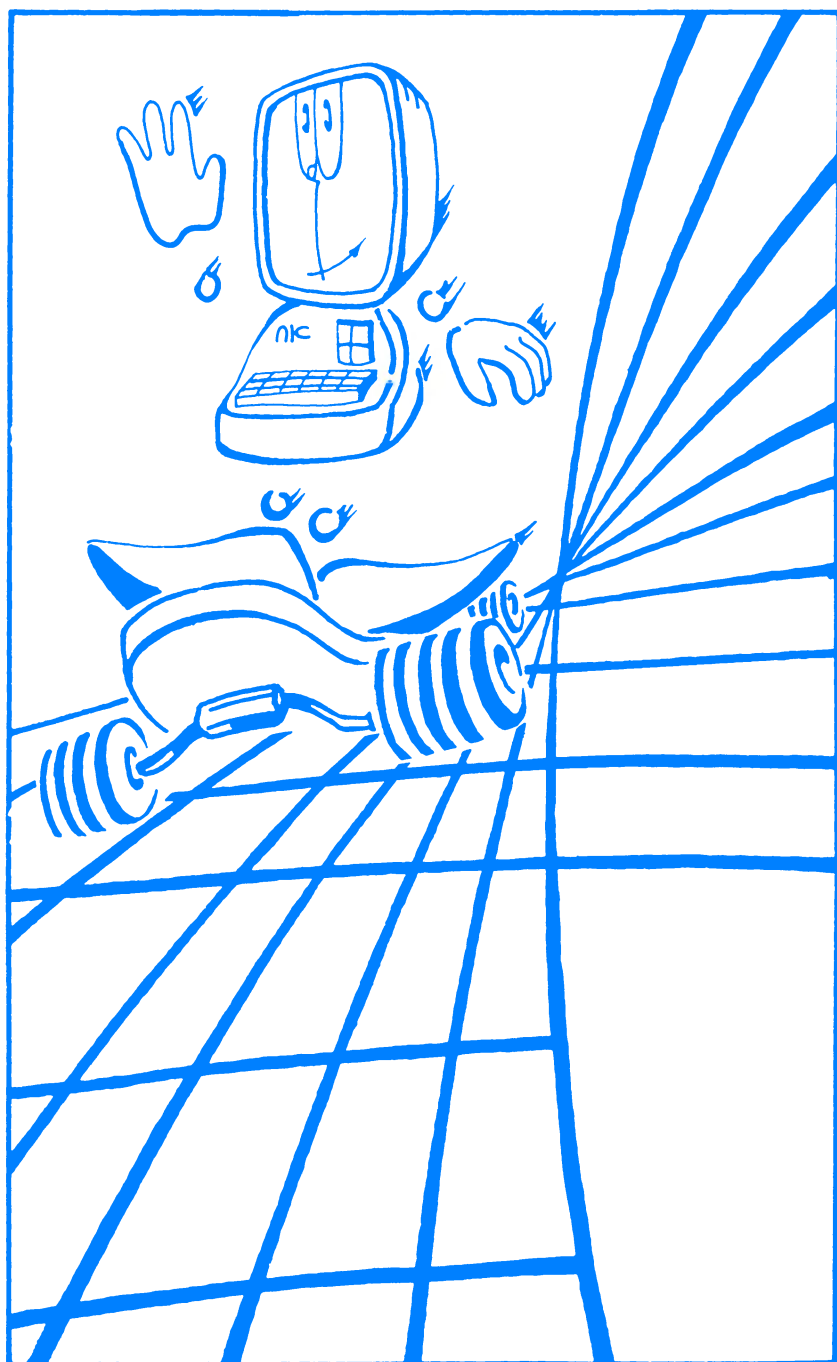




В. Н. КАСАТКИН

ЧЕРЕЗ ЗАДАЧИ- К ПРОГРАММИРОВАНИЮ

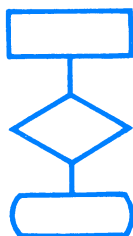




СЕРИЯ „КОГДА СДЕЛАНЫ УРОКИ“

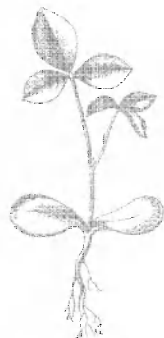
В. Н. КАСАТКИН

ЧЕРЕЗ ЗАДАЧИ— К ПРОГРАММИРОВАНИЮ



Для старшего школьного возраста

КИЕВ „РАДЯНСЬКА ШКОЛА“ 1989



Scan AAW

ББК 22.18
К28

Рецензенты: *И. Н. Антипов*, доктор педагогических наук, профессор, заведующий лабораторией обучения программированию и ЭВМ НИИ школ МП РСФСР;
Р. В. Фрейвалд, доктор физико-математических наук, заместитель директора по научной работе Вычислительного центра Латвийского государственного университета им. П. Стучки.

