендуется использовать утилиту LABTOTXT.EXE. Эта утилита позволяет переводить лабиринтные коды в псевдографический рисунок. Число кирпичей в кладке записывается утилитой перед рисунком лабиринта.

$N4$ — оценка за "красоту" выражается числами 3, 6 или 9. Оценку 3 получают некрасивые лабиринты, оценку 6 — привлекательные, оценку 9 — необычайно красивые.

В учебнике приведена таблица-пример с оценками, построенными на основе предложенного критерия для первых трех встроенных лабиринтов и лабиринта FIRST.LAB, который построили дети — персонажи учебника.

Продолжение следует

Азы информатики

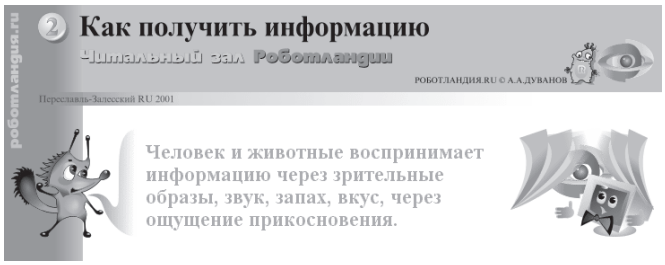
А.А. ДУВАНОВ

Материалы Роботландского университета

Книга 1. См. № 1, 2/2002
Книга 2 (начало). См. № 5/2002

Книга 2. В мире информации (продолжение)

Книга для ученика



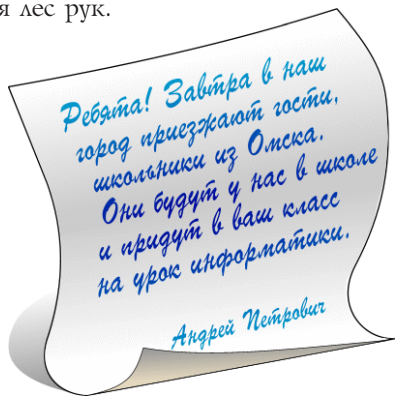
2. КАК ПОЛУЧИТЬ ИНФОРМАЦИЮ

Читальный зал

Виды представления информации

Сан Саныч попросил ребят привести примеры информации, напомнив им, что информация — это любые сведения.

Поднялся лес рук.



— Записка директора школы! Она содержит сведения о предстоящем визите гостей.

— Таблица умножения. Это сведения о результатах умножения однозначных чисел.

— План нашего города с транспортными маршрутами. Если на него посмотреть, то получишь сведения о том, как добраться до нужного места.

И записка, и таблица умножения, и план города — это примеры информации: они дают людям интересующие их сведения. Вместе с тем эти три сообщения представлены по-разному, они имеют разный вид.

Информация первая — записка Андрея Петровича. Это текст, состоящий из слов. Такую информацию называют обычно **текстовой**.

Информация вторая — таблица умножения. Она состоит из чисел.

Это **числовая** информация.

Информация третья — план города. Она представляет собою рисунок, условное графическое изображение местности. Такую информацию называют **графической**.

Здесь перечислены далеко не все виды представления информации. Прежде всего многие сообщения смешивают информацию разных видов. Например, информация о прибытии поезда:

Поезд 16 прибывает
в 22 часа 15 минут на 3-ю платформу!

— это смесь текстовой и числовой информации.

На плане местности, кроме линий, обозначающих реки, дороги, улицы, мосты, часто записывают поясняющие слова. Это текстово-графическая информация. Глядя на шкалу градусника, врач рассматривает метки расчерченной шкалы и расставленные на шкале числа. Измеряя температуру больного, он пользуется смешанной графическо-числовой информацией.

Кроме числовой, текстовой и графической информации, которую мы обычно видим глазами, человек способен воспринимать информацию и во многих других формах.

