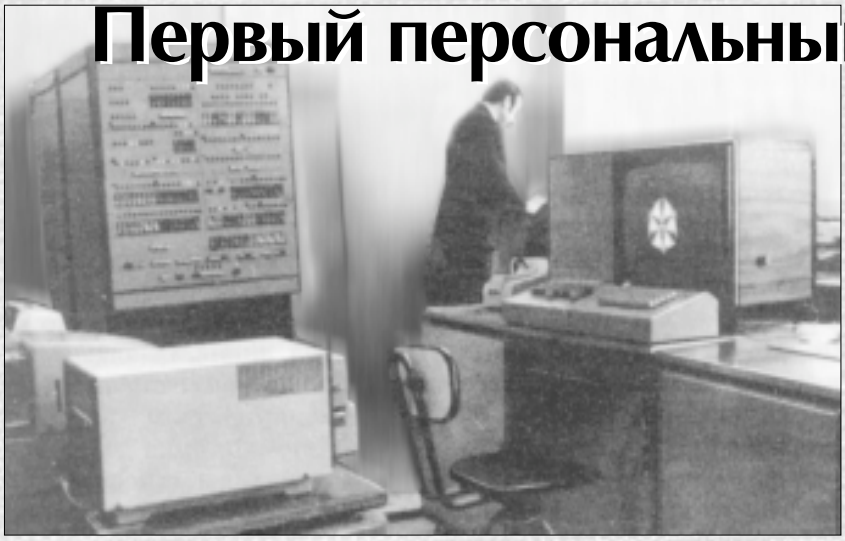
№ 37 (230) октябрь 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Первый персональный компьютер



В 1969 году, 30 лет назад, появилась электронная вычислительная машина “МИР-2”, в которой впервые использовались дисплей со световым пером и память на магнитных картах. (Первая персональная ЭВМ — “МИР” была создана в СССР в 1965 году, на 10 лет раньше, чем в США.)

Об истории создания ЭВМ серии “МИР” читайте на стр. 32

Читайте в номере

Информатика в этюдах

В.М. Казиев

Продолжение. Начало в № 35/99

2. Системы счисления

Этюды с решениями

1. Вычислить наибольшее и наименьшее 5-разрядное целое число в системе счисления с основанием 4.

Решение. Наибольшее целое n-разрядное число в системе счисления с основанием p равно:

$$x_p^{max} = \sum_{i=0}^{n-1}(p-1)p^i = \sum_{i=0}^{n-1}p^{i+1} - \sum_{i=0}^{n-1}p^i = p^n - 1.$$

Наименьшее целое n-разрядное число в этой системе равно:

$$x_p^{min} = -x_p^{max} = 1 - p^n.$$

Например, для двоичной системы

$$x_2^{max} = \underbrace{1\ 1\ 1\ \ldots\ 1\ 1\ 1}_{n\ \text{разрядов}}{}_2 = (2^n - 1)_{10},$$

$$x_2^{min} = (1 - 2^n)_{10},$$

а в восьмеричной системе счисления эти числа определяются в виде:

$$x_8^{max} = \underbrace{7\ 7\ 7\ \ldots\ 7\ 7\ 7}_{n\ \text{разрядов}}{}_8 = (8^n - 1)_{10},$$

$$x_8^{min} = (1 - 8^n)_{10}.$$

Таким образом, в системе счисления с основанием 4 в пяти разрядах могут быть представлены числа из следующего диапазона:

$$-1023 = 1 - 4^5 \le (x)_4 \le 4^5 - 1 = 1023.$$

2. Привести пример того, что любой набор двоичных цифр есть изображение числа в десятичной системе, но при этом не любой набор цифр в двоично-шестнадцатеричной системе есть представление десятичного числа.

Решение. Например,

$$11010110_2 = 2^7 + 2^6 + 2^4 + 2^2 + 2^1 =$$
$$= 128 + 64 + 16 + 4 + 2 = 214_{10},$$

но при переводе двоично-шестнадцатеричной комбинации 11010110 ($D6$) получаем цифру 13 (D), которой нет в десятичной системе.

3. Найти число x, записанное в системе счисления с основанием p, если известно, что оно совпадает со своим обратным кодом (т.е. с числом, получаемым при дополнении каждой цифры данного числа до максимально возможной цифры в данной системе счисления). При каких p задача имеет решение?

Решение. Число в системе счисления с основанием p имеет вид:

$$x = x_n x_{n-1} \ldots x_1 x_0 = x_n \cdot p^n + x_{n-1} \cdot p^{n-1} + \ldots + x_0.$$

Обратный код числа y имеет вид:

$$y = y_n y_{n-1} \ldots y_1 y_0,$$
$$y_i = p - 1 - x_i,$$
$$i = 0, 1, \ldots, n.$$

Так как по условию задачи $x = y$, то $y_i = x_i$, $i = 0, 1, \ldots, n$. Отсюда получаем: $x_i = p - 1 - x_i$ или $x_i = (p-1)/2$, $i = 0, 1, \ldots, n$. Из последней системы равенств видно, что решение задачи возможно лишь в системах счисления с нечетным основанием (в противном случае x_i — не целое). У полученного числа в любой системе счисления с нечетным основанием все цифры прямого и обратного кода совпадают.

4. Доказать, что $\mathrm{mod}(x, 2) = 0$, если число $x = 156$ рассматривается в системе счисления с основанием p, $11 < p < 36$.

Решение. Так как

$$x = p^2 + 5p + 6 = (p + 2)(p + 3),$$

то один из этих двух последовательных множителей является четным, т.е. $\mathrm{mod}(x, 2) = 0$.

Темы для этюдов

1. Имеются ящики: 4 черных, 3 красных, 2 желтых и 1 зеленый. В каждом черном ящике — $(21)_p$ шара, красном — $(23)_p$ шара, желтом — $(23)_p$ шара, зеленом — $(111)_p$ шара. Определить основание p системы счисления, в которой были посчитаны шары, если всего было $(244)_p$ шаров. Чему равно общее число шаров в восьмеричной системе?

2. Числа $A = (100)_x$ и $B = (411)_y$ заданы, соответственно, в системах счисления с основаниями x и y ($0 < x, y < 17$). Найти x и y, если известно, что в десятичной системе счисления x в 2 раза

больше y, а если от B отнять A, то получается десятичное число 7. Найти A в шестнадцатеричной системе счисления.

3. Число $x = (176)_p$ (рассматриваемое в системе счисления с основанием p, $1 < p < 20$) делится нацело на 7. Найти p (не перебирая все значения из указанного диапазона). Вычислить x в восьмеричной системе.

4. Какова разрядность двоичной системы счисления, в которой представим лишь диапазон чисел из интервала $(-255; 255)$?

5. Найти числа, соответствующие математическим ("арифметическим") понятиям "*нуль*" и "*бесконечность*" в n-разрядной системе счисления, и сравнить их. Указать эти числа для двоичной, восьмеричной и шестнадцатеричной систем.

6. Сформулировать какие-то признаки делимости на 8 (на 11) числа x, записанного в системе счисления с основанием $p = 12$, не переводя эти числа в десятичную систему. Рассмотреть другое значение p.

7. Найти число x, записанное в системе счисления с основанием p, если оно совпадает со своим дополнительным кодом. При каких p это возможно?

8. В факториальной системе счисления целые числа записывают как линейную комбинацию факториалов, например, число 2457 в этой системе:

$$2457_! = 2 \cdot 3! + 4 \cdot 2! + 5 \cdot 1! + 7 \cdot 0!$$

$(0! = 1,\ n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n)$

Позиционна ли эта система и почему?

3. Алгебра высказываний и предикатов

Этюды с решениями

1. Заданы два предиката вида p = "число x делится нацело на 5" и q = "y — день недели". Найти множество истинности предикатов p и q, если $x \in \{16, 20, 26, 30, 35\}$, $y \in \{$*первый, вторник, среда,* 1999, *выходной, праздник, воскресенье*$\}$.

Решение

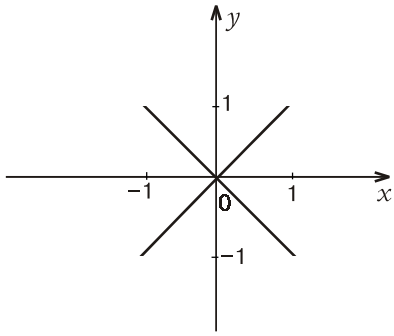
$D(p) = \{20, 30, 35\}$, $D(q) = \{$*вторник, среда, воскресенье*$\}$.

2. Изобразить графически множество истинности двухместного предиката вида $p(x, y) =$ "*модуль числа x равен модулю числа y*", если задана область изменения аргументов: $x, y \in [-1; 1]$.

Решение

$$E(p) = \{(x, y) : |x| = |y|\} =$$
$$= \{(x, y) : (x = y) \vee (x = -y)\} =$$
$$= \{(x, y) : x = y\} \cup \{(x, y) : x = -y\} =$$
$$= E(p_1) \cup E(p_2).$$

Смысл предикатов $p_1(x, y)$ и $p_2(x, y)$ очевиден.



3. Можно ли функции $z = \bar{x} \vee \overline{x \vee y} \vee y$, $w = x \vee \overline{y \vee x} \vee \bar{y}$ реализовать одним и тем же наименьшим числом логических схем (вентилей) и почему? Ответ обосновать.

Решение. Используя аксиомы де Моргана и поглощения, получаем:

$$z = \bar{x} \vee \overline{x \vee y} \vee y = \bar{x} \vee \bar{x} \wedge \bar{y} \vee y = \bar{x} \vee y,$$
$$w = x \vee \overline{y \vee x} \vee \bar{y} = x \vee \bar{y} \wedge \bar{x} \vee \bar{y} = x \vee \bar{y}.$$

Следовательно, обе функции реализуются двумя одинаковыми вентилями: одним инвертором и одним дизъюнктором.

4. Упростить, используя как можно меньше аксиом алгебры предикатов и за как можно меньшее число их применений:

$$z = \overline{x \vee y} \vee \bar{x} \wedge y \vee \bar{x}.$$

Решение. Используем аксиому поглощения: $x \wedge y \vee x = x$. Применяя ее к последним трем операндам, получим: $\bar{x} \wedge y \vee \bar{x} = \bar{x}$. Итак, $z = \overline{x \vee y} \vee \bar{x}$. По аксиоме де Моргана, $z = \bar{x} \wedge \bar{y} \vee \bar{x}$. По аксиоме поглощения, получим:

$$z = \bar{x} \wedge \bar{y} \vee \bar{x} = \bar{x}.$$

Использованы 2 аксиомы (поглощения — 2 раза и де Моргана — 1 раз).

5. Упростить, используя минимум аксиом алгебры предикатов:

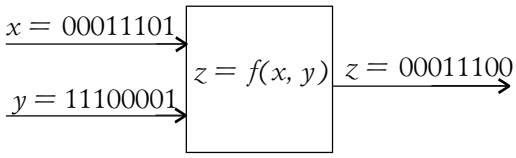
$$z = \bar{x} \vee \overline{x \vee y} \vee \overline{y \wedge \overline{x \wedge y}}.$$

Решение. Над знаком равенства в кружочке будем записывать аксиому, с помощью которой из выражения слева получено выражение справа:

$$z = \bar{x} \vee \overline{x \vee y} \vee y \wedge \overline{\overline{x \wedge y}} \overset{①}{=} \bar{x} \vee \overline{x \vee y} \vee \bar{y} \vee \overline{\overline{x \wedge y}} \overset{②}{=}$$
$$= \bar{x} \vee \overline{x \vee y} \vee \bar{y} \vee x \wedge y \overset{①}{=} \overline{x \wedge y} \vee x \wedge y \vee \overline{x \vee y} = 1.$$

① аксиома де Моргана
② аксиома инволюции

6. Дан "черный ящик" с двумя входами и одним выходом. На входы подаются логические константы 0 и 1, на выходе так же получается логическое значение. По данным последовательностям исходных данных x, y и результата z найти подходящую логическую функцию для "черного ящика"



Решение. Из анализа последовательностей x, y и z видно, что подходит, например, функция $z = x \wedge \bar{y}$.

Темы для этюдов

1. Определить множество истинности предиката p, заданного на множестве X:
$p(x) =$ "*число* x *кратно* 4", $X = [2; 10]$.

2. Составить таблицу истинности и затем упростить функцию:

$$z = (\overline{x \vee y} \vee x \wedge y) \wedge \bar{x} \wedge y.$$

3. Обсуждая свои возможности по поступлению в вуз, абитуриенты *А*, *Б*, *В* высказали предположения: *А* — "Я не смогу поступить, а *В* — поступит"; *Б* — "*В* не поступит, а *А* — поступит"; *В* — "Или я не поступлю, или *В* не поступит". После сдачи экзаменов выяснилось, что каждый высказал одно верное и одно ложное простое утверждение. Кто поступил в вуз, если не поступил только один из них?

4. Построить логическое выражение, определяемое таблицей истинности:

x	y	?	?	?	?	?
0	0	1	1	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	1	0	1	1	1	0

5. Подобрать три функции, эквивалентные данной, но более простого вида (с меньшим числом операций и операндов): $z = \overline{x \wedge y} \vee x \vee y \wedge \overline{y \vee x}$.

6. Из указанных ниже функций отметить (с обоснованием всех своих рассуждений) эквивалентные:

а) $z = x \vee \overline{x \vee y} \vee y \vee \overline{x \wedge y \vee x \vee y} \vee y \vee x \wedge y$;

б) $u = x \vee \overline{x \vee y} \vee \overline{y \vee x} \wedge y$;

в) $w = x \vee \overline{x \vee y} \vee y \vee \overline{x \wedge y}$;

г) $s = \overline{x \vee \overline{x \vee y}} \vee \overline{y \vee \overline{x \wedge y}}$;

д) $s = x \vee \overline{y \vee x \wedge y} \vee y \vee \overline{x \wedge y}$.

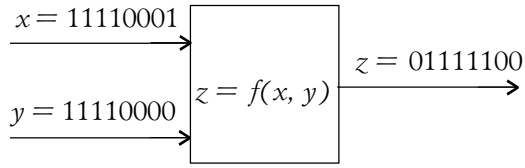
7. Упростить логическое выражение z, затем построить таблицу истинности и указать хотя бы одну функцию, эквивалентную данной:

1) $z = \overline{x \wedge y} \wedge y \vee \overline{x \vee y} \wedge y \vee y$;

2) $z = \overline{x \wedge y} \wedge \overline{x \vee y} \vee \bar{x}$;

3) $z = x \wedge \overline{y \wedge \overline{x \vee y}} \vee \bar{y}$.

8. Найти подходящую логическую функцию для "черного ящика" по входным (x, y) и выходной (z) последовательностям.



9. Построить логические схемы, соответствующие функциям:

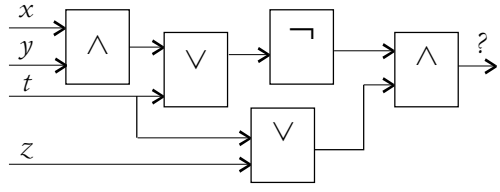
а) $z = \overline{x \wedge y} \vee t \wedge \bar{x} \wedge y \vee t$;

б) $z = \bar{x} \vee t \wedge \bar{x}$;

в) $z = x \vee \overline{y \wedge x} \vee \overline{t \vee x} \wedge y$.

Построить эквивалентные логические схемы с минимумом инверторов, конъюнкторов, дизъюнкторов. Покажите вычислениями, что построенная схема оптимальна по сравнению с эквивалентными ей по входу-выходу.

10. Определить логическую функцию, реализуемую схемой:



Продолжение следует

Элементы математической логики в курсе школьной информатики

Л.Л. Акуленко-Босова

Продолжение. Начало в № 35/99

4. Законы логики высказываний

Сложные высказывания (формулы) A и B называются **равносильными**, если их истинностные значения совпадают для любых наборов истинностных значений простейших высказываний, входящих в эти формулы.

В алгебре логики имеется ряд законов, позволяющих производить равносильные преобразования формул. Приведем соотношения, отражающие эти законы.

1. Закон двойного отрицания:

$$\overline{\overline{A}} = A$$

2. Переместительный (коммутативный) закон:

— для логического сложения:

$$A \vee B = B \vee A;$$

— для логического умножения:

$$A \,\&\, B = B \,\&\, A.$$

В обычной алгебре $a + b = b + a$, $a \cdot b = b \cdot a$.