Художник-оформитель *С. Н. Кобрин*
Редактор *В. Н. Кириченко*

Касаткин В. Н.

К28 Через задачи — к программированию: Для ст. шк. возраста—К.: Рад. шк., 1989.— 128 с.— (Сер. «Когда сделаны уроки»).

ISBN 5—330—00741—0.

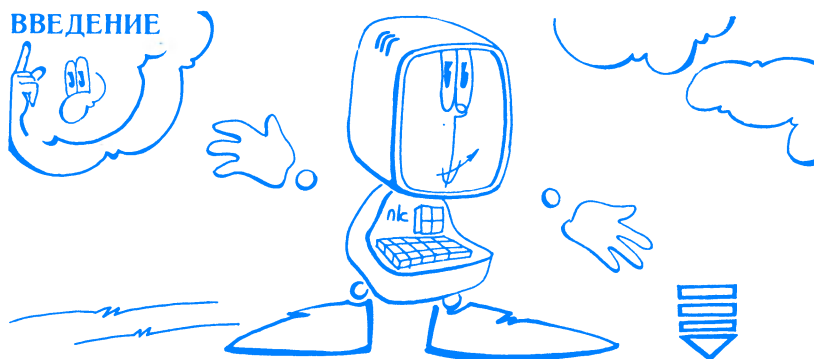
В книге на примере занимательных задач рассматриваются теоретические и практические основы программирования (машинного и безмашинного). Предлагаемые задачи использовались в практике работы Малой академии наук «Искатель» и полностью соответствуют интересам учащихся. При решении задач раскрываются существенные связи знакомого учащимся математического материала с важнейшими идеями программирования и основами компьютерной грамотности.

К $\frac{4802020000-278}{M210(04)-89}$ 342—89

ББК 22.18

ISBN 5—330—00741—0

© Издательство «Радянська школа»,
1989



Через задачи — к программированию — это значит через яркие примеры к знакомству с необычными и поучительными приемами, с изюминками искусства программирования.

В предлагаемом школьнику сборнике занимательных задач есть задания на разработку вычислительных алгоритмов, задания, выполняя которые, ЭВМ обратится к средствам обработки символической информации, продемонстрирует умение решать грамматические задачи. Несколько геометрических задач иллюстрируют большие графические возможности ЭВМ. Уделено внимание и логическим задачам.

Все программы написаны на языке Бейсик (версия MSX). Конечно, наибольшее впечатление при изучении заданий будет достигнуто, если читатель может использовать ЭВМ. В этом случае школьник вместе с машиной совершит путешествие в удивительный мир программирования. На его глазах машина вычислит гигантское число $100!$, построит сложные геометрические фигуры, выступит в роли справочного бюро, партнера в игре и умело разберется в хитросплетениях логической задачи.

Задания можно изучать в любом порядке. В каждой задаче читатель встретит сначала разъяснения алгоритма и лишь после этого приводится текст программы. Некоторые из средств Бейсика разъясняются, однако на книгу нельзя смотреть как на книгу учебную. Цель книги — расширить представление о круге средств, применяемых для решения задач в сотрудничестве с ЭВМ.

Завершая вступительный разговор с читателем, подчеркнем, что сотрудничество с ЭВМ требует от человека особой собранности, ясности мысли, настойчивости. Читателя книги ждут задачи занимательные, однако решение их является делом серьезным и поэтому следует настроиться не на легкое чтение, а на работу с книгой с карандашом в руках. В каждой задаче сначала требуется понять, в чем идея решения, в чем суть алгоритма. Это требует настойчивости, понимания математических вопросов.

Лучшим вариантом работы с книгой является, несомненно, чтение «вместе с ЭВМ». Очень хорошо, если читатель сможет самостоятельно проверить все программы. Это не только прибавит опыта в работе за пультом ЭВМ, это может привести к открытиям. Школьник может увидеть иной подход в формировании текста программы, может найти ошибку или недочет в приводимых алгоритмах.