Как человек воспринимает информацию

У человека есть пять органов чувств (глаза, уши, нос, язык, кожа), которые позволяют ему получать информацию. Человеческие чувства восприятия информации называются “зрение”, “слух”, “обоняние”, “вкус”, “осязание”.

Глазами люди различают цвета, воспринимают **зрительную** информацию, к которой относятся и текстовая, и числовая, и графическая.

Ушами воспринимается **звуковая** информация — речь, музыка, звуковые сигналы, шум.

Носом люди получают информацию о **запахах** окружающего мира.

С помощью языка можно получить информацию о том, каков предмет на **вкус,** — горький, кислый, сладкий, соленый.

Кончиками пальцев (или просто кожей), **на ощупь**, можно получить информацию о температуре предмета, горячий он или холодный, о качестве его поверхности — гладкий или шершавый.

Органы чувств есть не только у людей, но и у животных. Важность того или иного информационного чувства у животных может быть совсем иной, чем для человека.

Больше всего информации об окружающем мире человек получает с помощью зрения, на втором месте — слух.

А вот для лисы, собаки, да и многих других животных основная информация та, которая поступает через орган обоняния — нос.

Для летучих мышей, которые видят плохо, главная информация — звуковая. Они воспринимают ее своими большими, чуткими ушами.

Информатика — это универсальная наука

Информатика изучает способы хранения, передачи и алгоритмы обработки информации. Эта наука не изучает важность информации, ее новизну и полезность.

В этом нет ничего удивительного! Например, математика изучает числа, но для нее не важно, что эти числа означают.

Математика учит: 10 поделить на два, получится 5. При этом для математики совершенно все равно, что делится на два: кролики, яблоки или рубли.

Представляете, что кролики делились бы на два не так, как яблоки, а яблоки не так, как рубли!

Математика работает с числами, и для нее не важен их смысл (кролики, яблоки или рубли).

Информатика работает с информацией, и для нее не важно ее содержание (прогноз погоды, стихи или сказка про Буратино).

Если алгоритм умеет находить в тексте символ, который встречается чаще других, то он это сделает с равным успехом и в тексте про Незнайку, и в тексте про Красную Шапочку.

Информатика выводит правила работы с информацией независимо от смысла информации, как математика выводит правила работы с числами независимо от того, что эти числа обозначают. Поэтому информатика и математика — это универсальные науки.

Конспект

Человек и животные воспринимают информацию через свои органы чувств.

У человека и животных пять разных чувств восприятия информации:

— зрение (глаза);
— слух (уши);
— обоняние (нос);
— вкус (язык);
— осязание (кожа).

Больше всего информации человек воспринимает глазами. Это тексты (**текстовая** информация), числа (**числовая** информация) и картинки (**графическая** информация).

Информатика — универсальная наука. Это означает, что она изучает общие правила работы с информацией, которые справедливы для любого содержания.