3. *Сочетательный (ассоциативный) закон:*

— для логического сложения:

$$(A \vee B) \vee C = A \vee (B \vee C);$$

— для логического умножения:

$$(A \,\&\, B) \,\&\, C = A \,\&\, (B \,\&\, C).$$

В обычной алгебре

$$(a + b) + c = a + (b + c) = a + b + c,$$

$$a \cdot (b \cdot c) = a \cdot (b \cdot c) = a \cdot b \cdot c.$$

4. *Распределительный (дистрибутивный) закон:*

— для логического сложения:

$$(A \vee B) \,\&\, C = (A \,\&\, C) \vee (B \,\&\, C);$$

— для логического умножения:

$$(A \,\&\, B) \vee C = (A \vee C) \,\&\, (B \vee C).$$

В обычной алгебре

$$(a + b) \cdot c = a \cdot c + b \cdot c.$$

5. *Закон общей инверсии (законы де Моргана):*

— для логического сложения:

$$\overline{A \vee B} = \overline{A} \,\&\, \overline{B};$$

— для логического умножения:

$$\overline{A \,\&\, B} = \overline{A} \vee \overline{B}.$$

6. *Закон идемпотентности* (от латинских слов *idem* — тот же самый и *potens* — сильный; дословно — равносильный):

— для логического сложения:

$$A \vee A = A;$$

— для логического умножения:

$$A \,\&\, A = A.$$

7. *Законы исключения констант:*

— для логического сложения:

$$A \vee 1 = 1,\ A \vee 0 = A;$$

— для логического умножения:

$$A \,\&\, 1 = A,\ A \,\&\, 0 = 0.$$

8. *Закон противоречия:*

$$A \,\&\, \overline{A} = 0.$$

Невозможно, чтобы противоречивые высказывания были истинными одновременно.

9. *Закон исключенного третьего*:

$$A \vee \overline{A} = 1.$$

Из двух противоречивых высказываний одно истинно.

10. *Закон поглощения*:

— для логического сложения:

$$A \vee (A \,\&\, B) = A;$$

— для логического умножения:

$$A \,\&\, (A \vee B) = A.$$

11. *Закон исключения (склеивания):*

— для логического сложения:

$$(A \,\&\, B) \vee (A \,\&\, \overline{B}) = B;$$

— для логического умножения:

$$(A \vee B) \,\&\, (A \vee \overline{B}) = B.$$

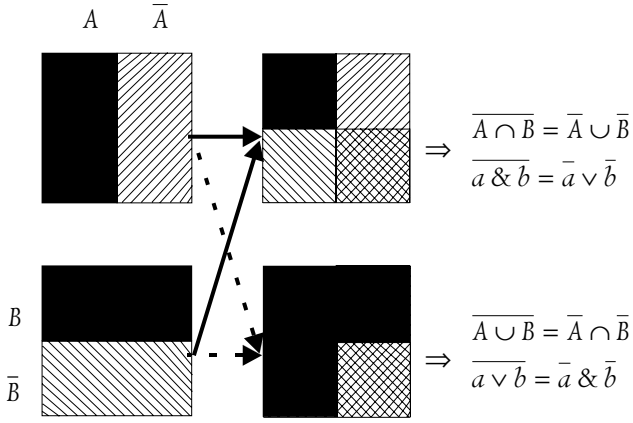
Переместительный, сочетательный (для логических сложения и умножения) и распределительный (для логического сложения) законы имеют полную аналогию с обычной алгеброй. Для других законов такой аналогии нет.

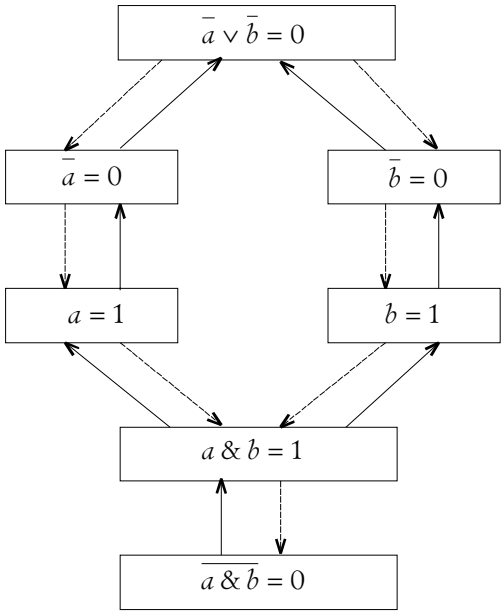
Справедливость приведенных законов можно доказать с помощью таблиц истинности: надо выписать все наборы значений A и B, вычислить для них значения левой и правой частей и убедиться, что результирующие столбцы совпадут.

Пример. Докажем справедливость закона инверсии для логического сложения:

A	B	$A \vee B$	$\overline{A \vee B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \& \overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Доказательства логических законов с помощью таблиц истинности просты, но совершенно не наглядны, поэтому ученики их не запоминают. Для лучшего усвоения доказательства законов можно иллюстрировать схемами и рисунками. Для примера приведем две схемы, иллюстрирующие законы де Моргана.





Пример. Используя дистрибутивный закон (4) и закон исключенного третьего (9), упростим формулу

$$(A \& B \& C) \vee (A \& B \& \overline{C}) =$$

$$= (A \& B) \& (C \vee \overline{C}) = (A \& B) \& 1 = A \& B.$$

Правильность результата можно проверить, составив таблицы истинности для исходной и полученной формул.

Упражнения

1. Докажите самостоятельно законы 1—11.

2. Вычислите:

а) $1 \vee X \& 0 =$; б) $X \& X \& 1 =$;

в) $0 \& X \vee 0 =$; г) $0 \vee X \& X =$.

3. Найдите X, если $\overline{(X \vee A)} \vee \overline{(X \vee \overline{A})} = B$.

Решение

Воспользуемся законом де Моргана для логического сложения и законом двойного отрицания:

$$(\overline{X} \& \overline{A}) \vee (\overline{X} \& A) = \overline{X} \& (\overline{A} \vee A) = \overline{X} \& 1 = \overline{X} = B.$$

Тогда $X = \overline{B}$.

4. Логическая формула называется **тождественно-ложной**, если она принимает значения 0 на всех наборах входящих в нее простых высказываний. Упростите следующее высказывание и покажите, что оно тождественно-ложно:

$$(A \& B \& \overline{B}) \vee (A \& \overline{A}) \vee (B \& C \& \overline{C}).$$

5. Путем преобразования докажите равносильность следующих высказываний:

$$\overline{(A \& \overline{B}) \vee (B \& \overline{C})} = (\overline{A} \& \overline{B}) \vee (\overline{A} \& C) \vee (B \& C).$$

Решение

Применив к левой части законы де Моргана и закон двойного отрицания, получим:

$$(\overline{A} \vee B) \& (\overline{B} \vee C);$$

теперь применим дистрибутивный закон для логического сложения:

$$((\overline{A} \vee B) \& \overline{B}) \vee ((\overline{A} \vee B) \& C) =$$

$$= (\overline{A} \& \overline{B}) \vee (B \& \overline{B}) \vee (\overline{A} \& C) \vee (B \& C).$$

Последнее преобразование очевидно.

6. Упростите логические формулы:

а) $(A \& B \& C) \vee (\overline{A} \& B \& C)$;

Ответ: $B \& C$;

б) $(A \vee \overline{B} \vee C) \& \overline{(A \vee B \vee C)}$;

Ответ: $\overline{A} \& \overline{B} \& \overline{C}$.

Мы рассмотрели выше конъюнкцию, дизъюнкцию и инверсию. Дополним список логических операций.

СТРОГАЯ ДИЗЪЮНКЦИЯ. В отличие от рассмотренной выше операции дизъюнкции, можно рассмотреть строгую дизъюнкцию (двойное или), которой в естественном языке соответствует связка **либо..., либо...** Суть этой операции ясна из приведенной ниже таблицы:

A	B	$A \forall B$
0	0	0
0	1	1
1	0	1
1	1	0

ИМПЛИКАЦИЯ (лат. *implicatio* — тесно связываю), или логическое следование:

— соответствует обороту **если..., то...**;

— обозначается $\Rightarrow$

Высказывание $\boldsymbol{A \Rightarrow B}$ ложно в том и только в том случае, когда условие (первое высказывание) истинно, а следствие (второе высказывание) ложно.

Таблица истинности импликации:

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

Пример. "Если завтра будет хорошая погода, то я пойду гулять". Здесь

A = {Завтра будет хорошая погода};

B = {Я пойду гулять}.

Ясно, что сказанное будет ложью лишь в том случае, если погода действительно окажется хорошей, а гулять наш герой не пойдет. Если же погода будет плохой, то независимо от того, пойдет он гулять или нет, сказанное будет истиной: обещание пойти гулять он давал лишь при условии, что погода будет хорошей.

1-я и 2-я строки таблицы истинности импликации показывают, что импликация не полностью соответствует обычному пониманию слова *следует*. Из этих строк вытекает, что заведомо $A \Rightarrow B$ истинно, если условие A ложно. Другими словами, из неверного высказывания следует все, что угодно. Например, высказывание: "Если $2 > 3$, то -1 — положительное число" — является истинным.

Первую и вторую строки таблицы истинности хорошо иллюстрирует следующий пример. Сигнал семафора (свободен — занят) должен обеспечивать безопасность движения.

Рассмотрим высказывание $(A \Rightarrow B)$ = {Если поезд прибывает на данный путь, то подается сигнал, что путь закрыт}, составленное из двух простых высказываний:

A = {Поезд прибывает на данную станцию};

B = {Подается сигнал, что путь закрыт}.

Рассмотрим все четыре сочетания истинности и ложности высказываний A и B и установим в каждом случае значение истинности высказывания $A \Rightarrow B$:

	поезд прибывает $A = 1$	поезд не прибывает $A = 0$
сигнал "закрыт" $(B = 1)$	1	1
сигнал "свободен" $(B = 0)$	0	1

Рассматриваемое сложное высказывание истинно, если:

1) поезд прибывает; сигнал "закрыт" (1 1 1);
2) поезд не прибывает; сигнал "свободен" (0 0 1);
3) поезд не прибывает; сигнал "закрыт" (0 1 1) — если поезд не прибывает, безопасен любой сигнал.

Высказывание ложно (безопасность не обеспечивается) только в том случае, если

4) поезд прибывает; сигнал "свободен" (1 0 0).

ЭКВИВАЛЕНЦИЯ (лат. *aequivalens* — равноценное), или равнозначность:

— соответствует оборотам речи **тогда и только тогда** и **в том и только в том случае**;

— обозначается $\Leftrightarrow$

Выражение $\boldsymbol{A \Leftrightarrow B}$ истинно в том и только в том случае, когда оба исходных высказывания одновременно истинны или одновременно ложны.

Таблица истинности эквиваленции:

A	B	$A \Leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

Примеры

1. Рассмотрим два высказывания:

A = {Петя выучит уроки};

B = {Пете поставят хорошую отметку}.

Их эквиваленцией является новое высказывание:

$A \Leftrightarrow B$ = {Петя выучит уроки тогда и только тогда, когда Пете поставят хорошую отметку}.

Это высказывание истинно, если:
1) Петя не выучит уроки и Пете не поставят хорошую отметку;
2) Петя выучит уроки и Пете поставят хорошую отметку.

Это высказывание ложно, если:
1) Петя выучит уроки, но Пете не поставят хорошую отметку;
2) Петя не выучит уроки, хотя Пете поставят хорошую отметку.

2. Продавец зоомагазина уверял покупателя, что купленный им попугай будет повторять каждое услышанное слово. Каково же было удивление покупателя, когда он убедился, что попугай нем как рыба. Тем не менее продавец не лгал. В каком случае такое могло быть?

Решение

Пусть A = {попугай услышал слово} и B = {попугай повторил слово}. Слова продавца запишем как $A \Rightarrow B$. Известно, что эта импликация истинна. Истинность импликации достигается на наборах (0 0), (0 1) и (1 1). Только в первом случае попугай молчит ($B = 0$), но в этом случае и $A = 0$, т.е. попугай не слышит; скорее всего попугай глух.

К рассмотренным ранее 11 законам алгебры логики, посвященным инверсии, дизъюнкции и конъюнкции, добавим

12. *Закон контрапозиции (правило перевертывания):*

$$(A \Rightarrow B) = (\overline{B} \Rightarrow \overline{A}).$$

Упражнения

1. Обратим внимание, что только один из перечисленных 12 законов посвящен импликации, а об эквиваленции не говорится вообще. Дело в том, что операции импликации и эквиваленции могут быть заменены конъюнкцией, дизъюнкцией и инверсией. Убедитесь в этом, доказав самостоятельно следующие соотношения:

а) $A \Rightarrow B = \overline{A} \vee B$;

б) $A \Leftrightarrow B = (A \& B) \vee (\overline{A} \& \overline{B})$;

Дополнение 1

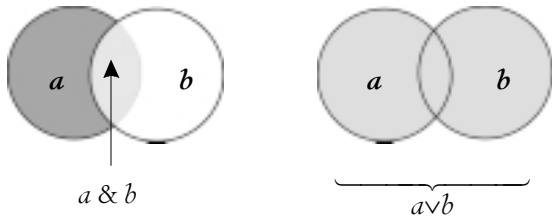
Можно установить соответствие между алгеброй логики и алгеброй множеств. Выделим некоторую высказывательную форму p и рассмотрим множество I всевозможных объектов, к которым эта высказывательная форма относится. Можно утверждать, что каждой высказывательной форме p будет отвечать некоторое подмножество P множества I, на всех элементах которого высказывательная форма p будет истинным высказыванием. Определенное таким образом P называется множеством истинности высказывательной формы p.

Пример. Рассмотрим множество I учеников некоторого класса. Известно, что Руслан и Юра — отличники. Высказывательная форма p = {он отличник} будет истинным высказыванием только по отношению к названным ученикам и ложным по отношению ко всем остальным.

Пусть A и B соответственно являются множествами истинности высказывательных форм a и b.

Тогда пересечение множеств A и B будет не чем иным, как множеством истинности $a \& b$, а объединение множеств A и B будет множеством истинности $a \vee b$.

Графическая интерпретация:



Дополнение 2

На сегодняшний день в логике не существует унифицированной символики для обозначения логических операций, поэтому полезно иметь следующую таблицу:

Обыденная речь	Символика			
	Шредера—Пирса	Пеано—Рассела	Гильберта	Лукасевича
не p	p'	$-p$	$\overline{p}$	N_p
если p, то q	$p \to q$	$p \supset q$	$p \to q$	C_{pq}
p тогда и только тогда, когда q (p если и только если q)	$p \equiv q$	$p \equiv q$	$p \sim q$	E_{pq}
p или q	$p + q$	$p \vee q$	$p \vee q$	A_{pq}
p и q	$p \cdot q$	$p \cdot q$	$p \& q$	K_{pq}

Продолжение следует

Наборы из нулей и единиц

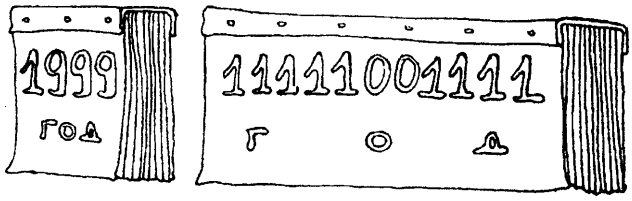
КОМПЬЮТЕРНЫЕ ИНСТРУМЕНТЫ В ОБРАЗОВАНИИ

Данная статья написана по материалам второго занятия в Заочной школе современного программирования, которая работает в Санкт-Петербурге. Объявление о приеме в школу было опубликовано в № 33/99, а материалы первого занятия — в № 35/99.

И.В. Романовский

Когда мы начинаем заниматься программированием, мы узнаем о двоичной системе счисления... А вы еще ничего не узнали? Ну, тогда несколько слов для тех, кто еще не узнал.

В двоичной системе счисления в отличие от привычной для нас десятичной всего две цифры — 0 и 1. Счет идет так: 0, 1, потом сразу 10. Потом 11, это естественно, ведь всегда $10 + 1 = 11$. Потом 100; если подумать, то тоже естественно — когда к единице в первом разряде добавляешь единицу, то получится 10, 0 пишем, а единицу переносим в старший разряд, где тоже получается 10. Потом (уже проще) 101, 110, 111, потом (уже можем догадаться) 1000, и так далее.