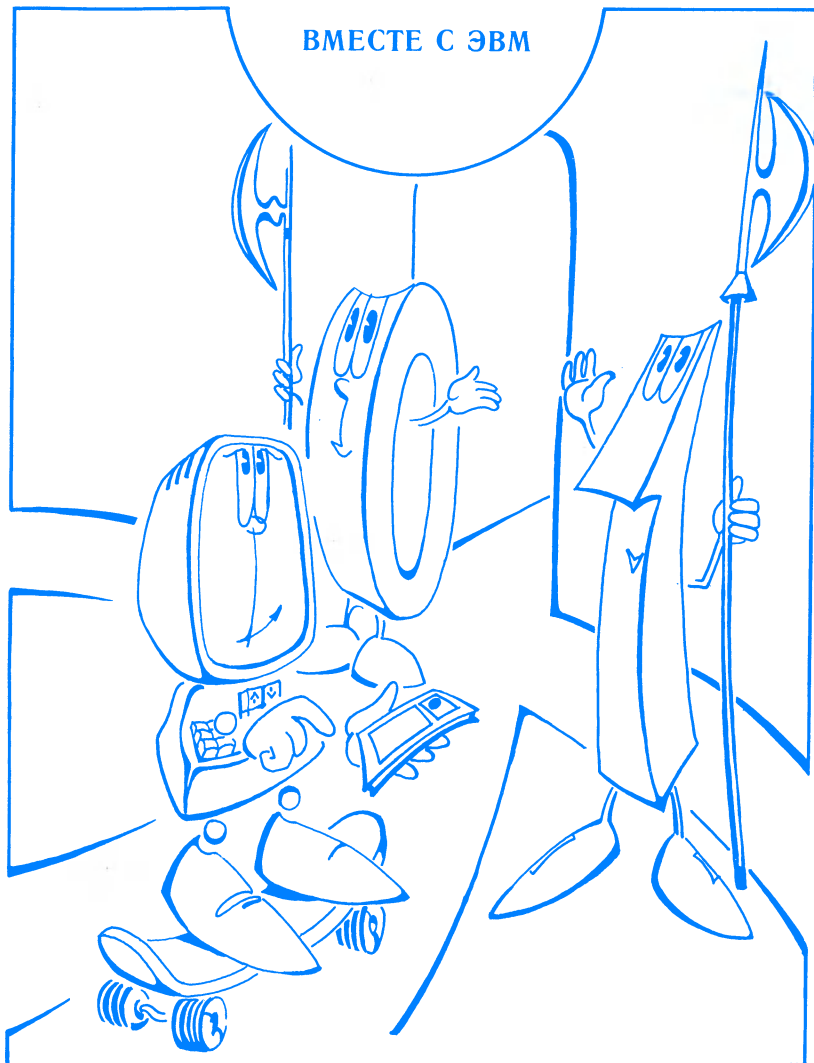
Очень полезно проверить программы для различных исходных данных. Неплохо, если читатель сумеет организовать исследование задачи, применив для этого ЭВМ. Дело в том, что многие программы можно обобщить, преобразовать так, чтобы решать аналогичные задачи. И, конечно, совсем хорошо, если читатель сможет предложить новый алгоритм — это повлечет и новый текст программы, а, возможно, и более эффективное решение задачи в целом.

Большинство предложенных в книге задач было решено членами Малой академии наук школьников Крыма «Искатель».

ГЛАВА

I.

ВЫЧИСЛЯЕМ
ВМЕСТЕ С ЭВМ



КАК НАЙТИ СУММУ? СКОЛЬКО ЦИФР В ЧИСЛЕ 100!?
ОСНОВАНИЕ СИСТЕМЫ СЧИСЛЕНИЯ — МИЛЛИОН. ШАХ-
МАТНОЕ ЧИСЛО НА ЭКРАНЕ ЭВМ. ЗАДАЧА О ДЕШИФРА-
ТОРЕ. ВЫЧИСЛЯЕМ В СОК. ЧИСЛА ФИБОНАЧЧИ. ЧИСЛА
БЕРНУЛЛИ. ЧИСЛА КАТАЛАНА. ЧИСЛА МЕРСЕННА.
ГЕНЕРАТОР ПРОСТЫХ ЧИСЕЛ.