Вопросы

1. Приведите примеры текстовой информации.
2. Приведите примеры числовой информации.
3. Приведите примеры графической информации.
4. Приведите примеры смешанной информации.
5. Назовите 5 органов чувств человека, посредством которых он воспринимает информацию.
6. Какой вид информации человек воспринимает глазами?
7. Какой вид информации человек воспринимает ушами?
8. Какой вид информации человек воспринимает носом?
9. Какой вид информации человек воспринимает языком?
10. Какой вид информации человек воспринимает кожей?
11. Приведите примеры, когда одно и то же чувство восприятия информации для животных дает больше сведений, чем для человека.
12. Укажите, в каком виде представлена информация в следующих примерах:
— примеры по математике;
— географическая карта;
— письмо;
— картина;
— радиопередача;
— телевизионная передача;
— ароматный запах, желтый цвет и кислый вкус лимона;
— этот учебник.
13. Что изучает информатика?
14. Почему информатика является универсальной наукой?
15. Можно ли назвать универсальной науку, которая учит считать кроликов?
16. Можно ли назвать универсальной науку, которая изучает алгоритмы обработки текста сказки “Красная шапочка”?
17. Важен ли для человека смысл информации?
18. Нахождение периметра прямоугольника Рома выполняет по следующему алгоритму:
1) Сложи два заданных числа.
2) Умножь результат на два.
Ответьте на следующие вопросы:
— Информацию какого вида обрабатывает алгоритм?
— Информация какого вида получается в результате обработки?
— Что обозначают числа, сумма которых вычисляется в первом пункте?
— Будет ли алгоритм работать для квадратов?
— Будет ли работать алгоритм одинаково хорошо для зеленых и красных прямоугольников? Почему?
— Будет ли этот алгоритм работать для такой задачи: “Известно число помидоров, необходимое для заполнения наполовину первого ящика, и число помидоров, необходимое для заполнения наполовину второго ящика. Найти число помидоров, которое необходимо для заполнения доверху двух ящиков вместе”?
— Объясните на примере алгоритма Ромы, почему для информатики не важен смысл информации, с которой она работает.
19. Валя подсчитывает удвоенное число букв “а” в сказке про Красную Шапочку по следующему алгоритму:
1) Подсчитай число букв “а” в тексте.
2) Умножь результат на два.
Ответьте на следующие вопросы:
— Информацию какого вида обрабатывает алгоритм?
— Информация какого вида получается в результате обработки?
— Нужен ли другой алгоритм для подсчета удвоенного числа букв “а” для сказки про Чиполлино?
— Зависит ли результат вычислений от исходного текста?
— Объясните на примере алгоритма Вали, почему для информатики не важен смысл информации, с которой она работает.

Задания на дом

Вариант 1

1. Перечислите органы и чувства, которые человек и животные используют для получения информации. Приведите примеры и запишите их в таблицу:

Чувство	Орган чувства	Пример информации

2. Какие виды информации являются главными для человека? Если человек лишен возможности получать информацию определенного вида, то может ли он получать ту же информацию в другой форме?

Вариант 2

1. С какими видами информации умеет работать ваш домашний компьютер?
2. Какой из видов информации можно считать главным для компьютера?
3. В каком виде представлена информация в любой компьютерной игре?
4. Какие устройства компьютера можно назвать его “органами чувств”?

Вариант 3

1. Запишите историю, убедительно свидетельствующую о том, какие виды информации являются главными для животных.

2. Представьте ту же самую историю с помощью графической информации.

3. Как вы думаете, существует ли у человека шестое чувство, которое позволяет ему читать мысли других людей, предсказывать будущее?

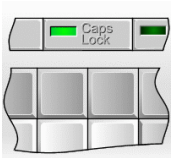
Запишите свое мнение, приведите высказывания на эту тему из книг, газет, журналов, других информационных источников.

Практикум

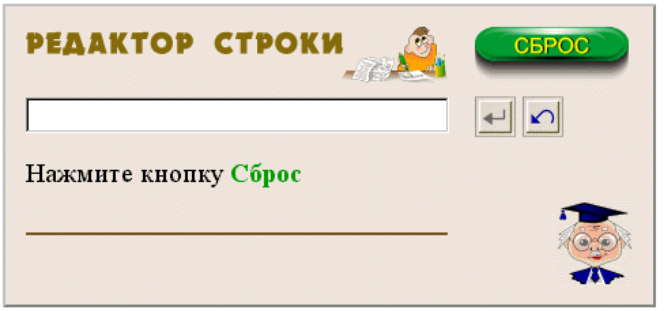
Продолжаем осваивать работу с Редактором строки.

1. Заглавные буквы

Эта клавиша (она называется "капс лок") переключает режим "прописные/строчные" для букв, которые набираются с клавиатуры.



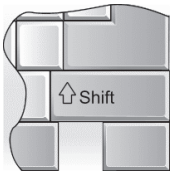
С клавишей Caps Lock связан световой индикатор справа вверху на клавиатуре. Если лампочка горит, будут вводиться заглавные буквы, если нет — строчные.