Числа при записи в этой системе получаются очень длинными, например, 1999 выглядит так: 11111001111. Почему? Дело в том, что в этой системе числа 10, 100, 1000 и т.д. — это степени двойки: $100 = 4$, $1000 = 8$, $10\,000 = 16$. Полученная двоичная запись выражает равенство

$$1999 = 1024 + 512 + 256 + 128 + 64 + 8 + 4 + 2 + 1.$$

Нули в записи соответствуют отсутствующим слагаемым 16 и 32.

В этой системе очень просто складывать числа, а уж прибавить к числу единицу... ну, просто ничего не стоит. Нужно просматривать запись с конца и, если встречается единица, заменить ее нулем, а если нуль, то заменить его единицей и остановить просмотр. Так что $11111001111 + 1 = 11111010000$. (Так должна выглядеть запись числа 2000. Проверьте!)

Числа, записанные в двоичной системе, удобно умножать и делить на 2, ведь два — это двоичное десять. Значит, для умножения нужно просто приписать в конце нуль, а для деления — стереть последний разряд, который должен быть нулем. Если же в конце стоит единица, то при делении получится остаток. Так что мы сразу видим, что число 11111010000 можно разделить на два без остатка четыре раза.

Наборы из нулей и единиц важны из-за того, что память компьютера состоит из мельчайших элементов (битов), каждый из которых может хранить одну двоичную цифру. Очень скромная память в 1 мегабайт* содержит больше 8 миллионов таких битов.

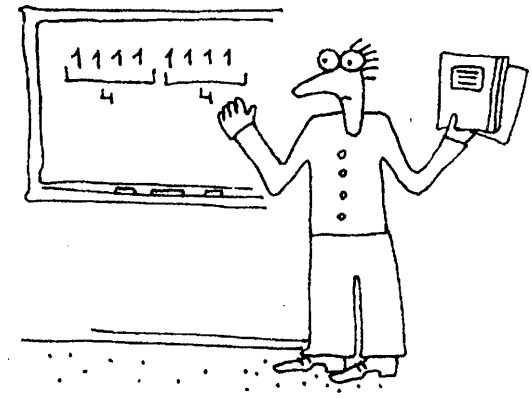
Представьте:

0100101000100101000000101111111101
0110101100111110001010000111000 1,

как в таком ужасе разбираться?

Обычно строку из битов разбивают на четверки, или иначе тетрады (не путать с тетрадями, хотя происхождение этих двух слов одно и то же):

0100 1010 0010 0101 0000 0101 1111 1101 0110
1011 0011 1110 0010 1000 0111 0001.



В свою очередь, тетрада может трактоваться как цифра в системе счисления с основанием 16 — *шест-*

* Частичка "кило" (а иногда и "мега") встречается и в обычной жизни, хотя для многих из нас работа с компьютером — это уже обычная жизнь. Вспомним километр, килокалорию и др. В обычном использовании, вы помните, кило обозначает ровно тысячу. Некоторые предлагали писать по-разному обычные и компьютерные кило. Но все решили, что всегда и так понятно, что имеется в виду. А вы понимаете? Если подержанный автомобиль продается за $2K, какая сумма имеется в виду?

Обозначение	Название	Смысл	Точное значение	Степень двойки
К	Кило	Тысяча	1024	10
М	Мега	Миллион	1048576	20
Г	Гига	Миллиард	1073741824	30
Т	Тера	Триллион	1099511627776	40

надцатеричной системе счисления. Приняты следующие обозначения для цифр в этой системе: от 0 до 9 обычные цифры, а дальше первые шесть букв латинского алфавита: *A, B, C, D, E, F*. Легко выписать соответствие между тетрадами, цифрами и числами, которые в них закодированы (табл. 1).

Таблица 1

0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	*A*	10
1011	*B*	11
1100	*C*	12
1101	*D*	13
1110	*E*	14
1111	*F*	15

Руководствуясь этой таблицей, мы можем нашу строку переписать так (проверьте):

4A2505FD6B3E2871.

При работе с памятью компьютера удобны более крупные единицы. В качестве самой маленькой единицы памяти принимается восьмерка битов, которую именуют байтом. Байт состоит из двух тетрад — старшей и младшей, так что содержимое байта записывается двумя шестнадцатеричными цифрами, как мы привыкли:

4A 25 05 FD 6B 3E 28 71.

Для решения в машине многочисленных задач хранения и переработки текстовой информации приходится вводить специальные способы кодирования букв и знаков. В языках, использующих латиницу, постепенно завоевал всемирное признание код **ASCII** — ***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*, первоначально разработанный и стандартизованный в США. В этом коде твердо фиксированы первые 128 возможностей, вторая часть может варьироваться. Вот эта вторая часть и используется для кириллицы, причем, к сожалению, общепринятой кодировки нет. Наиболее популярна у специалистов по программированию при работе в MS-DOS альтернативная кодировка, которая задается приводимой здесь таблицей* (табл. 2). Строки соответствуют старшему полубайту, столбцы — младшему. (В системе Windows используется другая кодировка.)

Важность наборов из нулей и единиц (или просто 0 — 1 наборов) связана с тем, что и в компьютерах, и в других приложениях они появляются в самых разнообразных математических моделях, и часто оказывается полезно знать и совместно использовать различные варианты их трактовки. Попробуем некоторые из них перечислить.

Таблица 2

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	/
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
9	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
B																
C																
D																
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

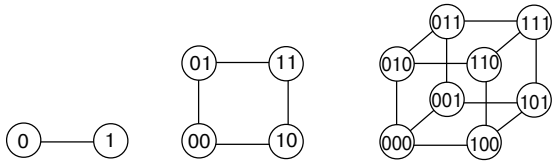


Рис. 1. Вершины единичного куба

A. Прежде всего мы можем считать, что набор из нулей и единиц — это... точка многомерного евклидова пространства, причем он определяет *одну из вершин куба*, построенного на координатных ортах (*рис.* 1). Вы знакомы с двухмерным, а некоторые и с трехмерным евклидовым пространством. Но мы можем представить себе и более сложные пространства, а единичные кубы нам в этом помогут. Когда наш набор состоит из m чисел, мы говорим об m-мерном пространстве, а сам набор называем вектором в этом пространстве.

B. Такой набор может рассматриваться и как способ задания *подмножества* множества целых чисел от 1 до m (обозначение $1 : m$). Именно, если i-й элемент набора равен 1, то число i принадлежит множеству, в противном случае не принадлежит. Так определенный набор называется *характеристическим вектором* подмножества.

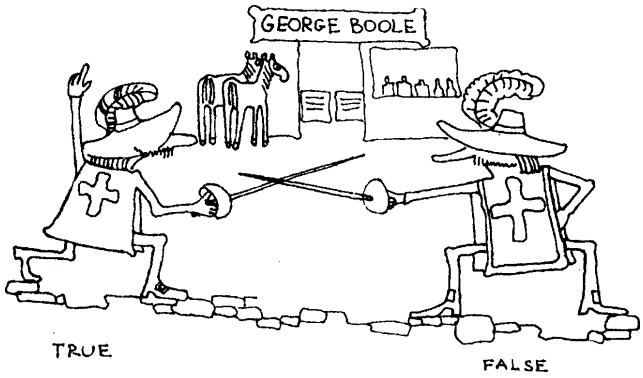
С помощью характеристических векторов удобно описывать операции над множествами: например, характеристический вектор пересечения двух множеств имеет i-й компонентой 1, если i-е компоненты всех характеристических векторов пересекаемых множеств равны 1.

Упражнение. Сформулируйте правила образования характеристических векторов объединения, разности и симметрической разности множеств**.

* В таблицу не вписаны две первые строки, в которых содержатся управляющие коды, и последняя строка, в которой символы слишком сложные.

** Симметрической разностью множеств A и B называется множество элементов, которые входят в одно из этих множеств, но не входят в другое.

С. Очень близка к этой трактовке логическая интерпретация векторов из нулей и единиц. В XIX веке английский математик Дж. Буль (George Boole, 1815—1864) предложил для математического моделирования понятий формальной логики использовать специальные переменные (называемые сейчас *логическими*, или *булевыми*). Эти переменные могут принимать только два значения: **TRUE** и **FALSE (ИСТИНА** и **ЛОЖЬ)**.

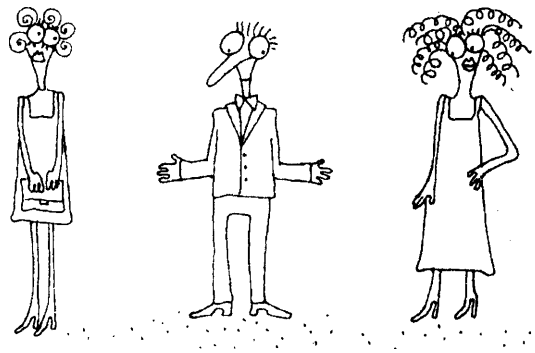


Над логическими величинами можно выполнять специальные *логические операции.*

Одноместная операция (то есть имеющая один операнд) **NOT** (НЕТ) описывает *отрицание* — она вырабатывает значение **TRUE** при значении аргумента **FALSE**, и наоборот (в математической логике эту операцию обозначают ¬).

Операция *логического умножения* **AND** (И), имеющая два операнда, вырабатывает значение **TRUE** только тогда, когда истинны оба операнда (эта операция называется также *конъюнкцией* и обозначается ∧ или &). Это можно сказать и по-другому: операция вырабатывает логическое значение "оба операнда истинны".

Операция *логического сложения* **OR** (ИЛИ) вырабатывает значение **TRUE** только тогда, когда истинен хотя бы один из операндов (эта операция называется *дизъюнкцией* (*disjunction*) и обозначается ∨ или |). Опять-таки можно сформулировать это так: операция вырабатывает значение "хотя бы один из операндов истинен".



Операция **EQU** (*EQUivalence* — эквивалентность), или ≡, вырабатывает значение "операнды равны", она называется *тождественностью*. Наряду с ней встречается операция **XOR** (*eXclusive OR* — Исключающее ИЛИ), или ≢, вырабатывающая ее отрицание, то есть значение "истинен ровно один из ее операндов".

Упражнения

1. Докажите, что (a **XOR** b) **XOR** $b = a$ (это свойство операции **XOR** часто используется в компьютерной графике).

2. Как мы знаем, точку на окружности можно задавать числом от 0 до 2π — углом поворота от начальной оси. Отрезок на окружности определяется упорядоченной парой (a, b). Докажите, что $x \in (a, b) \equiv ((a < b) \equiv ((a < x) \equiv (x < b)))$.

Таблица 3

x	y	$x \wedge y$	$x \vee y$	$x \equiv y$	$x \not\equiv y$
0	0	0	0	1	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	1	1	0

Все эти операции легко формулируются в терминах нулей и единиц (табл. 3). Их можно перенести на векторы, потребовав, чтобы операции над векторами выполнялись "покомпонентно", то есть независимо над соответствующими элементами — компонентами векторов-операндов.

В табл. 4 пример, в котором все названные логические операции выполняются над двумя векторами.

Таблица 4

x	00011101010111001000111011101
y	11101010111010110101110101010
$x \wedge y$	00001000010010000000110001000
$x \vee y$	11111111111111111101111111111
$x \equiv y$	00001000010010000010110001000
$\neg (x \equiv y)$	11110111101101111101001110111

Сравнение двух последних строчек дает пример действия операции отрицания **NOT.**

Иногда удается получать интересные вычислительные эффекты сочетанием арифметических и логических операций.

В качестве примера рассмотрим способ поиска единиц в векторе. Каждый раз будет разыскиваться самый младший разряд с единицей. Вектор, в котором идет поиск, рассматривается одновременно и как набор битов, и как двоичное представление целого числа. Первоначально вычтем из этого числа единицу. Это

действие "разменяет" младшую единицу в последовательность единиц нижестоящих разрядов. Так, если мы имели число

$$a = 0001011100011000,$$

после вычитания единицы мы получим

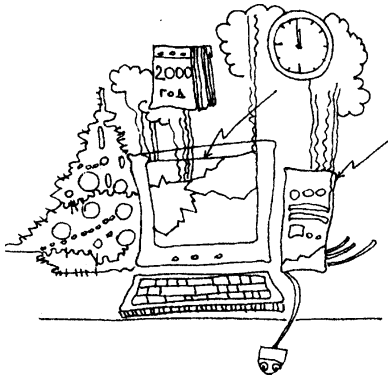
$$b = 0001011100010111.$$

Теперь действие **XOR** выделит разряды, в которых эти два набора отличаются:

$$c = 0000000000001111,$$

после чего достаточно логически умножить этот набор на a, чтобы получить окончательный результат.

D. Мы уже говорили об интерпретации вектора из нулей и единиц как состояния памяти вычислительной машины. Для такой трактовки существенно, что в зависимости от обстоятельств содержимое памяти может трактоваться по-разному. Например, в операционной системе MS-DOS для запоминания даты создания информационного объекта (файла) используется два байта — 16 битов. Семь старших битов — это год (считая нулевым годом 1980-й), следующие четыре бита — месяц (с некоторым запасом), последние пять — день месяца. В частности, пара байтов 06 27 интерпретируется и как число 1575, и как дата 7 февраля 1983 г. (а почему не января?).



E. Последовательность нулей и единиц может еще рассматриваться и как *сообщение, передаваемое по каналу связи* (передаются импульсы, каждый из которых принимает одно из двух значений), и как запись результатов экспериментов, каждый из которых может кончиться успехом (1) или неудачей (0).

F. В некоторых случаях полезно представлять себе эту последовательность как "монотонный" путь на прямоугольной решетке, в котором, например, нули соответствуют шагам направо, а единицы — шагам наверх. На *рис.* 2 представлен путь (1, 0, 0, 0, 0, 0, 1, 1, 0).

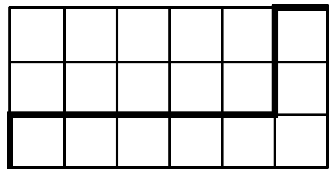
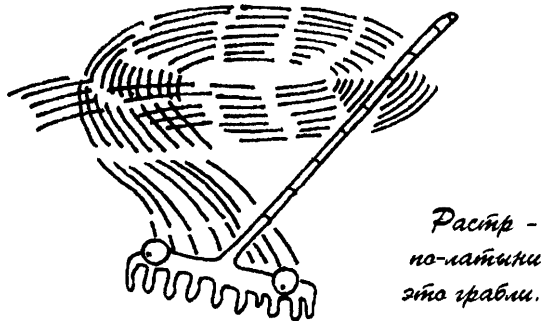


Рис. 2

G. Еще одна очень важная трактовка набора из нулей и единиц — это кодировка геометрического изображения. Двухцветная картинка (будем по традиции говорить о черно-белой картинке — черный рисунок на белом фоне) может трактоваться как *растр* — совокупность отдельных точек, расставленных на прямоугольной решетке.



Сопоставляя черным точкам единицы, а белым нули, мы и закодируем нашу картинку в виде набора нулей и единиц.

Пример. Рассмотрим решетку 16 × 16 и картинку на ней, изображенные на *рис.* 3. Кодируя столбцы числами в диапазоне $0 : 2^{16} - 1$, получаем вектор, который удобнее представить в шестнадцатеричной системе:

0000, 0000, 1800, 2C00, 2604, 7208, 4208, 4605,
2B83, 3C7C, 0C08, 0030, 0020, 0010, 0000, 0000.

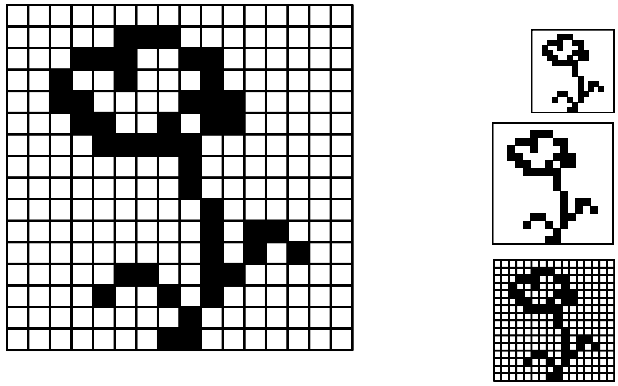


Рис. 3. Битовый скромный цветочек

Упражнение. Проверьте правильность кодировки и установите, какой край картинки выбран для младших разрядов.

Нарисуйте сами и закодируйте какой-либо символ 8×8.

Эта возможность рисования картинок точками широко используется в вычислительной технике и в обработке изображений. Например, экран дисплея рассматривается как такой растр, в котором (фиксируем конкретную ЭВМ, очень старую) 350 строк по 640 точек в каждой строке, то есть в общей сложности 224 000 точек. Для сохранения такого набора нужно (по 8 точек в одном байте) 28 000 байтов.