*Не бойтесь начинать програм-
мирование сначала, не жалеете
времени на анализ алгоритма.*

*Из фольклора
программистов*

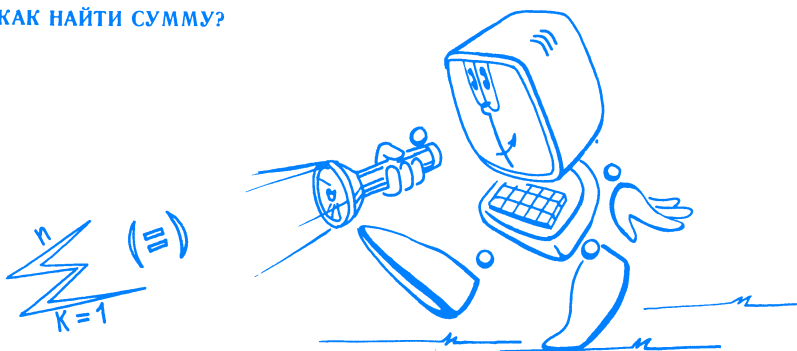
Вычисления с помощью ЭВМ — это не только и не столько возрастание скорости счета. Это изменение взглядов на вычисления, появление новых подходов к их организации. Гордость математики — алгоритмы в виде формул начинают сотрудничать с алгоритмами в форме программ. Программа вычислений, написанная для ЭВМ, — это, подчас, новый и более общий способ задания алгоритма, чем формула.

Известно, что ЭВМ обладает ограниченным набором внутренних средств и эти средства очень просты. Однако изобретательность программиста и разработчика алгоритмов может творить чудеса. В сотрудничестве человека и ЭВМ — залог успеха в решении многих практических задач.

Предлагаемые задачи относятся к разделу: обработка числовой информации на ЭВМ. Однако вычислительные задачи представлены в необычном ракурсе. В каждой задаче подмечены трудности, с которыми встретится человек, если будет решать эти задачи вручную. Рассказывается о том, как следует воспользоваться техническими возможностями ЭВМ и преодолеть трудности.

Может быть, некоторые задачи покажутся отдаленными от школьной математики, например, рассказ о системе счисления остаточных классов. Но, если Вы прочтете рассказ, то увидите, что для понимания сути дела достаточно знаний по математике для седьмого класса. Роль же новых понятий для освоения искусства программирования необычайно велика.

Подчеркнем, что рассказ идет о широко известных проблемах элементарной математики, о знаменитых задачах. То, что эти задачи сегодня решает ЭВМ, — это новый шаг в математическом прогрессе. Разобраться в методах подготовки к решению задачи на ЭВМ — важная часть современного математического образования. Это особенно важно для школьников.



Сложение чисел, на первый взгляд, кажется делом очень простым. Действительно, если каждое число, входящее в сумму, известно или может быть вычислено непосредственно перед суммированием, то все сводится к прибавлению нового числа к уже накопленной сумме. Действуя последовательно, можно вычислить сумму любого количества слагаемых:

$$\sum_{k=1}^n k = 1 + 2 + 3 + \dots + n; \quad \sum_{k=1}^n (2k-1) = 1 + 3 + 5 + \dots + (2n-1);$$

$$\sum_{k=1}^n k^2 = 1^2 + 2^2 + 3^2 + \dots + n^2; \quad \sum_{k=1}^n (2k-1)^2 = 1^2 + 3^2 + 5^2 + \dots + (2n-1)^2;$$

$$\sum_{k=1}^n k^3 = 1^3 + 2^3 + 3^3 + \dots + n^3; \quad \sum_{k=1}^n (2k-1)^3 = 1^3 + 3^3 + 5^3 + \dots + (2n-1)^3.$$

Задача, однако, приобретает новое качество, становится трудной, если n — число слагаемых — велико. Математики приложили немало сил и проявили большую изобретательность, разыскивая общие формулы для вычисления сумм n первых членов последовательностей. Наиболее известными из таких формул являются формулы суммирования n первых членов арифметической и геометрической прогрессий. Напомним их:

$$S_n = \frac{a_1 + a_n}{2} \cdot n \text{ — для арифметической прогрессии,}$$

$$S_n = \frac{a_1(q^n - 1)}{q - 1} \text{ — для геометрической прогрессии.}$$

Используя эти общие формулы, можно получить формулы для вычисления конкретных сумм:

$$\sum_{k=1}^n k = 1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2};$$

$$\sum_{k=1}^n (2k-1) = 1 + 3 + 5 + \dots + (2k-1) = n^2.$$

Менее известны формулы для вычисления:

$$\sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6} \quad \text{— суммы квадратов первых } n \text{ натуральных чисел;}$$

$$\sum_{k=1}^n k^3 = \frac{n^2(n+1)^2}{4} \quad \text{— суммы кубов первых } n \text{ натуральных чисел;}$$

$$\sum_{k=1}^n (2k-1)^2 = \frac{n(4n^2-1)}{3} \quad \text{— суммы квадратов первых } n \text{ нечетных чисел;}$$

$$\sum_{k=1}^n (2k-1)^3 = n^2(2n^2-1) \quad \text{— суммы кубов первых } n \text{ нечетных чисел.}$$