Задания исполнителя: наберите указанное слово.

1) МИР;
2) НЕБО;
3) КОМПЬЮТЕР;
4) КЛАВИАТУРА;
5) РОБОТЛАНДИЯ.

Клавиша Caps Lock удобна, когда нужно вводить много заглавных букв подряд.



Когда заглавная буква нужна только одна, лучше пользоваться клавишей Shift (читается — "шифт", в переводе — "сдвиг", "перемена").

Действие этой клавиши состоит в том, что она меняет регистр: переключает буквы из нижнего регистра (набора малых букв) в верхний (набор заглавных букв) и обратно. Поэтому эту клавишу часто так и называют — клавиша регистра.

На клавиатуре таких клавиш две. Одна из них находится слева от пробела (самой длинной клавиши), другая справа. Действуют обе клавиши совершенно одинаково. А повторены они для того, чтобы обеспечить удобство человеку: иногда удобнее нажать левую клавишу регистра, иногда — правую.

Переключение регистра клавишей Shift выполняется только на то время, которое эта клавиша удерживается нажатой. Отпустили Shift, и регистр вернулся в прежнее состояние.

Для записи имени ослика Иа нужно нажать клавишу Shift и, не отпуская ее, клавишу с буквой "и". Затем Shift отпускается и нажимается клавиша с буквой "а".

Задания исполнителя: наберите указанные слова.

1) Иа;
2) лиса Алиса;
3) кот Базилио;
4) Маленький Мук;
5) Василий Александрович Кук.

2. Удаление символов

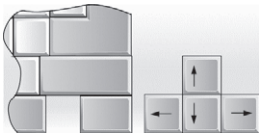
Удалять последний набранный символ удобно клавишей обратного пробела BS. Когда нужно удалить символ в середине набора, используйте клавишу Del.

Удаление символа в позиции курсора (справа от курсора) выполняется клавишей Del. На некоторых клавиатурах написано "Delete". Эта клавиша называется "клавишей удаления". Слово *delete* читается "дилит", переводится с английского "стирать".

Клавиша Del удаляет символ в позиции курсора. А как подвести курсор к нужному символу? Для этого используют клавиши с нарисованными стрелками (вправо и влево).

Задания исполнителя: удалите лишние символы.

1) эккран;
2) иерраррхия;
3) клаввиатурра;
4) инфор матика;
5) р о б о т л а н д и я.



Вопросы по теме практикума

1. Сколько нажатий на клавиши нужно выполнить для набора слова "ЛиСа", если использовать клавишу Caps Lock для переключения регистра?

2. Сколько нажатий на клавиши нужно выполнить для набора слова "ЛиСа", если использовать клавишу Shift для переключения регистра?

3. Сколько разных клавиш будет нажато при наборе слова "Молоко"?

4. Сколько нажатий на клавиши нужно выполнить для удаления лишнего символа в слове "перегнос"?

5. Сколько нажатий на клавиши нужно выполнить для удаления лишних символов в слове "стттол"?

Зачетный класс

1. Сколько органов чувств у человека?

2. Отметьте те пункты, в которых человек получает информацию:

— пробует суп;
— слушает радио;
— трогает воду;
— смотрит на фотографию;
— нюхает цветок.

3. Укажите названия чувств, посредством которых человек воспринимает информацию.

а) Петя прочитал сказку.
б) Ира лизнула мороженое.
в) Ваня обжегся горячим утюгом.
г) Чай был таким ароматным!
д) За стеной протопали лошади.

4. Укажите вид информации.

а) ***Два умножить на два равно четырем.***
б) ***1, 1, 2, 3, 5, 8, 13, 21***
в) ***Ту-154***
г) ***За стеной протопали пять лошадей.***
д) 

5. Нахождение периметра прямоугольника Рома выполняет по следующему алгоритму:

1) Сложи два заданных числа.
2) Умножь результат на два.