Символы, изображаемые на дисплее или на печатающем устройстве, также имеют растровое представление. При этом растр, в зависимости от конкретного устройства, трактуется как последовательность строк (дисплей и символы на нем) или как последовательность столбцов (матричный принтер).

H. Многократное использование таких элементарных конструкций может привести к довольно сложным структурам данных (естественно, все, что есть в компьютерах, состоит из нулей и единиц). Рассмотрим в качестве примера структуру данных в видеопамяти EGA при использовании цветового графического режима (и приводимые здесь параметры дисплея, и адаптер **EGA** — ***E**nhanced **G**raphical **A**dapter* — безнадежно устарели и не используются. Но этот адаптер удобнее для иллюстрации, чем более простые современные). Как было сказано, растр дисплея составляет 224 000 точек. В цветовом режиме каждой из этих точек можно сопоставить один из 16 цветов, что, очевидно, требует 4 бита на каждую точку.

Эти биты в системе EGA было принято располагать не все вместе, а в отдельных областях памяти по 224 000 битов каждый (на самом деле, конечно, по 256 килобитов = 32 килобайта). Эти области традиционно называются *цветовыми плоскостями.* Имеется возможность записи информации одновременно во все цветовые плоскости. Однако поскольку адресуемой единицей информации в компьютере является байт, запись естественно вести сразу в восемь смежных битов.

Чтобы все-таки иметь возможность выбирать в байте те биты, в которые требуется заносить информацию, предусмотрена система "маски". Специальной командой устанавливается байт, который определяет "картинку", заносимую в видеопамять. Здесь мы видим еще одно использование 0—1 вектора.

Система VGA, в которой число допустимых цветов значительно больше, аналогична, но в ней на задание каждого цвета отводится по 2 байта.

I. Поговорим теперь про отрицательные числа. Представление целых чисел любого знака требует новых решений. Из многих мыслимых вариантов устойчиво закрепились два — кодирование со смещением и дополнительный до двух код.

При *кодировании со смещением* к каждому числу r прибавляется константа, выбранная так, чтобы при любом r сумма $r + D$ была неотрицательной, и эта сумма кодируется вместо r. Остается решить, какой должна быть эта константа, но на этот вопрос ответить несложно. Именно, если для представления чисел выбрали k двоичных разрядов, то можно закодировать 2^k различных чисел. Будет естественно поделить эти возможности поровну между положительными и отрицательными числами. Правда, еще есть нуль. Припишем его к положительным числам, так удобнее. Тогда константа смещения равна $D = 2^{k-1}$. Например, если $k = 5$, то $D = 2^4 = 16$ и $\mathrm{code}(0) = 16$, $\mathrm{code}(11) = 27$, $\mathrm{code}(-7) = 9$, а code(17) и code(−19) не определены, так как эти числа выходят за границы интервала определения кода — 16 : 15.

Дополнительный код оперирует с остатками от деления чисел на 2^k, при этом остаток r, лежащий в интервале $[2^{k-1}; 2^k - 1]$ считается отрицательным числом $r - 2^k$. Таким образом, для тех же примеров $\mathrm{code}(0) = 0$, $\mathrm{code}(11) = 11$, $\mathrm{code}(-7) = 32 - 7 = 25$. В основной части арифметических расчетов используется дополнительный код, так как расчеты с этой записью проводятся легче и естественнее, код со смещением применяется только в специальных случаях.

Упражнения

1. Чем отличаются правила сложения чисел, заданных в обеих кодировках, и правила их сравнения?
2. Определите правила перехода от одной кодировки к другой.

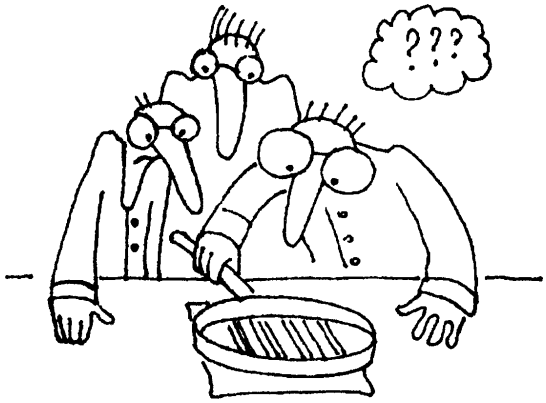
J. Еще один пример системы, построенной над двоичными системами, дают *штриховые коды (barcodes).* Их используют во многих практических информационных системах — в магазинах, в почтовой службе и других местах, где требуется быстро и просто считывать в компьютер небольшую числовую или текстовую информацию. Мы встречаем их на упаковках товаров, на книгах. Имеется много разновидностей этих кодов, и пример одной из них, так называемого *кода 3 из 9*, представлен на *рис.* 4.

Рис. 4

Каждый символ кодируется 9 полосками, поочередно черными и белыми, в конце добавляется еще одна белая полоска. Три полоски из основных девяти имеют увеличенную ширину. Так можно закодировать $9 \times 8 \times 7/6 = 84$ разных символа, что позволяет закодировать весь латинский алфавит (хотя на самом деле кодируются только прописные буквы), цифры и некоторые знаки. Сопоставляя каждой широкой полоске 1, а узкой полоске 0, получаем 10-битовые последовательности, в которых последний бит всегда равен 0, так что существенными для информации оказываются 9 битов. Посмотрим, как трактуются эти биты.

Таблица 5

Знак	**Ч** Б **Ч** Б **Ч** Б **Ч** Б **Ч**	Код черн.	Код бел.	Группа	Номер
*	**У** Ш **У** У **Ш** У **Ш** У **У**	00110	1000	*U*	5
B	**У** У **Ш** У **У** Ш **У** У **Ш**	01001	0010	*A*	2
A	**Ш** У **У** У **У** Ш **У** У **Ш**	10001	0010	*A*	1
R	**Ш** У **У** У **У** У **Ш** У **Ш**	10010	0001	*K*	8
C	**Ш** У **Ш** У **У** Ш **У** У **У**	11000	0010	*A*	3
O	**Ш** У **У** У **Ш** У **У** Ш **У**	10100	0001	*K*	5
D	**У** У **У** У **Ш** Ш **У** У **Ш**	00101	0010	*A*	4
E	**Ш** У **У** У **Ш** Ш **У** У **У**	10100	0010	*A*	5
3	**Ш** У **Ш** Ш **У** У **У** У **У**	11000	0100	1	3
9	**У** У **Ш** Ш **У** У **Ш** У **У**	01010	0100	1	9



Каждой кодовой комбинации соответствует трехэлементное подмножество, которое задается вектором из трех единиц и шести нулей. Для цифр и нулей выбраны такие коды, в которых одна широкая белая полоска и две черных. Удобно определять отдельно векторы черных и белых полосок. Для цифр в качестве вектора белых полосок берется вектор [0, 1, 0, 0], а в качестве векторов черных полосок для цифр 1, 2, 3, ..., 9, 0 берутся подряд векторы [1, 0, 0, 0], [0, 1, 0, 0], [1, 1, 0, 0] и т.д., кроме тех, где больше двух единиц. Пятая компонента вектора выбирается так, чтобы дополнить число единиц до 2. Таким образом, для 3 мы имеем [1, 1, 0, 0, 0], а для 7 — [0, 0, 0, 1, 1].

Буквы *A*—*J* кодируются как цифры, но с вектором белых полосок [0, 0, 1, 0], буквы *K*—*T* — с вектором [0, 0, 0, 1], буквы *U*—*Z*, тире, запятая, пробел и звездочка — с вектором [1, 0, 0, 0].

Теперь можно проверить пример на *рис.* 4. Составим таблицу (табл. 5), в которой первый столбец заполнен символами, далее следует 9 знаков, описывающих ширину колонок (У — узкая, Ш — широкая), затем по этим знакам сформированы коды черных и белых полосок, и, наконец, эти коды переведены в условные обозначения группы и номера в группе. В качестве представителя в каждой группе выбран символ с кодом 1. Напомним, что пропуск в кодировке чисел 7 и 11, имеющих слишком много единиц, сдвинул нумерацию, и это повлияло на представление кода черных полосок у символов *R* и 9.

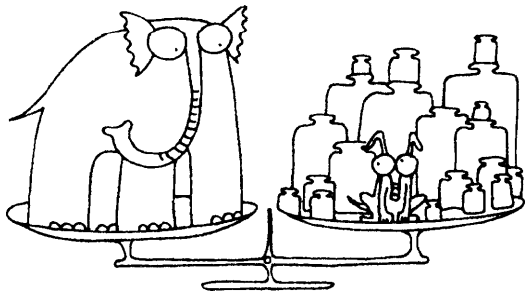
В связи с развитием так называемых *multimedia* — всевозможных периферийных видео- и аудиоустройств, появились многочисленные новые форматы, в частности, для звуковых файлов и кинофайлов (например, формат AVI). Имеется много специальных форматов графического вывода (например, метафайлы Windows), которых просто не перечесть. Вы можете найти богатую информацию в многочисленных сейчас справочниках по графическим форматам. Об одной группе форматов нужно упомянуть особо — о форматах сжатия текстов.

ЗАДАЧИ

Задачи подготовлены П.Г. Черкасовой

ЗАДАЧА 1

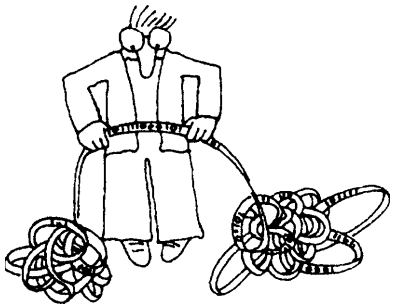
Уровень 1. Какой минимальный набор гирь вам понадобится, чтобы иметь возможность взвешивать любой вес от 1 до 200 кг?



ЗАДАЧА 2

Перевести введенное число из десятичной системы в двоичную.

Уровень 1. Переведите число 1200 в двоичную систему.



Уровень 2. Напишите программу.

Формат ввода:
Десятичная запись целого числа.

Формат вывода:
Двоичная запись.

Пример

Ввод:	*Вывод*:
13	1101

ЗАДАЧА 3

Уровень 1. Попробуйте записать в двоичной системе счисления дробные числа 1/2, 1/5 и 1/3 по аналогии с десятичной записью (0.5, 0.2 и 0.3(3)).

ЗАДАЧА 4

Выяснить, делится ли введенное двоичное число (до 1000 знаков) на 7.

Уровень 1.
Делится ли на 7 число 11000011110101101010011?
Уровень 2. Напишите программу.

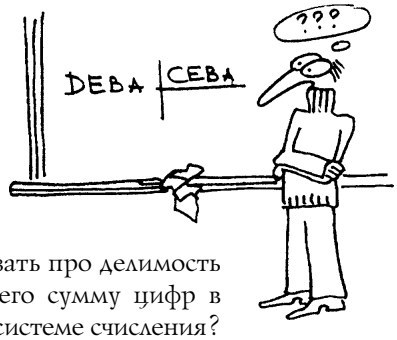
Формат ввода:
Число, в конце точка.

Формат вывода:
“Да” или “Нет”.

Пример

Ввод:	*Вывод:*
1101	Нет

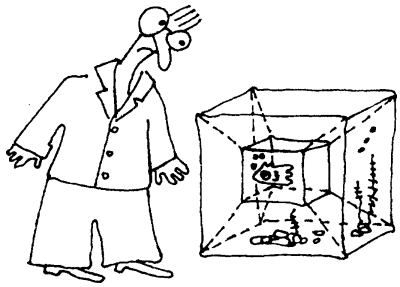
ЗАДАЧА 5



Уровень 1. В десятичной системе счисления по сумме цифр числа мы можем определить его делимость на 3 и на 9. Почему? Что мы сможем сказать про делимость числа, если узнаем его сумму цифр в шестнадцатеричной системе счисления?

ЗАДАЧА 6

В трехмерном пространстве задан куб с ребром длиной 1. Одна из вершин в точке с координатами (0,0,0), а противоположная с координатами (1,1,1).



Уровень 2. По двум введенным вершинам определить, находятся ли они на одном ребре куба или на одной его грани.

Формат ввода:
Первая вершина
Вторая вершина

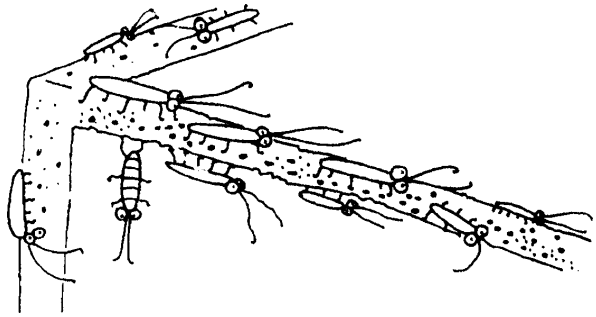
Формат вывода:
Принадлежат/Не принадлежат одному ребру.
Принадлежат/Не принадлежат одной грани.

Пример

Ввод:	*Вывод:*
001	Не принадлежат одному ребру.
010	Принадлежат одной грани.

ЗАДАЧА 7

По ребрам куба ползут тараканы. Все они ползут из вершины (000) в вершину (111) так, что каждый из них проползает при этом три ребра. Найти самое "истоптанное" ребро.



Уровень 2. Напишите программу.

Формат ввода:
Количество тараканов *M*
Путь первого таракана
Путь второго таракана
...
Путь *M*-го таракана

Путь определяется двумя промежуточными вершинами: первая вершина, пробел, вторая вершина (при этом нулевая вершина всегда (000), а третья — (111)).

Формат вывода:
Первая вершина ребра
Вторая вершина ребра

Пример

Ввод:	*Вывод:*
4	000
100 110	010
010 110	
010 011	
001 011	

ЗАДАЧА 8А

Уровень 1. Существует множество задач про рыцарей и лжецов. (В таких задачах речь идет о бинарных вопросах. То есть о вопросах, на которые отвечают либо "Да", либо "Нет".) Вот одна из них.

В стране рыцарей живут только рыцари, которые, естественно, говорят только правду. А в стране лжецов всегда лгут. Как, задав прохожему только один вопрос, определить — в какой из этих двух стран вы находитесь?

ЗАДАЧА 8Б

Уровень 1. Придумайте вопрос, на который невозможно ответить неправду.

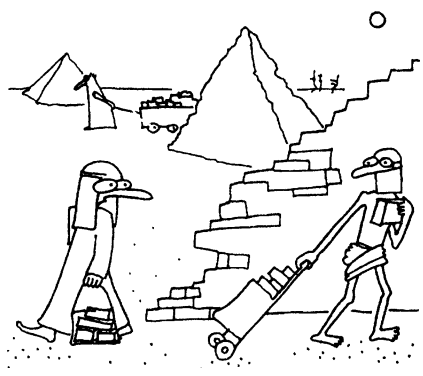
ЗАДАЧА 8В

Уровень 1. Постройте отрицание фразы: "У любого уважающего себя дворника не валяется ни одной соринки".

ЗАДАЧА 9

Игра "Ним".

Несколько (до 100) камней разложены в три кучки. Игроки по очереди забирают один или несколько камней из любой кучки (но не из нескольких сразу!). Выигрывает тот, кто берет последний камень.



Полный анализ этой игры (для любого числа камней и любого числа кучек) сделал в 1901 году Чарлз Л. Бутон, профессор математики Гарвардского университета, который и дал ей название "Ним" — от устаревшей формы английских глаголов "стянуть", "украсть".

Идею анализа можно объяснить, используя двоичные записи количества камней в кучках. Возьмем для примера $x = 011$ (3 камня), $y = 100$ (4 камня), $z = 101$ (5 камней).

Применим к этим наборам операцию "исключающее или": x **XOR** y **XOR** z. (В нашем примере результат равен 010.)

Стратегия выигрыша такова: игрок должен взять столько камней, чтобы результат вычисления состоял из одних нулей. (В нашем примере это получится, если из первой кучки взять 2 камня; тогда новые значения x, y, z будут 001, 100, 101 соответственно, а x **XOR** y **XOR** $z = 000$.)

Уровень 1. Придумайте алгоритм выигрышной игры второго игрока, если в кучках лежат 3, 4 и 5 камней и первый своим ходом берет 4 камня из последней кучки.

Уровень 2. Выясните, существует ли выигрышный ход в любой заданной игровой ситуации. Если существует, то найдите его.

Формат ввода:
Количество камней в первой кучке.
Количество камней во второй кучке.
Количество камней в третьей кучке.