Последние четыре формулы уже нельзя получить из выше-приведенных формул для суммирования членов арифметической и геометрической прогрессий, так как ни квадраты, ни кубы первых натуральных и первых нечетных чисел прогрессий не образуют. Такие формулы разыскиваются специально, и это важная математическая работа. Во-первых, эти формулы чрезвычайно упрощают суммирование членов некоторых последовательностей, а во-вторых,— помогают в анализе тех числовых последовательностей, члены которых суммируются. Так, например, располагая общей формулой суммирования n первых членов последовательности, можно дать точный ответ на вопрос, имеет ли бесконечная сумма чисел такого типа предел или не имеет.

Отыскание формул для суммирования членов последовательности требует фантазии и огромного труда. Некоторые формулы просто поражают. Разве не вызывает уважение и удивление общая формула для вычисления суммы N -х степеней первых n натуральных чисел:

$$\sum_{k=1}^n k^N = \frac{1}{N+1} \sum_{k=1}^n C_{N+1}^k (n+1)^{N+1-k} \cdot B_k ?$$

В этой формуле B_k — так называемое *число Бернулли* (рассказ об этих числах будет ниже). Заметим, что из этой общей формулы при $N=2$ и $N=3$ можно получить, как частный случай, формулы, приведенные выше.

Не менее интересна формула:

$$\sum_{k=1}^n \frac{1}{k^{2N}} = \frac{1}{1^{2N}} + \frac{1}{2^{2N}} + \dots + \frac{1}{n^{2N}} = \frac{(-1)^{N+1} \cdot (2\pi)^{2N}}{2 \cdot (2N)!} \cdot B_{2N}.$$

К сожалению, такие формулы — редкость — и поэтому задача суммирования до появления ЭВМ считалась задачей отнюдь

не простой. Появление ЭВМ изменило наши представления о суммировании чисел, были найдены неожиданные подходы, поражающие своей простотой и эффективностью.

Рассмотрим сначала *суммирование членов числовых последовательностей* $\{a_n\}$. Несколько общих замечаний:

1) суммированию подвергаются члены числовой последовательности, каждый из которых ЭВМ вычисляет как значение функции $FNZ(k)$, где $k \in N$, ($k=1, 2, \dots$). Находится сумма:

$$S_n = a_1 + a_2 + a_3 + \dots + a_n,$$

или

$$S_n = FNZ(1) + FNZ(2) + FNZ(3) + \dots + FNZ(n);$$

2) функция $FNZ(k)$ задается человеком, который поручает ЭВМ выполнение суммирования.

Примеры

$$FNZ(k) = k,$$

$$FNZ(k) = 2k - 1,$$

$$FNZ(k) = (2k - 1)^2 \dots$$

3) суммирование осуществляется «шаг за шагом»: сначала полагают равной нулю исходную сумму ($S=0$), затем одно за другим к ней прибавляют вычисленные ЭВМ слагаемые ($FNZ(i)$). Число слагаемых n задается до начала суммирования. На рисунке 1 приведена схема алгоритма.

Здесь N — число слагаемых, задается до начала суммирования; $FNZ(k)$ — формула общего члена последовательности; как функция от $k \in N$, задается до начала суммирования.

Располагая схемой алгоритма, можно выписать соответствующую программу на языке Бейсик.

```
5 REM СУММИРОВАНИЕ ЧЛЕНОВ ПОСЛЕДОВАТЕЛЬНОСТИ
10 LET S=0:LET K=1
15 INPUT "N=";N
20 DEF FNZ(K)=2*K-1
25 LET S=S+FNZ(K)
30 LET K=K+1
35 IF K<=N THEN 25
40 PRINT "S=";S;"-ИСКАЯ СУММА"
45 END
```

Пример программы приведен на рисунке 2.

Работая по такой программе, ЭВМ найдет сумму N чисел, каждое из которых будет значением функции $FNZ(k)$. Если, например, $FNZ(k) = a \cdot k + b$, то суммироваться будут члены арифметической прогрессии с первым членом, равным $(a+b)$ и разностью, равной a .