Запишите результаты выполнения этого алгоритма над следующими парами чисел:

1) 3, 8;
2) 10, 10;
3) 1, 2.

6. Валя работает по следующему алгоритму:

1) Подсчитай число букв "а" в тексте.
2) Умножь результат на два.

Запишите результаты выполнения этого алгоритма над следующими текстами:

1) мама;
2) крокодил Гена;
3) Дядя Федор.

Книга для учителя

Информатика — это абстрактная наука

Информатика — это абстрактная наука, такая же, как математика. В этом смысле информатика в "чистом виде" не имеет отношения к реальной жизни.

Так же, как математика не изучает исчисление рублей, кроликов и груш, так и информатика не изучает обработку ведомостей на зарплату, частотный анализ слов в книге "Золотой теленок" и способы работы с текстовым процессором Word. Математика изучает общие правила работы с числами, а информатика — общие правила работы с информацией.

При этом для математики все равно, что стоит за числами (рубли, кролики или груши), а для информатики безразличен смысл обрабатываемой информации. Если информатика создает алгоритм упорядочивания символов текста по возрастанию их порядковых номеров в кодовой таблице, то этот алгоритм будет работать и на числовой, и на текстовой информации, и для кодовой таблицы Windows, и для кодовой таблицы DOS.

Для информатики не важен смысл информации. Значит, эта наука изучает правила работы с бессмысленными сообщениями?

Ответ положительный. Математика изучает бессмысленные числа. А смысл в них вкладывает не математика, а прикладные науки, связанные с математикой: бухгалтерский учет, инженерное проектирование...

Информатика изучает бессмысленные сообщения, а смысл в них вкладывают прикладные науки, связанные с обработкой информации: лингвистика, криминалистика, журналистика... — практически все области человеческой деятельности.

Одним словом, информатика, как и математика, абстрактная наука. В Читальном зале вместо трудного слова "абстрактная" использовано более простое — "универсальная".

Как получить информацию

Информация поступает к человеку (и животному) через его органы чувств.

Несмотря на попытки некоторых ученых обосновать наличие чувства, связанного с биоэнергетикой (шестого чувства), оно пока еще всерьез не рассматривается наукой. Поэтому считается, что чувств пять: зрение, слух, вкус, осязание и обоняние. Таким образом, можно говорить об информации, поступающей через изображение, звук, запах, осязательные и вкусовые ощущения.

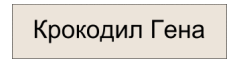
Важной классификацией является разделение визуальной информации на числовую, текстовую и графическую. Эти виды информации по-разному представляются на компьютерных информационных носителях, а для их обработки используются существенно отличные алгоритмы. Именно поэтому в Читальном зале вводится такое пропедевтическое разделение.

Однако упражнения, которые предлагаются для закрепления этой темы, содержат определенные условности, допустимые для первого знакомства с предметом младшего школьника.

Приведу примеры.

Крокодил Гена

Это текстовая информация. Она записана как текст в компьютерной книге.



Это графическая информация. Она задана картинкой на компьютерной странице.

Первый "крокодил" — текст. Он занимает 13 байтов в компьютерной памяти. Второй — картинка. Она хранится на диске под именем GENA.GIF и занимает 1940 байт. Визуально оба эти объекта почти неотличимы.

Является ли следующая запись числовой информацией:

1, 1, 2, 3, 5, 8, 13, 21

Для человека — да, для компьютера — нет. Компьютер прочитает эту запись как текст, затем, если это предусмотрено программой обработки, выделит числа и запишет их в ячейках памяти так, как это принято для компьютерной записи чисел.

Ответы на вопросы

1. Приведите примеры текстовой информации.

"Один мельник в наследство своим трем сыновьям оставил всего-навсего мельницу, осла и кота".

2. Приведите примеры числовой информации.