Формат вывода:
Выигрышный ход существ./не существ.
Если существует:
Номер кучки
Количество взятых камней.

Пример

Ввод:	*Вывод:*
0	Выигрышный ход существует
1	3
5	4

ЗАДАЧА 10

Уровень 1. Расшифруйте текст, записанный штриховым кодом в кодировке 3 из 9.

Уровень 2. Нужно расшифровать длинное секретное донесение. Таблица кодировки неизвестна, зато известно, что каждый символ закодирован восемью битами.

Обычно в таких случаях поступают следующим образом. Берут обычный "среднестатистический" текст (например, из какого-нибудь литературного произведения) и подсчитывают, сколько раз встречается каждый символ (включая пробелы и знаки препинания).

Далее мы делаем предположение, что в закодированном тексте символы встречаются с той же частотой, и декодируем его.

Напишите программу, декодирующую текст. Обычный текст находится в файле `common.txt`. Закодированный текст находится в файле `secret.cod`. Его следует расшифровать и вывести в файл `secret.txt`.

ЗАДАЧА 11

Имеется прямоугольная решетка $N \times N$. Мы идем по ней из левого нижнего угла в правый верхний. Можно идти либо вверх, либо вправо (*рис.* 1).

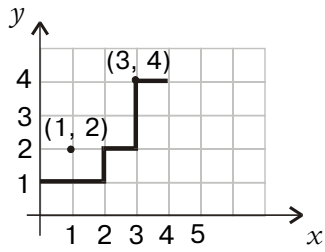


Рис. 1. Путь 10010110

Если мы идем вправо, то пишем ноль, а если вверх, то единицу. Таким образом, мы записываем наш путь.

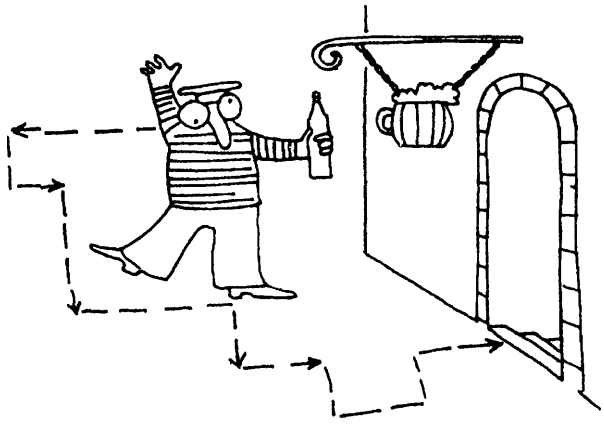
Имеется запись пути. Нужно по ней определить, проходит ли путь через точку с координатами x и y.

Уровень 1. Проходит ли путь 00101101 (в решетке 4×4) через точку (2, 3).

Уровень 2. Напишите программу.

Формат ввода:
Запись пути
Координата x
Координата y

Формат вывода:
Проходит/Не проходит



Пример

Ввод:	*Вывод:*
01010101	Проходит
2	
2	

ЗАДАЧА 12

Прямоугольная решетка $N \times N$ поставлена вертикально на один из своих углов (*рис.* 2). Некоторые клеточки перечеркнуты. В верхнюю клетку время от времени кладут мячик. Мячик катится по решетке либо направо, либо налево. При этом он не может закатиться в перечеркнутые клетки. Если мячику некуда катиться, то он останавливается, и после этого в эту клетку больше никакой мячик попасть не может.

Требуется определить, докатится ли какой-нибудь мячик до самого низа.

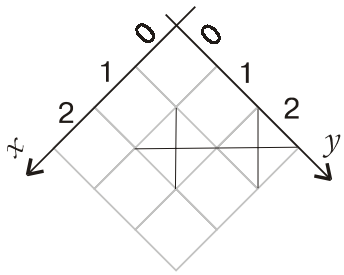
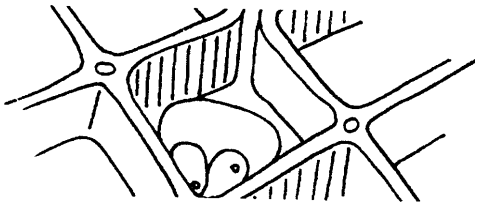


Рис. **2**

Уровень 1. Сформулируйте условие для решетки, при котором когда-нибудь один из мячиков докатится до нижней клетки.



Уровень 2. Напишите программу.

Формат ввода:
Число N
Количество перечеркнутых клеток M
Координаты первой перечеркнутой клетки через пробел
Координаты второй перечеркнутой клетки
...
Координаты M-й перечеркнутой клетки

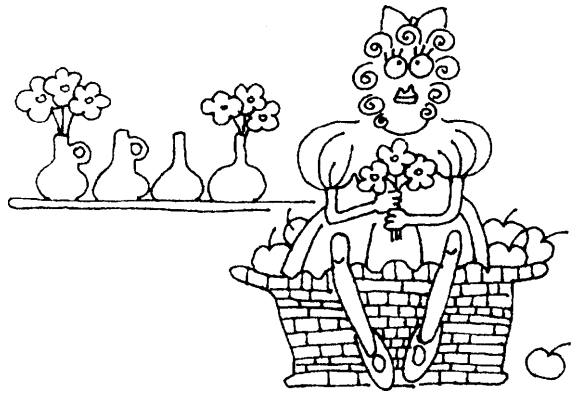
Формат вывода:
Докатится/Не докатится

Пример

Ввод:	*Вывод:*
3	Докатится
2	
0 2	
1 1	

ЗАДАЧА 13

Мальвина играет в интересную игру. Перед ней на окошке стоят несколько разноцветных ваз. В некоторых из них находятся букеты цветов. Мальвина может либо поставить новый букет в пустую вазу, либо из одной вазы вынуть букет и подарить Буратино. Если при этом получается новая комбинация букетов в вазах, то Мальвина получает от папы Карло яблоко. Если такая комбинация уже встречалась, то игра кончается. Как Мальвине набрать побольше яблок?



Уровень 1. Имеются четыре вазы. В первой из них стоят цветы. Сколько яблок удастся заработать Мальвине?

Уровень 2. В заданной ситуации заработать как можно больше яблок.

Операции описываются следующим образом: "Ставим букет в вазу № 4" или "Вынимаем букет из вазы № 1". Нумерация ваз начинается с нуля.

Формат ввода:
Количество ваз $n <= 10$
Количество полных ваз m
номер 1-й полной вазы
номер 2-й полной вазы
...
номер m-й полной вазы

Формат вывода:
Количество операций k
Описание первой операции
...
Описание k-й операции

Пример

Ввод:	*Вывод:*
2	Ставим букет в вазу № 0
1	Вынимаем букет из вазы № 1
1	Вынимаем букет из вазы № 0
3	

ЗАДАЧА 14

В математической библиотеке все книжки имеют шестизначные номера. Появилась новая книжка, и нужно дать ей номер. Математики — народ великий и безалаберный, поэтому такими мелкими вещами, как составление каталогов, они не занимаются. Требуется найти наименьший из номеров, который не принадлежит ни одной книжке. При этом не нужно сортировать книги по номерам, потому что они отсортированы по тематике. Можно просмотреть книги по очереди не

более двадцати раз. Опишите алгоритм — как найти номер для новой книжки. При этом не хватит памяти для запоминания номеров всех книг.

Формат ввода:
Количество книг в *M*.
Номер первой книжки.
Номер второй книжки.
...
Номер *m*-й книжки.

Формат вывода:
Номер новой книжки.

Пример

Ввод:	*Вывод:*
3	000000
004685	
000001	
456790	

ЗАДАЧА 15

Уровень 1. Выполните упражнения, сформулированные по ходу изложения материала занятия.

ОБЗОР НЕКОТОРЫХ РЕШЕНИЙ ЗАДАЧ ЗАНЯТИЯ 1

В № 35/99 была опубликована статья "Алгоритмы над целыми числами", соответствующая первому занятию в Заочной школе современного программирования. В конце статьи были приведены задачи. Здесь рассматриваются решения некоторых из них.

ЗАДАЧА 3

Ваш знакомый живет в стандартном двенадцатиэтажном доме в квартире 87. На каком этаже может располагаться его квартира? (На лестничной площадке одно и то же число квартир.) Вообще, как быстро определить, может ли квартира с данным номером находиться на данном этаже?

Уровень 1. После некоторого размышления можно увидеть, что на нуль никакое другое число не делится, кроме самого нуля, который, в свою очередь, делится на любое число. Предположим, что на каждом этаже помещается n квартир. Остатки от деления на n номеров квартир, расположенных на одном этаже, будут соответственно 1, 2, ..., $n - 2$, $n - 1$, 0.

При переходе на следующий этаж все номера квартир увеличиваются на n, и при этом остаток от деления на n не меняется. Зато меняется частное от деления на n с остатком. Для квартир на первом этаже оно будет равняться нулю для всех квартир, кроме квартиры с номером n. Почему такая несправедливость? Вот если бы нумерация квартир и этажей начиналась с нуля, то у нас не было бы никаких проблем — частное от деления номера квартиры на n в точности совпадало бы с номером этажа, на котором она находится. Если хочешь быть счастливым, будь им — давайте вычтем и прибавим там, где нужно, эту единичку.

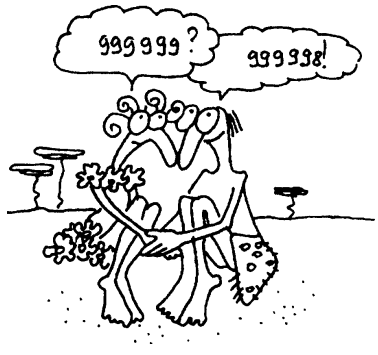
Тогда для квартиры с номером p номер этажа q будет равняться

$$q = (p - 1) \operatorname{div} n + 1$$

Следующий вопрос — может ли квартира с номером p находиться на этаже с номером q? Предположим, что может. Тогда $p - 1 = n(q - 1) + r$, где r — это номер квартиры на своем этаже (от нуля до $n - 1$), и, следовательно, он должен быть меньше n. Таким образом, нам нужно подобрать такое n, чтобы r получилось как можно меньше. Но несложно заметить, что самое маленькое r мы получим, если разделим $p - 1$ с остатком на $q - 1$. Если при этом r окажется меньше n, то, значит, мы сумели расположить квартиру на требуемом этаже. В противном случае квартира не может располагаться на q-м этаже.

ЗАДАЧА 4

Язык племени мумбу-юмбу состоит из шестибуквенных слов, составленных из букв {А, Б, В, Г, Д, Е, Ж, З, И, К}. В Оксфорде издан полный словарь слов этого языка (упорядоченных по алфавиту):



ААААА,
ААААБ,
ААААВ,
...
КККККИ,
КККККК.

На каждой странице словаря помещается 15 слов.

На каких страницах и в каких строках находятся слова ДЕКАДА и ЗАБАВА?

Уровень 1. Так уж получилось, что букв в племени мумбу-юмбу ровно столько, сколько и цифр в десятичной системе счисления. Это дает нам возможность каждую букву заменить на соответствующую цифру — *А* на 0, *Б* на 1, ..., *К* на 9. Таким образом, все слова превратятся в числа от нуля (000000) до 999 999. Достаточно выяснить частное от деления такого слова на 15 с остатком, чтобы узнать, сколько полных страниц набралось из слов, стоящих раньше данного, если смотреть в алфавитном порядке. А когда мы увеличим это число на единицу, мы узнаем номер страницы, на которой записано слово:

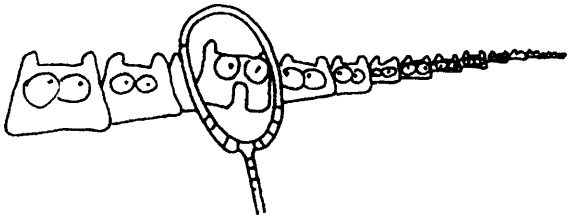
ДЕКАДА mod 15 + 1 = 459 040 mod 15 +1 = 30 603

ЗАБАВА mod 15 + 1 = 701 020 mod 15 +1 = 46 735

ЗАДАЧА 5

Найдите цифру с номером n в последовательности 01234567891011121314151617181920... записанных подряд натуральных чисел.

Уровень 1. Для начала признаемся в неточной формулировке задачи в предыдущем номере. Дело в том, что последовательность 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 ... не является последовательностью записанных подряд натуральных чисел. Как известно, натуральные числа начинаются с единицы. Поэтому следовало написать: последовательность записанных подряд целых чисел, начиная с нуля.



Теперь приступим к разбору. Сначала разберемся, со "сколько значными" числами нам придется работать. Если номер n меньше десяти, значит, соответствующее число состоит из одной цифры.

В противном случае оно состоит из двух или более цифр. В таком случае давайте выкинем из рассмотрения однозначные числа, то есть уменьшим число n на 10, и будем считать, что последовательность начинается с десяти. Наше число 1999, таким образом, превращается в 1989.

Давайте, таким образом, уменьшим наш ряд по максимуму. В итоге должна остаться последовательность записанных подряд чисел одинаковой длины, которые начинаются со степени десятки.

Каждый шаг уменьшения ряда сводится к тому, чтобы определить, можем ли мы выкинуть ведущий кусок ряда из чисел одинаковой длины k так, чтобы в получившейся последовательности осталась интересующая нас цифра с номером n.

После первого шага (удаление всех однозначных чисел) $k = 2$.

Сколько существует k-значных чисел? Их количество равняется $10^k - 10^{k-1}$. Следовательно, количество цифр в них p будет равняться $k(10^k - 10^{k-1})$. Если наш номер n больше или равен p, то нужно уменьшить его на p и считать, что наш ряд начинается с 10^k (увеличиваем k на единицу). Если $n < p$, то это значит, что мы уже уменьшили наш ряд насколько возможно.

Посмотрим, что происходит с числом 1989.

n	k	Вырезаемый кусок	Его длина
1989	2	двузначные	$90 \cdot 2$
1809	3	трехзначные	$900 \cdot 3 > n$

Итак, нам нужно найти цифру с номером n (1809) в последовательности подряд выписанных k-значных (трехзначных) чисел.

Теперь мы можем определить число A, в котором находится искомая цифра. Его номер в последовательности равен n div k (если считать с нуля). Следовательно, $A = n$ div $k + 10^{k-1}$ (так как 10^{k-1} — первое число в последовательности). То есть само число $A = 1809$ div $3 + 100 = 703$.

Далее, нам нужно определить цифру с номером n mod k в k-значном числе A. В нашем случае это нулевая цифра числа 703 — то есть семерка.

Уровень 2

```
var n, dec : longint;
{в переменной dec хранится k-я степень
 десятки}
A, k, p, i: integer;
{p - количество k-значных чисел}
begin
 readln(n);
 k:=1;
 dec:=10;
 p:=10;
 while p >= n do
    begin
      n:=n-p; {уменьшаем ряд}
      inc (k); {k - число цифр в числах в
      dec:=dec*10; {начале ряда}
      p:=dec-dec div 10;
    end;
A:=n div k + dec div 10;
dec:=1;
for i:=1 to k - n mod k do dec:=dec*10;
{вычисляем 10^(k-n mod k)}
writeln((A mod dec) div (dec*10));
end.
```

ЗАДАЧА 6

Для любого натурального числа алгоритм совершает следующие операции: отделяет от числа первую цифру и прибавляет ее к числу из оставшихся цифр. Процесс продолжается до тех пор, пока в числе останется одна

цифра. Определить результат работы алгоритма для чисел вида $\underbrace{88...8}_{n}$ при различных значениях n.

Уровень 1. Обычно в таких задачах следует искать величину, которая от шага к шагу не меняется. Она называется инвариантом. В данном случае инвариантом является остаток от деления на девять.

Посмотрим, что происходит с числом при каждом шаге алгоритма в арифметике остатков. Во-первых, мы "отрезаем" от него первую цифру — то есть вычитаем из него первую цифру, помноженную на какую-то степень десятки. В арифметике остатков по модулю 9 любая степень десятки равна единице. Поэтому "отрезание" первой цифры числа в арифметике остатков равносильно вычитанию этой цифры. А затем (во-вторых) мы прибавляем к нему эту же цифру. В итоге остаток от деления на 9 не меняется.

На последнем шаге алгоритма мы можем получить число от 1 до 9 (потому что остается только одна цифра). У всех этих чисел различные остатки от деления на 9. Поэтому, зная остаток исходного числа, мы можем однозначно указать результат.

А остаток исходного числа легко вычислить по сумме цифр. Он равен $8n \bmod 9$.