Отметим важную особенность суммирования по приведенной программе. Сумму чисел вида $FNZ(k)$ можно вычислять, начиная не с первого члена последовательно, а, например, с седьмого

или десятого. Для этого достаточно начальное значение k считать равным не единице, а семи или десяти. В программе вместо оператора $LET K=1$, следует записать оператор $INPUT «K=»$; K — этим будет достигнута большая общность в решении задачи.

Еще одна важная особенность состоит в возможности наращивать значения K различными числами. Речь идет о том, что оператор $LET K=K+1$ можно заменить на более общий: $LET K=K+D$. При $D=2$ это позволит *суммировать члены последовательности с нечетными номерами*, то есть вычислить сумму:

$$S = FNZ(1) + FNZ(3) + FNZ(5) + \dots$$

Легко обнаружить еще одну возможность обобщения программы. Обратим внимание на оператор $LET S=S+FNZ(K)$, в который можно внести изменения, записав вместо $FNZ(K)$ степень: $FNZ(K)^R$. Теперь при $R=2$ ЭВМ будет *суммировать уже квадраты членов некоторой последовательности*, а при $R=3$ — *кубы членов последовательности*.

Все сказанное приводит к более общему тексту программы:

```
100 REM СУММИРОВАНИЕ ЧИСЕЛ(1)
110 LET S=0
120 INPUT "N-ЗАДАЕТСЯ ЧИСЛО СЛАГАЕМЫХ";N
130 INPUT "K-ЗАДАЕТСЯ НАЧАЛЬНОЕ ЗНАЧЕНИЕ АРГУМЕНТА";K
140 INPUT "D-ЗАДАЕТСЯ ПРИРАЩЕНИЕ АРГУМЕНТА";D
150 INPUT "R-ЗАДАЕТСЯ СТЕПЕНЬ СЛАГАЕМОГО";R
160 DEF FNZ(K)=2*K-1
170 LET S=S+(FNZ(K))^R
180 LET K=K+D
190 IF K<=N THEN 170
200 PRINT "S=";S"-ИСКАЯ СУММА"
210 END
```

Если учесть, что в языке Бейсик имеется очень сильный *оператор цикла*, оператор FOR — $NEXT$, то приведенная программа может быть записана более кратко:

```
1 REM СУММИРОВАНИЕ ЧИСЕЛ(2)
5 LET S=0
10 INPUT N,K,D,R
15 DEF FNZ(K)=2^K
25 FOR I=K TO N STEP D
35 LET S=S+(FNZ(I))^R
45 NEXT I
55 PRINT "S=";S
65 END
```

Программа суммирования чисел (2) обеспечивает *сложение чисел* — членов данной последовательности $\{a_n\}$, сгруппированных различным способом:

- $a_1 + a_2 + a_3 + \dots + a_{20}$ — сумма первых 20 членов;
- $a_1 + a_3 + a_5 + \dots + a_{31}$ — сумма первых 16 членов, стоящих на нечетных местах;
- $a_7 + a_{11} + a_{15} + \dots + a_{35}$ — сумма членов последовательности, вычисленная при $k=7$, $D=4$, $N=8$.

Столь большой общности в решении задач, связанных с суммированием, которая выражается программой, алгебра предложить не может. Если текст программы образно назвать формулой, то столь общих формул в алгебре нет.

Прежде чем рассмотреть суммирование в более специальных случаях, еще раз подчеркнем, что выше рассматривались случаи суммирования членов последовательности, каждый из которых мог быть вычислен перед его включением в сумму. Иначе говоря, $a_k = FNZ(k)$ всегда было функцией от номера члена последовательности (k) и вычислялось независимо для любого заданного k .

Существует, однако, немало практически важных числовых последовательностей, члены которых не могут быть вычислены как значения функции только от номера члена в последовательности. Речь идет о последовательностях, задаваемых *рекуррентными формулами*. Напомним, что при рекуррентном способе задания последовательности вычисление следующего члена требует знания значений предшествующих членов. Примером может быть последовательность *чисел Фибоначчи*: 1; 1; 2; 3; 5; 8; 13; 21; 34; ... , задаваемая формулами $a_1 = a_2 = 1$, $a_n = a_{n-1} + a_{n-2}$ для $n \geq 3$.