1, 1, 2, 3, 5, 8, 13, 21, 34, 55

3. Приведите примеры графической информации.

4. Приведите примеры смешанной информации.

Четные числа — это такие числа, которые делятся на 2. Например, числа 1024 и 5462 — четные.

5. Назовите 5 органов чувств человека, посредством которых он воспринимает информацию.

Чувство	Орган чувства
зрение	глаза
слух	уши
обоняние	нос
вкус	язык
осязание	кожа

6. Какой вид информации человек воспринимает глазами?

Зрительную (визуальную). К этой информации относятся тексты, картинки и числа.

7. Какой вид информации человек воспринимает ушами?

Звуковую информацию. Сообщения по радио, человеческая речь, пение, голоса животных, шелест ветра, журчание ручейка, другие звуки природы.

8. Какой вид информации человек воспринимает носом?

Информацию, которую содержат запахи. Приятен запах обеда, который готовит мама. Запах дыма настораживает: не горит ли что-нибудь на кухне? Запах кофе бодрит. Запах резиновых сапог напоминает о рыбалке, а запах сена — о бабушкиной корове Зорьке в деревне.

9. Какой вид информации человек воспринимает языком?

Вкусовую информацию. Мы легко можем определить на вкус сахар и соль, хотя на вид они очень похожи. Горький гриб мы не станем есть — скорее всего он ядовитый. А вот сладкая малина нам по душе! Лимон хоть и кислый, но с сахаром очень вкусный. А перец бывает горьким — его используют как приправу. Хороший шоколад тоже горчит, но он очень вкусный.

10. Какой вид информации человек воспринимает кожей?

Осязательную информацию. Она поступает к нам при прикосновении к предметам. Снег — холодный, утюг — горячий, диван — мягкий, табуретка — жесткая. Ножик — острый, а если нет, надо поточить. Земля сухая — надо срочно полить редиску. Глина мокрая, но недостаточно, надо добавить воды. С закрытыми глазами трогаем мяч и понимаем, что он круглый. Какого цвета, не знаем, но круглый — это точно!

11. Приведите примеры, когда одно и то же чувство восприятия информации для животных дает больше сведений, чем для человека.

У собаки не очень хорошее зрение. Часто бросается она в погоню за клочком бумаги, думая, что это птичка. Но вот слух и обоняние у нее отличные! Посетитель только подошел к вашей двери, не нажал еще даже на кнопку звонка, а собака услышала и бросается в прихожую с громким лаем. Хорошо известно, как собаки держат след, ориентируясь на запах.

А уж конфетку в школьном портфеле она учует обязательно и... прощай припрятанная сладость! Для человека слух и обоняние стоят на втором месте после зрения, а у собаки — на первом.

12. Укажите, в каком виде представлена информация в следующих примерах:

— примеры по математике;
— географическая карта;
— письмо;
— картина;
— радиопередача;
— телевизионная передача;
— ароматный запах, желтый цвет и кислый вкус лимона;
— этот учебник.

— Примеры по математике — это числовая информация, а чаще — смесь чисел и текста (а иногда и рисунков).

— Географическая карта — смесь графической, текстовой и числовой информации.

— Письмо, как правило, содержит текст, но могут быть числа и рисунки.

— Картина — это графическая информация. Даже подпись художника и дата создания выполнены художественным образом. Хотя, конечно, можно из картины выделить тексты и числа и рассматривать их как отдельный вид информации. Картина рассказывает о красотах осеннего леса, подпись сообщает имя художника, а число — год создания произведения.

— Радиопередача — это чисто звуковая информация. Героев передачи нельзя увидеть, потрогать, понюхать, лизнуть.

— Телевизионная передача — это изображение плюс звук, то есть смесь зрительной и звуковой информации.

— Ароматный запах, желтый цвет и кислый вкус лимона — включаются нос, глаза и язык! Мы получаем представление о фрукте через информацию запаха, зрения и вкуса!