Уровень 2. Напишем программу, которая выдает все промежуточные результаты для чисел из не более чем 50 цифр. Здесь нет ничего сложного. Нужно просто аккуратно выполнить все действия. Цифру будем прибавлять столбиком — как учили в первом классе.

```
var a: array[0..49] of 0..9;
n, i:1..50;
p: byte; {число "в уме"}
begin
  readln(n);
  {будем записывать цифры в массив так,
   чтобы номер ячейки совпадал со степенью
   десятки в данном разряде}
  for i:=n-1 downto 0 do readln(a[]);
  while n>1 do
    begin
      p:=a[n-1]; {вычеркиваемая цифра}
      dec (n);
      j:=0; {позиция в числе}
      while p>0 do
        begin
          p:=p+a[j];
          a [J]:= P mod 10;
          p : = p div 10;
          inc [ j ];
          if (j>n-1) and (p>0)
          {а вдруг мы вылезли}
            tnen begin
                  a[j]:=0; {за пределы числа}
                    n:=j+1;
                  end;
        end;
      for i:=n-1 downto 0 do write(a[j]);
      writeln;
    end;
end.
```

ЗАДАЧА 7

Известно, что любую дробь $\frac{a}{b}$, где a и b — натуральные числа, можно представить в виде цепной дроби. Опишите алгоритм. Представьте в виде цепной дроби $\frac{17}{239}$.

Уровень 1. Для тех, кто разобрался в теории, описанной в прошлом номере, алгоритм прост.

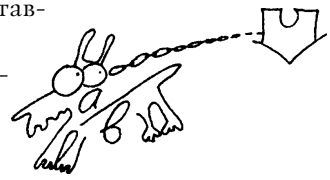
Шаг алгоритма: вычисляем целую часть от деления a на b и запоминаем в качестве очередного c. Вместо a записываем остаток от деления a на b. Если он равен 1, то запоминаем b в качестве последнего частного. Если нет, то меняем местами a и b и переходим к следующему шагу.

Вот что произойдет с числом $\frac{17}{239}$:

a	b	c	r
17	239	0	17
239	17	14	1

$$\frac{17}{239} = 0 + \frac{1}{14 + \frac{1}{17}}$$

Уровень 2. Напишите программу, которая по данным числам a и b представляет дробь $\frac{a}{b}$ в виде цепной ($a, b < 1\,000\,000$).




```
var a, b, r, n: integer;
c: array[0..200];
begin
 readln(a);
 readln(b);
 n:=0; {количество элементов в с}
 r:=5; {остаток - любой, лишь бы не единица}
 while r<>1 do
   begin
     c[n] : =a div b;
     inc (n);
     r:=a mod b;
     b: =a;
     a: =r;
     end;
   c[n] :=a;
   inc (n);
   writeln(n);
   for i:=0 to n-1 do writeln(c[i]);
end.
```

ЗАДАЧА 9

Уровень 1. Обобщить алгоритм Евклида с делением на 2 и вычитанием для наборов из более чем двух чисел. Применить обобщенный алгоритм к набору {72; 84; 132; 144}.

ШАГ 1. Положим НОД равным единице. Пока все числа из набора четные, будем их всех делить на 2 и домножать на эту двойку НОД. Когда хотя бы одно из чисел станет нечетным, переходим к шагу 2.

ШАГ 2. Делим все четные числа набора на 2 до тех пор, пока деление возможно.

ШАГ 3. Если в наборе более одного ненулевого числа, то заменяем большее из чисел набора разностью со следующим по величине и переходим к шагу 2, иначе умножаем НОД на полученный результат и заканчиваем работу.

				НОД
72	84	132	144	1
36	42	66	72	2
18	21	33	**36**	4
9	**21**	**33**	9	4
9	21	**12**	9	4
9	**21**	3	9	4
9	**12**	3	9	4
9	3	3	**9**	4
9	**3**	3	0	4
6	3	3	0	4
3	**3**	3	0	4
3	0	**3**	0	4
3	0	0	0	12

Уровень 2. Доказать, что если одно из чисел *a, b* четное, а другое нечетное, то следующее преобразование сохраняет множество общих делителей *a* и *b*:

ЕСЛИ ровно одно из чисел *a, b* четное

ТО поделить его на 2

ИНАЧЕ заменить большее из чисел *a, b* их разностью

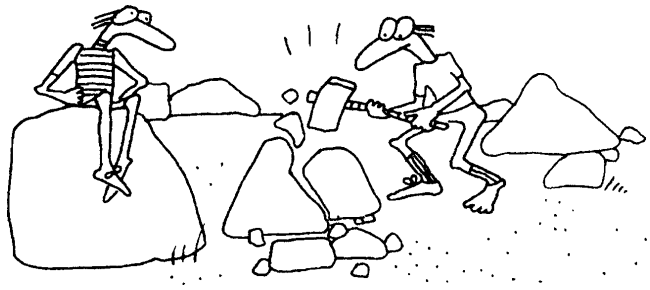
Понятно, что при делении одного из чисел на 2 новых общих делителей не появится.

Что же будет со старыми? Предположим, *d* является одним из общих делителей. Пусть число *a* четно. Тогда $a = 2dk$ (*d* не может делиться на два, так как одно из чисел не кратно двум, а *d* — общий делитель). Видно, что от деления числа *a* на 2 оно не перестанет делиться на *d*.

Следовательно, при первом преобразовании множество общих делителей не изменится. То, что замена числа разностью не меняет множества общих делителей, доказывалось на первом занятии.

ЗАДАЧА 11

Из кучки камней двое играющих по очереди берут 1, 2 или 3 камня. Проигрывает тот, кто берет последний камень. Предположим, что всего камней *N*. Кто из игроков имеет выигрышную стратегию? Опишите ее. А если выигрывает взявший последний камень?



Уровень 1. Давайте определимся, что такое выигрышная стратегия. Говорят, что игрок имеет выигрышную стратегию, если на любые ходы противника он может отвечать так, что придет к победе.

Выигрышная позиция — это позиция, в которой у игрока, который ходит первым, существует выигрышная стратегия. Проигрышная позиция — это позиция, в которой любые ходы приводят к проигрышу (при умном противнике).

Разберем позиции в нашей игре. Понятно, что ситуация, когда остался 1 камень, является проигрышной. Игрок неминуемо терпит поражение.

Если позиция такова, что существует ход, приводящий к проигрышной позиции, значит, она выигрышная. Действительно, делая такой ход, мы предоставляем противнику проигрышную позицию. Таким образом, ситуации, когда осталось 2, 3 или 4 камня, являются выигрышными.

Если любой ход приводит в выигрышную ситуацию, значит, позиция проигрышная — мы любым ходом неминуемо приводим противника к хорошему расположению. Так, позиция с пятью камнями является проигрышной.

Таким образом, мы можем сосчитать любую игровую ситуацию (плюсами отмечены выигрышные):

№	1	2	3	4	5	6	7	8	9	10	11...
	−	+	+	+	−	+	+	+	−	+	+...

Вот табличка для игры, когда выигрывает взявший последний камень:

№	1	2	3	4	5	6	7	8	9	10	11...
	+	+	+	−	+	+	+	−	+	+	+...

Если в начале игры позиция выигрышная, выигрывает первый игрок, в противном случае — второй.

К задаче можно подойти и с другого конца — для того чтобы выиграть, нужно дополнять ход противника до четырех. Если он взял один камень, то мы три; если он — два, то и мы два. Тогда остаток от деления на четыре после нашего хода будет один и тот же. В первом случае это единица, а во втором — ноль.

ЗАДАЧА 12

Уровень 1. Алиса, попав в страну чудес, забыла таблицу умножения. Она говорит: семью семь будет... пятнадцать, девятью девять будет... тринадцать.

Определите, в какой модульной арифметике такие правила умножения справедливы. Сколько всего таких арифметик?

Но семью семь на самом деле 49, а девятью девять — 81. Значит,

$$49 \equiv 15 \pmod{m}$$
$$81 \equiv 13 \pmod{m}$$

Это означает, что 49 и 15, так же, как и 81 и 31, дают одинаковые остатки при делении на m. Другими словами, разность каждой пары делится на m, что можно записать так:

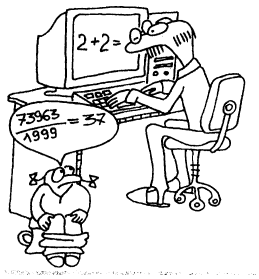
$$34 \equiv 0 \pmod{m}$$
$$68 \equiv 0 \pmod{m}$$

Следовательно, число m является одним из общих делителей чисел 34 и 68. Кроме того, это число должно быть больше 15, чтобы в модульной арифметике существовали числа 7, 9, 13 и 15. Этим условиям удовлетворяют числа 34 и 17.

Уровень 2. Напишите программу, которая по введенным примерам таблицы умножения определяет, существует ли модульная арифметика, в которых эти примеры имеют место.

```
var c: array[0..9];
    n, k, i: integer;
    maxn: integer;
    m: array[0..100] of integer;
function InputEquation: integer;
{Функция вводит уравнение и возвращает число,
 которое, исходя из уравнения, должно
 равняться нулю по данному модулю.
 Также она записывает в переменную maxn
 значение наибольшего введенного числа -
 нужно будет обеспечить его существование}
 var s:string;
     a, b, c, mult, eq, err: integer;
 begin
  readln(s);
  mult:=2;
  while s[mult] <> '*' do inc(mult);
  eq:=mult+2;
  while s[eq] <> '=' do inc (eq);
  val(copy(s, 1, mult - 1), a, err);
  val(copy(s, mult+1, eq-mult-1), b, err);
  val(copy(s, eq+1, length(s) - eq), c, err);
   if a>maxn then maxn:=a;
   if b>maxn then maxn:=b;
   if c>maxn then maxn:=c;
   InputEquation:= abs (a * b - c);
 end;
begin
  maxn:=1;
  readln(n);
  k:=0;
 {k - число ненулевых чисел в массиве c}
  for i:=0 to n-1 do
 {ввод и обработка уравнений}
    begin
      c[i]:=InputEquation;
      if c[i]<>0 then inc(k);
      {записываем только}
    end;
     {ненулевые значения}
  while k>1 do {вычисляем НОД массива c}
      begin
        if c[k-1]>c[k-2]
         then c[k-1]:=c[k-1] mod c[k-2]
         else c[k-2]:=c[k-2] mod c[k-1];
        if c[k-2]=0 then
         begin
           c[k-2]:=c[k-1];
           c[k-1]:=0;
         end;
        if c[k-1] = 0 then dec (k);
      end; {в итоге НОД находится в C[0]}
   k:=0;
   {теперь k - число общих делителей)
   for i:=maxn+1 to c[0] do
   {или делителей НОД, больших maxn}
     if c[0] mod i = 0 {bo}
       then begin
              m[k]:=i;
              inc (k);
            end
   writeln(k);
   for i:=0 to k-1 do writeln(m[i]);
end.
```

ЗАДАЧА 14



Уровень 2. Напишите программу, которая по заданным не более чем 25-значным числам a, b, c и d проверяет, верно ли, что $ab = cd$.

```
type
  LongNumber = record
      num: array[0..24] of 0 . . 9;
      len: integer;
  end;
var p: array[2..100];
{массив простых чисел}
pcount: integer; {их количество}
    a, b, c, d: LongNumber;
    i, j: integer;
{процедура вводит длинное число}
procedure InputLongNumber(var n: LongNumber);
  var s: string;
  begin
    readln(s);
```

```
    for i :=0 to length (s)-1 do
      n.num[i]:=s[length(s)-i];
      n.len :=length [s];
  end;
{функция находит остаток от деления
 длинного числа на k}
function Residual(var n: LongNumber;
                      k: integer): integer;
  var d, r, i :integer;
  begin
    r: =0;
    d: =1;
  for i:=0 to n.len-1 do
    begin
      r:=(r+d*n.num[i]) mod k;
      d:=d*10 mod k;
    end;
  Residual:=r;
  end;
begin
  for i:=2 to 100 do p[i]:=i;
  for i:=2 to 10 do
  {метод нахождения простых чисел}
    begin
    {называется Эратосфеновым решетом}
      if p[i] = 0 then continue;
      for j:=2 to 100 div i do p[i*j]:=0;
    end;
  pcount:=0;
  for i:=2 to 100 do
   {выкидываем из массива нули}
    if p[i]<>0 then begin
                      inc(pcount);
                      p[pcount+l] :=p[i];
                    end;
  InputLongNumber(a);
  InputLongNumber(b);
  InputLongNumber(c);
  InputLongNumber(d);
  eq: = true;
  for i:=2 to pcount+1 do
  {проверка равенства}
  if (Residual(a,p[i] ) * Residual(a,p[i])-
      Residual(a,p[i]) * Residual(a,p[i]))
       mod p[i]<>0
         then begin eq:=false; break; end;
  if eq then writeln ('Да')
        else writeln ('Нет');
end.
```

Дистанционная обучающая олимпиада по географии (ДООГ)

Международный детский компьютерный центр ИПС РАН г. Переславля-Залесского, Окружной методический центр СЗУО г. Москвы, при поддержке Российской Академии повышения квалификации работников образования, объединения педагогических изданий “Первое сентября” проводят в октябре — декабре 1999 года дистанционную обучающую олимпиаду по географии — ДООГ для школьников 6—11-х классов.

Цели олимпиады:

• Снабдить учителей географии дидактическим материалом, методическими пособиями для проведения школьных олимпиад по географии.

• Оказать помощь школьникам и педагогам в подготовке к олимпиадам, стимулировать олимпиадное движение за счет обмена опытом работы учителей географии и школьников, информирования о сетевых информационных ресурсах.

• Подготовить базу для создания виртуального методического объединения учителей географии на основе постоянно действующего списка рассылки (**dooglist@egroups.com**).

Участники олимпиады:

Команды школьников, состоящие из учащихся разных классов, интересующихся историей открытий Ойкумены и ее познанием, под руководством координатора.

Порядок проведения олимпиады:

Олимпиада проводится дистанционно, с помощью электронной почты. Для участия необходимо с 1 по 15 октября 1999 года направить заявку (форма прилагается) и оплатить организационный взнос (около $1 по курсу ЦБ).

• **Ноябрь**. Обучающий тур. Участники получают задания, ответы, методические рекомендации для проведения внутришкольных олимпиад. Начинает работать информационная поддержка участников (информирование о ресурсах Интернет для учителя географии, школьников, выполняются заявки на поиск материалов по сети).

• **Декабрь**. Участники получают задания конкурсного тура, методические рекомендации по его проведению. Проводится перекрестная проверка. Подводятся итоги. Высылаются призы.

Поощрения участников олимпиады:

Все участники ДООГ получат командные, а призеры и именные дипломы, педагоги — рекомендательные письма для руководителей учреждений, наиболее активные участники — документы от РАПКРО о продуктивности своей работы в проекте, информацию о ресурсах Интернет и компьютерных программах для учителей географии. Отличившиеся — ценные подарки (поиск спонсоров ведется): книги, CD и др., льготные условия приобретения путевок в международный детский компьютерный центр.

Вид заявки:

Заявку следует направлять на электронный адрес координаторов олимпиады (**doog@mail.ru**) можно заполнить ее на сайте **http://www.attend.to/doog** по следующей форме:

Заявка на участие в ДООГ от учителя географии (Ф.И.О. полностью):	
Образовательное учреждение (полное название):	
Руководитель учреждения (Ф.И.О. полностью):	
Технический консультант, если есть (Ф.И.О. полностью):	
Почтовый адрес (с индексом):	
Электронный адрес:	
Предполагаемое количество участников:	
Класс (возраст участников):	

Научный руководитель ДООГ: ***О.В. Устюкова**, руководитель образовательных программ МДКЦ ИПС РАН.*

Руководитель олимпиады: ***Ж.В. Пашкова**, методист по географии ОМЦ СЗУО.*

Координаторы: ***А.В. Калеников**, методист по телекоммуникациям ОМЦ СЗУО; **В.Е. Скородумов**, методист МДКЦ ИПС РАН.*

Познакомимся с экспертной системой

И.И. Данилина

Экспертные системы (ЭС) — яркое и быстро развивающееся направление исследований в области искусственного интеллекта. Отличительная их черта — способность накапливать знания и опыт квалифицированных специалистов (экспертов) в какой-либо области. Затем, пользуясь этими знаниями, пользователи ЭС, не имеющие необходимой квалификации, могут решать свои задачи почти столь же успешно, как это делают эксперты. Такой эффект достигается за счет того, что система в своей работе воспроизводит примерно ту же цепочку рассуждений, что и человек-эксперт.