Пусть необходимо найти сумму n первых членов последовательности Фибоначчи. Программа может быть такой, как приводимая ниже:

```

10 REM СУММИРОВАНИЕ N ПЕРВЫХ ЧИСЕЛ ФИБОНАЧЧИ
20 INPUT "N=";N
30 DIM A(N)
40 LET A(1)=1:LET A(2)=1:LET S=2
50
60 FOR I=3 TO N
70 LET A(I)=A(I-1)+A(I-2)  -вычисляется a_i
80 LET S=S+A(I)           -наращивается сумма
90 NEXT I
100 PRINT S
110 END

```

Иногда вычисление очередного слагаемого требует столь громоздких вычислений, что целесообразно создавать *подпрограмму*. Проиллюстрируем это на примере вычисления *суммы биномиальных коэффициентов*. Напомним, что биномиальным коэффициентом называется число $C_n^m = \frac{n!}{m!(n-m)!}$. Чтобы вычислить C_n^m , необходимо создать предварительно подпрограмму для вычисления *факториала числа*.

Вариант подпрограммы может быть таким:

```

1 REM ФАКТОРИАЛ ЧИСЛА K
5 INPUT K
10 LET F=1
15 FOR I=1 TO K
20 LET F=F*I
25 NEXT I
30 PRINT "F=";F
35 END

```

Теперь нетрудно создать программу для вычисления биномиального коэффициента C_n^m , она может быть такой:

```

10 INPUT N, M
15 LET K=N
20 GOSUB 65
25 LET A=F
30 LET K=M
35 GOSUB 65
40 LET B=F
45 LET K=N-M
50 GOSUB 65
55 LET C=A/(B*F)
57 PRINT "Искомый биномиальный коэфф."; C
60 END
65 LET F=1
70 FOR I=1 TO K
75 LET F=F*I
80 NEXT I
85 RETURN

```

Располагая программой вычисления C_n^m для различных n и m , можно создавать программу суммирования биномиальных коэффициентов, например, для вычисления суммы:

$$C_{10}^1 + C_{10}^2 + C_{10}^3 + C_{10}^4 + C_{10}^5$$

или

$$C_{12}^1 + C_{12}^3 + C_{12}^5$$

Предлагается очень красивая программа. В этой программе, которую можно назвать **суммирование биномиальных коэффициентов**, используются две входящие в нее подоперации: **вычисление факториала числа** и **вычисление биномиального коэффициента**. Схематически это можно представить так, как показано на рисунке 3.

Пусть для примера необходимо вычислить сумму:

$$C_{10}^1 + C_{10}^2 + C_{10}^3 + C_{10}^4 + C_{10}^5$$

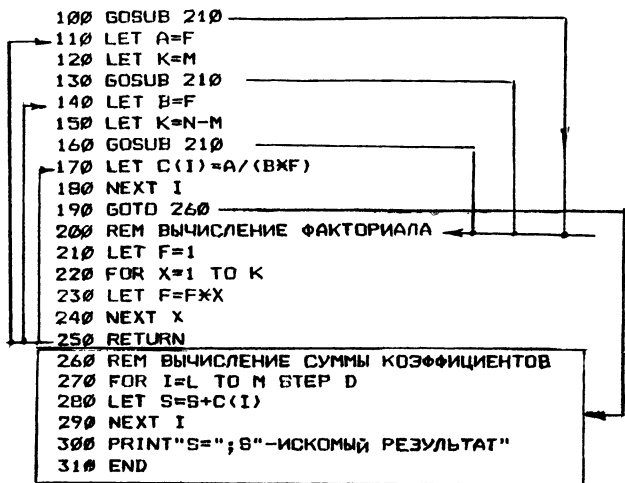
Еще раз обратимся к вспомогательной схеме (см. рис. 4).

Обращение к подпрограмме и возврат в основную программу осуществляется с помощью оператора передачи управления с возвратом: GOSUB — RETURN. Текст программы именно для этого случая может быть таким, как приведенный ниже:

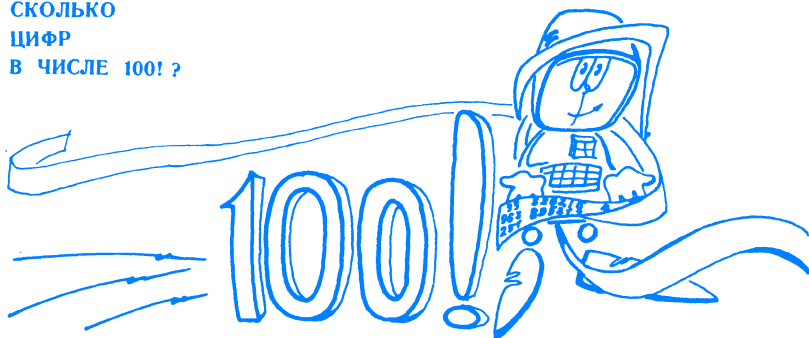
```

10 REM СУММИРОВАНИЕ БИНОМИАЛЬНЫХ КОЭФФИЦИЕНТОВ
20 INPUT N
30 INPUT M
40 INPUT L
50 INPUT D
60 LET S=0: DIM C(M)
70 FOR I=L TO M STEP D
80 REM ВЫЧИСЛЕНИЕ БИНОМИАЛЬНОГО КОЭФФИЦИЕНТА
90 LET K=N