— Этот учебник — здесь есть тексты, картинки и числа, то есть он воспринимается через глаза (визуальная информация). Компьютерный учебник не пахнет, его нельзя потрогать (он нарисован на экране), попробовать на вкус, и он молчит.

13. Что изучает информатика?

Информатика изучает способы хранения, передачи и обработки информации.

14. Почему информатика является универсальной наукой?

Информатика изучает общие законы работы с информацией, независимо от конкретного содержания. Алгоритмы информатики выводятся один раз, в общем виде, а потом используются для множества частных случаев.

Например, существует алгоритм нахождения минимального элемента во множестве элементов. Если такой алгоритм будет работать с числами:

99999 231 249 446 121 227 165 371 272 2866

— он выдаст результатом число 121.

Алгоритм будет правильно работать с любыми числами, независимо от их величины, количества и значения (рубли, кролики, груши). Более того, алгоритм будет работать и с буквами. Для такого набора:

у з п р о п ы ы ф р о л д

— он выдаст результатом букву "д".

Этот алгоритм будет работать и со словами. Для такого набора:

Петя Оля Саша Юра Лида Юля Рома Таня Лина Сережа

— он выдаст результатом слово "Лида".

15. Можно ли назвать универсальной науку, которая учит считать кроликов?

Считать кроликов, конечно, очень полезно, но лучше изучить математику. Тогда можно считать все что угодно, а не только ушастых!

16. Можно ли назвать универсальной науку, которая изучает алгоритмы обработки текста сказки "Красная Шапочка"?

Ученый муж такой "науки" похож на человека, который сверлит железный лист деревянной палочкой, имея под рукой электрическое сверло.

17. Важен ли для человека смысл информации?

Что за вопрос! Конечно, важен. Мы способны жить только потому, что умеем получать из мира информацию и осознавать ее смысл. Другое дело, когда информации очень много или она очень сложная. Тогда мы заставляем работать на себя компьютер, который, не вникая в смысл, делает работу по общим законам информатики для нашего конкретного случая.

18. Нахождение периметра прямоугольника Рома выполняет по следующему алгоритму:

1) Сложи два заданных числа.
2) Умножь результат на два.

Ответьте на следующие вопросы:

— Информацию какого вида обрабатывает алгоритм?

— Информация какого вида получается в результате обработки?

— Что обозначают числа, сумма которых вычисляется в первом пункте?

— Будет ли алгоритм работать для квадратов?

— Будет ли работать алгоритм одинаково хорошо для зеленых и красных прямоугольников? Почему?

— Будет ли этот алгоритм работать для такой задачи:

"Известно число помидоров, необходимое для заполнения наполовину первого ящика, и число помидоров, необходимое для заполнения наполовину второго ящика. Найти число помидоров, которое необходимо для заполнения доверху двух ящиков вместе"?

— Объясните на примере алгоритма Ромы, почему для информатики не важен смысл информации, с которой она работает.

Ответы.

— Алгоритм обрабатывает числовую информацию.

— В результате обработки получается число.

— Числа, сумма которых вычисляется в первом пункте для данного конкретного вычисления, обозначают ширину и высоту прямоугольника.

— Алгоритм работает для любых прямоугольников, в том числе и для квадратов (прямоугольников с равными сторонами).

— Алгоритм не берет в расчет цвет прямоугольника. Он вообще "не знает", что вычисляет периметр прямоугольника. Он просто работает с числами, "не вникая" в то, что эти числа обозначают.

— Этот алгоритм будет работать и для такой задачи. Ведь числовой расчет остается тем же самым.

— Для алгоритма все равно, что обозначают числа, с которыми он работает, — стороны прямоугольника или помидоры. Информатика — универсальная наука, и ее алгоритмы универсальны.

19. Валя подсчитывает удвоенное число букв "а" в сказке про Красную Шапочку по следующему алгоритму:

1) Подсчитай число букв "а" в тексте.
2) Умножь результат на два.