Экспертные системы используются в самых различных сферах человеческой деятельности — медицине, геологоразведке, экономике и многих других. Существуют как узкоспециализированные ЭС, так и "оболочки", пользуясь которыми, можно, не будучи программистом, создавать свои ЭС.

Школьная информатика сегодня уже не должна обходить эту тему молчанием, причем простого рассказа тут недостаточно. Желательно, чтобы ученики поработали с "настоящей" экспертной системой, посмотрели, как она устроена, и сделали что-то подобное сами. Для этого предлагаем использовать демонстрационную версию оболочки RuleBook. Она, конечно, не обладает всеми возможностями рабочей версии, но для первого знакомства вполне подходит.

Найти эту оболочку (вместе с образцами простейших ЭС) можно в разделе "Информатика" сервера "Первого сентября" **www.1september.ru**

Система обладает удобным и понятным (хотя и "англоязычным") интерфейсом, и освоить простейшие приемы работы с ней достаточно легко. "Заинтересованные" ученики могут самостоятельно освоить и некоторые дополнительные приемы.

Рассмотрим задачу, иллюстрирующую основные принципы разработки ЭС. Создадим систему, которая будет помогать ученику (или изучающему русский язык иностранцу) правильно писать такие слова, как КАСАТЬСЯ и КОСНУТЬСЯ, БЛЕСТЕТЬ и БЛИСТАТЬ, и т. п., где мы имеем дело с чередованием гласных "А" и "О" или "Е" и "И" в корнях слов.

А начнем с того, что сами разберемся в этом непростом вопросе.

Чаще всего знания эксперта можно представить в виде системы правил и набора фактов, на основании которых выбирается тот или иной вариант решения проблемы.

Вот что сказано в учебнике русского языка:

1) В корнях
   БЕР- — БИР-;
   ДЕР- — ДИР-;
   МЕР- — МИР-;
   ПЕР- — ПИР-;
   БЛЕСТ- — БЛИСТ-;
   ЖЕГ- — ЖИГ-;
   СТЕЛ- — СТИЛ-;
   ЧЕТ- — ЧИТ-
   пишется И, если за корнем следует суффикс -А-. В других случаях пишется Е.

   Исключения: *сочетать, сочетание, чета.*

2) ГАР- — ГОР- —
   под ударением пишется А, без ударения — О.

   Исключение: *пригарь.*

3) ЗАР- — ЗОР- —
   под ударением пишется та буква, которая слышится, без ударения — буква А.

   Исключения: *зорянка, зоревать.*

4) КАС- — КОС- —
   пишется А, если за корнем следует суффикс -А-. В других случаях пишется О.
5) ЛАГ- — ЛОЖ- —
   перед Г пишется А, перед Ж — О.
6) РАСТ- — РАЩ- — РОСТ- —
   А пишется перед СТ и Щ.

   Исключения: *росток, ростовщик, отрасль.*

7) МАК- (в значении "погружать во что-то") — МОК- (в значении "пропускать влагу").
8) КЛАН- — КЛОН- —
   под ударением пишется та буква, которая слышится, без ударения — О.
9) ТВОР- — ТВАР- —
   под ударением пишется та буква, которая слышится, без ударения — буква О.

   Исключение: *утварь.*

Это и есть те знания, которыми должна обладать экспертная система по интересующему нас вопросу.

Для создания ЭС нужно подготовить вопросы, которые система будет задавать пользователю, и варианты ответов. Кроме того, нужно установить порядок следования вопросов (зависящих, конечно же, от полученных ответов), то есть определить ход рассуждений эксперта. Ну и, разумеется, нужны варианты решений, которые ЭС предлагает пользователю.

Для нашего примера варианты решений перечислить нетрудно:

1) пишется Е,
2) пишется О,
3) пишется А,
4) пишется И или А (их целесообразно объединить),

5) пишется так, как слышится (правило 3).

Для удобства предусмотрен еще один вариант, позволяющий объединить все исключения в одном вопросе:

6) ответ ясен из вопроса.

Вопросы, которые будет задавать система, можно формулировать по-разному. Например, так.

1. К какой группе относится корень слова?

Варианты ответов:

1.1. БЕР — БИР, ДЕР — ДИР, МЕР — МИР, ПЕР — ПИР, БЛЕСТ — БЛИСТ, ЖЕГ — ЖИГ, СТЕЛ — СТИЛ, ЧЕТ — ЧИТ, КАС — КОС.
1.2. ГАР — ГОР, КЛАН — КЛОН, ТВОР — ТВАР.
1.3. ЗАР — ЗОР.

2. Гласная в корне ударная или безударная?

Варианты ответов:

2.1. Ударная
2.2. Безударная

3. Является ли слово одним из слов-исключений: СОЧЕТАТЬ, СОЧЕТАНИЕ, ЧЕТА, ЗОРЯНКА, ЗОРЕВАТЬ, РОСТОК, РОСТОВЩИК, ОТРАСЛЬ, УТВАРЬ?

Варианты ответов:

3.1. Да
3.2. Нет

4. Следует ли за корнем суффикс -А-?

Варианты ответов:

4.1. Да
4.2. Нет

5. Корень оканчивается на -Г- или на -Ж-?

Варианты ответов:

5.1. На -Г-
5.2. На -Ж-

6. Корень оканчивается на -СТ-, на -Щ- или на -С-?

Варианты ответов:

6.1. На -СТ-
6.2. На -Щ-
6.3. На -С-

7. Что означает слово?

Варианты ответов:

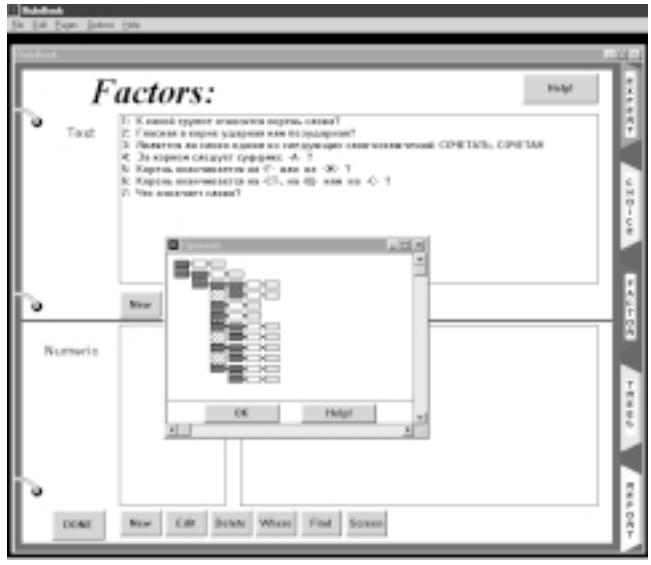
7.1. Погружать что-то (обмакнуть в сметану)
7.2. Пропускать влагу (ноги промокают)

Ну и, наконец, опишем (представим на схеме) ход рассуждений (см. стр. 27).

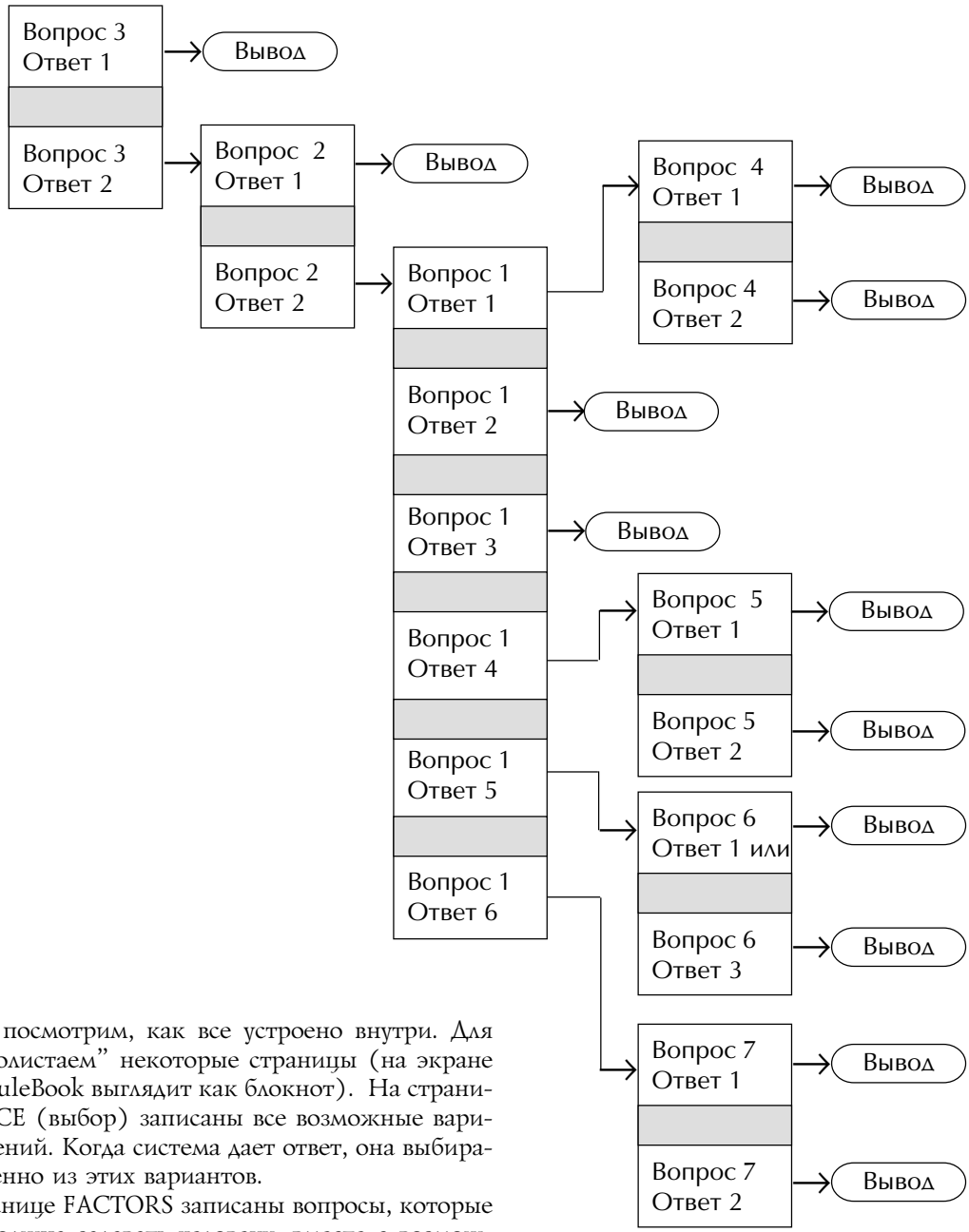
Именно эта задача реализована в прилагаемом к системе образце* (файл PRAVILO.RB1).

Если вы скопировали систему RuleBook и хотите увидеть ее в действии, то запустите программу на выполнение (файл RB_DEMO.EXE) и откройте файл с примером (PRAVILO.RB1). Выберите в верхнем меню сначала пункт `Options`, а затем `Run` (выполнение). Система начнет задавать вопросы, а вам нужно отвечать на них, выбирая подходящий вариант ответа.

Похожим образом действовал бы и специалист — человек, желающий помочь вам. В какой-то момент "электронный консультант" сочтет, что располагает достаточной информацией, чтобы дать вам дельный совет. Тогда этот совет появится на экране.



* Напомним, что получить систему RuleBook и файлы с примерами, рассматриваемыми в этой статье, вы можете на сервере **www.1september.ru**.



Теперь посмотрим, как все устроено внутри. Для этого "пролистаем" некоторые страницы (на экране система RuleBook выглядит как блокнот). На странице CHOICE (выбор) записаны все возможные варианты решений. Когда система дает ответ, она выбирает его именно из этих вариантов.

На странице FACTORS записаны вопросы, которые система должна задавать человеку, вместе с возможными вариантами ответов.

И, наконец, на странице TREES (деревья) все элементы системы связаны воедино. Эта связь носит название *дерево*, поскольку с каждым вопросом путь поиска решения разветвляется подобно дереву на столько ветвей, сколько вариантов ответа на вопрос предусмотрено.

Иногда эксперт не на сто процентов уверен в правильности выбранного варианта. Для учета такого обстоятельства в ЭС вводится *коэффициент уверенности*. В системе RuleBook он называется *Confidence* и указывается вместе с каждым ответом системы.

Следует иметь в виду, что демонстрационная версия позволяет создавать не более трех деревьев, у каждого из которых может быть не более 14 ветвей.

И в заключение несколько заданий для самостоятельной работы.

Задача 1. Советы по выбору профессии

Беседа со специалистами по профориентации позволила сформулировать ряд правил для выбора сферы деятельности (на основании информации, касающейся успехов в области различных школьных дисциплин).

1. Если вы успешно занимаетесь искусствами и неплохо успеваете по математике, то вам можно попробовать себя в архитектуре.
2. Если математика у вас "выше среднего" и вам хорошо дается экономика, то вы — прирожденный бухгалтер.

3. Если вас увлекает биология, да и все точные науки даются неплохо, то, возможно, ваше призвание — медицина.
4. Если вы хорошо владеете родным языком, "выше среднего" — математикой, да и другие точные науки даются вам неплохо, то вам можно порекомендовать посвятить себя юриспруденции.
5. Если же вы блестяще владеете родным языком и хоть немножко разбираетесь в гуманитарных науках, то вам стоит попробовать себя в литературе.
6. Если вы можете похвастаться успехами в родном языке и экономике, а математика вам не дается — попробуйте себя в политике.
7. И, наконец, успехи хотя бы в каких-нибудь (точных или гуманитарных) науках и в математике дают хороший шанс преуспеть в администрировании.

Создайте на основании этих правил экспертную систему — консультанта для школьников и абитуриентов, выбирающих для себя профессию.

Задача 2. Консультант по инвестициям

В одной американской книге, изданной в 1987 году, описывается стратегия поведения на финансовом рынке. Автор опирается на модель циклической смены подъемов и спадов деловой активности. Следует отметить, что со времен второй мировой войны такой цикл повторился уже 10 раз, то есть модель можно считать пригодной для описания реальной ситуации.

Каждый деловой цикл состоит из четырех этапов. Это — рост, пик, снижение и кризис. Каждому этапу соответствует своя стратегия поведения на рынке. Продолжительность этапов различна и меняется от цикла к циклу, а потому определить, на каком этапе мы находимся, непросто.

Возрастание краткосрочных процентных ставок дает основание ожидать скорого пика. А если, кроме того, краткосрочные ставки выше долгосрочных, то более вероятен спад.

О приближающемся пике скорее всего свидетельствует рост цен в течение нескольких месяцев на фоне положительного общественного мнения.

Если же общественное мнение скорее отрицательное, чем положительное, да еще деловая активность снизилась, то, по-видимому, близок кризис.

Если величина совокупного национального продукта уменьшается уже в течение двух кварталов, то пять шансов из десяти за то, что мы на пороге кризиса.

А вот если процентные ставки падают уже несколько месяцев, то надо ожидать роста. Приближающиеся выборы президента тоже позволяют ожидать роста.

Определив, на какой стадии мы находимся, можно давать рекомендации о том, как действовать, куда вложить средства.

В ожидании пика имеет смысл покупать акции или ценные бумаги. Если курс доллара упал по отношению к другим валютам, вероятно, следует попробовать купить акции. В преддверии кризиса можно с большей уверенностью советовать покупать акции. А если к кризису добавляется значительный прирост населения в районе, то лучше покупать землю.

Перед кризисом или началом подъема можно с большой долей уверенности порекомендовать не продавать землю и акции. А в обратной ситуации — на пике и перед началом спада — лучше придержать ценные бумаги.

Основываясь на этих принципах, довольно легко создать примитивную ЭС "Консультант по инвестициям" и попытаться давать советы инвесторам. Реальные экспертные системы такого типа успешно используются на финансовом рынке. Разумеется, они значительно больше нашей и содержат несколько сотен правил.

Примечание: эта задача потребует "двухступенчатого" принятия решения, то есть двух деревьев в системе RuleBook. Пример такого рода представлен в файле CHAPCEX.RB1.

Задача 3. Определитель растений

Создайте ЭС, помогающую определить, к какому семейству принадлежит найденное вами в лесу растение.

Задача 4. Как сделать выбор?

Создайте ЭС, помогающую выбрать наилучшую для вас (соответствующую вашим возможностям и потребностям) конфигурацию домашнего компьютера. (Варианты: выбрать магнитофон, автомобиль, платье, спутника жизни и т.п.)

Задания могут быть разными, в зависимости от интересов учеников. При этом желательно предлагать задания в такой последовательности:

1) четко поставленные задачи, где от учащихся требуется лишь "переложить" сформулированные правила на компьютер;
2) задачи, где учащиеся должны сами сформулировать правила на основании описания;
3) задачи, где очерчен лишь круг вопросов, на которые должна отвечать создаваемая ЭС, правила же создаются учащимися, использующими для этого свои знания в данной предметной области или жизненный опыт.

"Опасный" Интернет: мифы и реалии

Л.О. Сергеев

Несколько лет назад основной темой мифов компьютерной вирусологии были "разрушители" — страшные вирусы, выводящие из строя аппаратуру компьютера. Они "вводили в резонанс головки жестких дисков", "сжигали мониторы" и вообще пакостили как могли. Слухи о существовании таких вирусов не имели под собой никаких реальных оснований, что, впрочем, не ослабляло "энтузиазма" их распространителей. Лишь много позднее появился вирус, который при определенных условиях действительно может повредить материнскую плату компьютера (но опять же не винчестер и не монитор). В последние годы, знаменующиеся поистине революционным шествием по России "большого Интернета", "вирусосказатели" получили огромную аудиторию, с ужасом воспринимающую новые "страшилки", но и "вирусописатели", к сожалению, не замедлили воспользоваться новыми возможностями.

Самыми заметными и известными широкой публике фактами, связанными с темой "вирусы и Интернет", безусловно, являются вирус Морриса и "утка" GoodTimes (последняя мутировала бесчисленное число раз, что, впрочем, никак не отразилось на способности доверчивых пользователей ее кушать).

Вирус Морриса, хоть и является самым знаменитым компьютерным вирусом (по разным оценкам, написанное в разных странах и в разное время об этом вирусе на порядки превосходит написанное обо всех остальных вирусах, вместе взятых), в настоящий момент никому не угрожает. Это был "экспериментальный" (по утверждению автора, которому, впрочем, не удалось убедить присяжных в американском суде) вирус для Unix-систем, который умел только размножаться, чем и был опасен (от бесконтрольного размножения вируса сеть в короткое время оказалась перегружена и вышла из строя). Можно считать, что вирус этот был направлен против самой сети, до документов и программ пользователей ему дела не было.

"Утка" GoodTimes, которая впервые появилась в 1994 г., значительно более "опасна". Количество поверивших в "страшные" письма, часто рассылаемые от имени той или иной известной компьютерной фирмы (иногда и от имени антивирусных фирм), превосходит все разумные границы. Суть "предупреждения", которое содержится в этих письмах, такова: имеется вирус, который способен заразить ваш компьютер при чтении обычного (текстового) электронного письма. То есть такое письмо не только читать нельзя, но и получить опасно. Бойся сам и испугай другого! О том, что все это ерунда, много раз твердили миру различные авторитетные специалисты, профессионалы в области компьютерной вирусологии и защиты компьютерных сетей. Только кто же им поверит!

В этой короткой заметке мы дадим обзор *реальных* опасностей, которые могут подстерегать обычных пользователей Интернета. Мы совсем не затронем вопросы защиты серверов и корпоративных сетей — это тема более серьезного и развернутого обсуждения. Предположим, что наш читатель использует Интернет так, как сегодня предлагает большинство провайдеров: связь с сервером провайдера по протоколу PPP (реже SLIP), работа с электронной почтой по протоколам POP и SMTP. (Проще говоря, если вы работаете с Интернетом дома, связываетесь с провайдером посредством модема, для просмотра web-страниц используете какую-нибудь версию Internet Explorer или Netscape Navigator, а для работы с электронной почтой, например, Outlook Express, то это — ваш случай.)

Опасность № 1 – вложенные файлы

Рассмотрим типичную ситуацию: вы получили электронное письмо, во вложении к которому находится документ Microsoft Word. Например, с безобидным названием LETTER.DOC. Конечно, вы захотите поскорее познакомиться с содержимым файла, благо при использовании большинства современных почтовых программ для этого достаточно щелкнуть мышкой на имени файла. И ЭТО — ОШИБКА! Если в файле содержится макрокомандный вирус (а вирусы этого типа, заражающие документы Microsoft Word, Microsoft Excel и ряда других популярных систем документооборота, получили в последнее время огромное распространение), он немедленно заразит вашу систему. (В последних версиях систем, которые могут подвергнуться вирусной атаке, имеется проверка на наличие макрокоманд в открываемых

файлах, но это слабое утешение. Во-первых, данную проверку можно отключить. Во-вторых, файл может содержать вполне "мирные" макрокоманды: не отключать же их "на всякий случай".) Так что же делать с вложенным документом? **Потратить несколько лишних секунд: сохранить его на диске, проверить антивирусной программой *последней версии* и только потом, если вирусов в файле не обнаружено, открывать.** Есть другие решения. Например, имеются антивирусные программы, осуществляющие автоматическую проверку приходящей электронной почты. Если вы используете программу-сторож, то при открытии зараженного файла обязательно получите предупреждение (точнее, вам просто не дадут открыть зараженный файл). Ведь сторожу совершенно безразлично, напрямую вы открываете документ или "из-под" почтовой программы.

Опасность № 2 – троянские программы

Известные троянские программы, распространяющиеся через Интернет, по существу, представляют собой утилиты для удаленного администрирования компьютера. Проще говоря, посредством такой программы злоумышленник может получить доступ к вашему компьютеру и выполнять на нем различные операции (практически любые) без вашего ведома и участия.

Характерным представителем троянцев описанного типа является программа Back Orifice (BO). Back Orifice является системой удаленного администрирования, позволяющей пользователю контролировать компьютеры при помощи обычной консоли или графической оболочки. "В локальной сети или через Интернет BO предоставляет пользователю больше возможностей на удаленном Windows-компьютере, чем имеет сам пользователь этого компьютера", — это текст из "рекламного" объявления на одной из хакерских web-страниц. Разумеется, возможность удаленного администрирования вашего компьютера представляет серьезную опасность, но не такую большую, как кажется на первый взгляд. Обычные пользователи проводят в Сети не так много времени (качество телефонных линий этому, надо сказать, способствует), да и что такого "интересного", с точки зрения хакера, можно сделать на вашем компьютере? Оказывается, можно, только администрировать ничего не надо. Существенный интерес для хакера представляют пароли, которые вы используете для работы с сервером провайдера. Заполучив пароль, хакер может запросто "просадить" все ваши денежки (отметим, что так поступают неосторожные хакеры, осторожные постараются работать таким образом, чтобы вы максимально долго оставались в неведении). К счастью, троянцев, умеющих выполнять указанные функции, довольно мало и все они успешно детектируются антивирусными программами.

Опасность № 3 – HTML-вирусы

Вирусы данного типа встречаются редко, так что информация о них представляет скорее "академический" интерес, нежели практический. Суть такова: на самом языке HTML, который используется для разметки гипертекстовых документов, никакие вирусы, конечно, написать нельзя. Но для создания динамических страниц, организации взаимодействия с пользователем и прочих действий используются программные вставки (скрипты) в HTML-документы. Известные HTML-вирусы используют скрипты, написанные на языке Visual Basic. С их помощью они находят HTM- и HTML-файлы на локальной машине и записываются в них. Иногда такие вирусы как-нибудь проявляют себя (например, выводят MessageBox). Малому распространению вирусов данного типа (равно как и малому их числу) способствует то, что при стандартных настройках броузера выполнение "опасных" скриптов (а к таковым относятся и те, в которых происходят обращения к файлам локального компьютера) запрещено. Обычные же, "безопасные", скрипты не могут производить описанные манипуляции.

Опасность № 4 (гипотетическая) – Java-вирусы

В настоящее время известны два вируса, написанные на языке Java. Опасности они практически не представляют. Кратко поясним, в чем суть. Исполняемые модули программ, написанных на Java (CLASS-файлы), бывают двух типов: приложения и апплеты. Приложения выполняются под управлением интерпретатора и являются почти обычными программами (почти, ибо имеются все же некоторые ограничения, например, в области работы с памятью). Апплеты, в отличие от приложений, могут выполняться под управлением броузеров, но на них накладываются значительно более серьезные ограничения для обеспечения безопасности: апплеты, в частности, не имеют почти никакого доступа к файловой системе локального компьютера (кстати, в отличие от случая со скриптами, отключить данное ограничение в броузере невозможно). Таким образом, Java-вирусы могут быть оформлены только как приложения и для подавляющего большинства пользователей опасности не представляют.

Тривиальное решение

Д.М. Файликов

Эту историю ученику 7-го класса Пете Файликову рассказал его дедушка, бывший программист Файликов-старший.

Утром 1 апреля 1972 года Файликов-старший (а тогда он был просто Дмитрием Файликовым), которого с чьей-то легкой руки все в заводском вычислительном центре называли "программистом Божьей милостью", был вызван к директору завода Корневому. В кабинете присутствовало все заводское начальство. Директор был краток.

— Товарищ Файликов, — сказал он,— цилиндры, которые вы видите у меня на столе, — это четверть всей нашей продукции. Но под эту четверть занята половина станков нашего завода. Так что, товарищ Файликов, вот вам конкретная задача. Подумайте и смоделируйте на ЭВМ возможные варианты, прорешайте их и к 1 июля дайте предложения, как нам организовать производство цилиндров, чтобы уменьшить число задействованных под них станков.

Через полмесяца Файликов знал производство цилиндров не хуже главного инженера и главного технолога завода. Он работал с таким вдохновением, которого не испытывал уже давно. Задача была многомерной, и поэтому программа для ее решения оказалась довольно сложной. Пришлось использовать метод динамического программирования, метод Монте-Карло и метод наименьших квадратов. Для наглядности в программе была предусмотрена выдача результатов на экран (тогда это не было естественным, как сейчас).

30 июня программа была отлажена, и утром следующего дня, когда Файликов еще только приступал к решению задачи, в машинный зал вычислительного центра нагрянуло все начальство. Файликов волновался, как абитуриент на экзамене, когда запускал программу на выполнение. Наконец что-то щелкнуло, запищало, и на экране дисплея появилась надпись: для производства цилиндров использовать станков — 0, время работы первого станка — 0, второго станка — 0... И вскоре весь экран заполнился одними нулями.

Наступило гробовое молчание, которому, казалось, не будет конца.

— Это, — с трудом выдавил из себя Файликов, — тривиальное решение... Я сейчас...

Он суетливо стал нажимать на какие-то клавиши, вновь что-то щелкнуло, запищало, нули с экрана исчезли, и появилась новая надпись: второго решения нет.

Молчание прервал голос директора завода Корневого.

— Вычислительный центр, кажется, ваше хозяйство, — раздраженно обратился он к главному инженеру Каталогову. — Разберитесь же, что у вас за программисты и чем они занимаются.

Всю ночь Файликову снились кошмары, а утром он шел на завод, неся в своем портфеле заявление об уходе с работы. Файликов уже подходил к зданию вычислительного центра, когда вдруг увидел спешащего к нему главного инженера.

— Сколько же можно ждать тебя? На, прочти. — И Каталогов протянул ему какую-то бумагу. Это был приказ по министерству, предписывающий заводу с 1 июля прекратить производство цилиндров как нерентабельное.

На досуге

Подготовил В.Г. Федоринов

Прояви терпение, внимание и запиши слова из букв следующих слов:

1. Директория.
2. Редактор.
3. Архивирование.
4. Сумматор.
5. Фортран.
6. Роботрон.
7. Корректировка.

1. Из слова ДИРЕКТОРИЯ можно составить следующие слова:

Директор, диктор, дитя, до, док, дротик, ректор, рот, рекорд, род, ре, рея, ряд, ритор, ерик, кот, крот, кедр, кредо, кит, корт, код, тор, тир, трек, ток, тик, як, яр, яд, ядро.

2. РЕДАКТОР:

Рак, река, ректор, рокер, рекорд, рот, ре, род, рок, рота, еда, едок, дар, дека, до, ар, акр, ад, кот, код, кадр, корт, кеда, кета, кора, кредо, кар, кедр, крот, ток, тор, трек, трак, ода, орда.

3. АРХИВИРОВАНИЕ:

Архив, арена, ар, ров, рана, ре, раввин, хор, верх, вар, вера, вино, вена, вор, варан, веха, охрана, охра, овин, овен, нива, нерв, нрав.

4. СУММАТОР:

Сумма, сума, сом, сорт, смотр, сор, сота, мат, март, мусор, мост, мор, ум, утро, ус, ар, тор, торс, трус, том, оса, омар, рот, рост, рота, ром, роса.

5. ФОРТРАН:

Форт, франт, фора, фронт, фтор, фон, ор, рот, рота, рант, тор, трон, торф, ар, нота, нора.

6. РОБОТРОН:

Робот, рот, ротор, борт, бор, бот, трон, тон, тор, оборот, ор.

7. КОРРЕКТИРОВКА:

Корт, кора, ковка, кот, квота, корректор, кит, кров, крот, кок, кета, кирка, корка, корова, кратер, кивер, крокет, котировка, кивок, крик, ре, ров, река, ректор, рок, рота, рот, ритор, реактор, реактив, рак, рокировка, рокот, тор, тир, ток, трак, трек, террор, око, отрок, вор, вар, веко, вера, век, ива, икра, икар, ар, акр, актив.

Первый персональный компьютер

Окончание. Начало на стр. 1

Довольно долго ЭВМ были очень большими устройствами. Каждая машина занимала целый зал и обслуживалась группой сотрудников (инженеров, техников, операторов), работавших в две-три смены. Первый шаг к уменьшению размеров компьютеров стал возможен после изобретения в 1948 г. транзисторов — миниатюрных электронных приборов, которым удалось заменить в ЭВМ электронные лампы. Компьютеры, основанные на транзисторах (то есть ЭВМ второго поколения), появились во второй половине 1950-х годов.

Немного позже стали производиться так называемые малые ЭВМ. В 1965 году была выпущена ЭВМ “МИР” (машина для инженерных расчетов), которая могла разместиться в небольшой комнате. Пользователь садился за стол с электрифицированной пишущей машинкой (с ее помощью осуществлялись ввод и вывод информации), включал компьютер, решал свою задачу и выключал его [1]. Для этой ЭВМ употреблялся язык программирования Алмир-65, представляющий собой “русифицированное развитие” языка Алгол-60. В более поздней модели машины — “МИР-1” (1968) предусматривался ввод с перфоленты и вывод на нее.

В 1969 г. появилась ЭВМ “МИР-2”, в которой впервые был применен дисплей со световым пером (обеспечивающий оперативный вывод, контроль, редактирование информации и отображение на экране промежуточных и окончательных результатов решения задач), а также внешняя память на магнитных картах. Здесь использовался язык Аналитик — расширение языка Алмир.

Машины семейства “МИР” широко применялись в научно-исследовательских учреждениях и вузах. Освоить приемы работы на компьютерах данного семейства удавалось за 6—10 часов, причем программы для простейших задач студенты начинали самостоятельно составлять уже в течение первого часа знакомства с языком и пультом управления.

По сути дела, ЭВМ “МИР” представляла собой персональный компьютер. Во всяком случае для нее выполнялись три важных требования, предъявляемых к персональной ЭВМ [2]:

- возможность индивидуальной работы на компьютере без каких-либо посредников;
- простота использования, обеспечиваемая путем взаимодействия с машиной в режиме диалога;
- достаточно высокая надежность.

Литература

1. Уроки программирования // Вычислительная техника и ее применение. № 10/89.

2. *В.Э. Фигурнов.* IBM PC для пользователя. Изд. 6-е, перераб. и доп. М.: ИНФРА-М, 1995.

На фотографиях: ЭВМ “МИР-1”, “МИР-2”, “МИР-3”.

Гл. редактор
С.Л. Островский
Зам. гл. редактора
Е.Б. Докшицкая
Редакция:
Н.Л. Беленькая,
Н.П. Медведева
Дизайн и компьютерная верстка:
Н.И. Пронская
Корректоры:
Е.Л. Володина,
С.М. Подберезина

выходит четыре раза в месяц
При перепечатке ссылка
на ИНФОРМАТИКУ обязательна,
рукописи не возвращаются

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

Учредитель: ООО “Чистые пруды”
Регистрационный номер 012868
Отпечатано в типографии ОАО ПО “Пресса-1”.
125865, ГСП, Москва, ул. Правды, 24.
Тираж 5000 экз.
Заказ №

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта приложений 32744

Тел. (095)249 3138, 249 3386. Факс (095)249 3184

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