Ответьте на следующие вопросы:

— Информацию какого вида обрабатывает алгоритм?

— Информация какого вида получается в результате обработки?

— Нужен ли другой алгоритм для подсчета удвоенного числа букв "а" для сказки про Чиполлино?

— Зависит ли результат вычислений от исходного текста?

— Объясните на примере алгоритма Вали, почему для информатики не важен смысл информации, с которой она работает.

Ответы.

— Алгоритм обрабатывает текстовую информацию.

— В результате обработки получается числовая информация.

— Для подсчета удвоенного числа букв “а” в сказке про Чиполлино другой алгоритм не нужен: будет прекрасно работать тот же самый.

— Результат вычислений, конечно, зависит от исходного текста. Для Красной Шапочки и Чиполлино получатся разные ответы.

— Для информатики не важен смысл информации, с которой она работает, потому что это позволяет строить универсальные алгоритмы.

Ответы на вопросы практикума

1. Сколько нажатий на клавиши нужно выполнить для набора слова “ЛиСа”, если использовать клавишу Caps Lock для переключения регистра?

Придется нажать клавиши 8 раз: четыре клавиши с буквами и четыре раза клавишу Caps Lock.

2. Сколько нажатий на клавиши нужно выполнить для набора слова “ЛиСа”, если использовать клавишу Shift для переключения регистра?

Клавиши придется нажимать 6 раз: четыре клавиши с буквами и два раза клавишу Shift.

3. Сколько разных клавиш будет нажато при наборе слова “Молоко”?

Пять разных клавиш: клавиши с буквами “м”, “о”, “л”, “к” и клавиша Shift.

4. Сколько нажатий на клавиши нужно выполнить для удаления лишнего символа в слове “перегнос”?

Это зависит от положения текстового курсора. Если курсор расположен сразу за текстом, то 5 раз: четыре раза нужно нажать клавишу со стрелкой влево и один раз клавишу Del.

5. Сколько нажатий на клавиши нужно выполнить для удаления лишних символов в слове “сттол”?

Это зависит от положения текстового курсора и алгоритма удаления.

Пусть текстовый курсор расположен сразу за текстом. Тогда самое короткое удаление лишних символов можно выполнить за 6 нажатий: сначала 4 раза влево, затем два раза Del.

Решения зачетного класса

1. Сколько органов чувств у человека?

5.

2. Отметьте те пункты, в которых человек получает информацию:

— пробует суп; — смотрит на фотографию;
— слушает радио; — нюхает цветок.
— трогает воду;

Человек получает информацию во всех описанных случаях.

3. Выберите названия чувств, посредством которых человек воспринимает информацию.

а) Петя прочитал сказку.	Зрение
б) Ира лизнула мороженое.	Вкус
в) Ваня обжегся горячим утюгом.	Осязание
г) Чай был таким ароматным!	Обоняние
д) За стеной протопали лошади.	Слух

4. Укажите вид информации.

а) ***Два умножить на два равно четырем.***	Текстовая
б) ***1, 1, 2, 3, 5, 8, 13, 21***	Числовая
в) ***Ту-154***	Смешанная
г) ***За стеной протопали пять лошадей.***	Текстовая
д) 	Графическая

5. Нахождение периметра прямоугольника Рома выполняет по следующему алгоритму:

1) Сложи два заданных числа.
2) Умножь результат на два.

Запишите результаты выполнения этого алгоритма над следующими парами чисел:

3, 8	22
10, 10	40
1, 2	6

6. Валя работает по следующему алгоритму:

1) Подсчитай число букв “а” в тексте.
2) Умножь результат на два.

Запишите результаты выполнения этого алгоритма над следующими текстами:

мама	**4**
крокодил Гена	**2**
Дядя Федор	**0**

Продолжение в следующем номере

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 30.01.2002.
Номер подписан 30.01.2002.
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.