```



СКОЛЬКО
ЦИФР
В ЧИСЛЕ 100! ?



Кажется, на этот вопрос получить ответ легко. В самом деле, выше была составлена и успешно применялась программа **«Вычисление факториала числа N »**. Работая по такой программе, ЭВМ вычислит и выведет на экран или напечатает число $N!$ Нам останется только подсчитать число цифр в выведенном машинной числе. Но попытки применить этот подход к большим числам приводят к неудачам — дело в том, что ЭВМ в большинстве своем не могут вывести на экран число, содержащее более пятнадцати цифр, вычислительные машины имеют ограниченную разрядную сетку.

Ниже приведены факториалы нескольких чисел, больших 10, вычисленные на ЭВМ типа «Корвет».

Факториал числа 17 уже не поместился на экране, произошло переполнение разрядной сетки, и ЭВМ вместо точного результата 355 6874 2809 6000 выдала значение числа 17!

в нормализованной форме. ЭВМ предложила нам считать, что $17! \approx 3.55687428096 \text{ E}+14$.

```
3628800. ФАКТОРИАЛ ЧИСЛА 10
39916800 ФАКТОРИАЛ ЧИСЛА 11
479001600 ФАКТОРИАЛ ЧИСЛА 12
6227020800 ФАКТОРИАЛ ЧИСЛА 13
87178291200 ФАКТОРИАЛ ЧИСЛА 14
1307674368000 ФАКТОРИАЛ ЧИСЛА 15
20922789888000 ФАКТОРИАЛ ЧИСЛА 16
3.55687428096E+14 ФАКТОРИАЛ ЧИСЛА 17
6.402373705728E+15 ФАКТОРИАЛ ЧИСЛА 18
```

Означает ли это, что мы в принципе не можем на ЭВМ получить факториал большого числа, например, $100!$? Ответ простой — нет, не означает! Мы можем с помощью ЭВМ получить полную запись числа $100!$, не потеряв ни одной цифры. Это достигается путем использования программистской смекалки. Задача, которую мы собираемся решить, весьма примечательна для программирования. Речь пойдет о том, как человек использует ЭВМ, действует ли он по шаблону или же, вникнув в особенности ЭВМ, заставляет ее работать необычно, нестандартно. Разве хороший музыкант играет на необычной скрипке? Нет, он играет на обычной, но действует изобретательно, вдохновенно. Так же и в программировании — вдумчивый программист достигает удивительных высот в своем творчестве, применяя обычную ЭВМ. Может быть, решение задачи о вычислении числа $100!$ будет для вас примером вдохновенного труда в программировании.

Прежде чем рассказать о практически пригодной программе нахождения всех цифр числа $100!$, напомним о нескольких необходимых в дальнейшем средствах языка Бейсик.

В программе будет использована функция «целая часть от значения аргумента», в Бейсике эта функция обозначается так: $\text{INT}(x)$. Для любого действительного x значение функции $\text{INT}(x)$ равно целому значению x , не превышающему его.

Примеры

```
INT(2.5)=2,
INT(0.002)=0,
INT(9.95)=9,
INT(-6.3)=-7,
INT(-19.001)=-20.
```

Условимся все цифры получаемых факториалов чисел считать значениями переменных a_i . До начала вычисления $k!$ каждая переменная a_i равна нулю (рис. 5).

В процессе вычислений значения переменных a_i изменяются.

Напомним, что если известен факториал числа $(k-1)$, то факториал $k! = k \cdot (k-1)!$. Применим это правило к получению $4!$, если известно, что $3! = 6$. Действуем так: 6 умножаем на 4, полу-

чаем 24. Цифру 4 делаем значением переменной a_1 , а цифру 2 — значением переменной a_2 . Схематически все эти действия можно изобразить так, как показано на рисунке 6.

Продолжим работу: вычислим $5!$, для этого $4! = 24$ умножим на 5 и соответствующие цифры сделаем значениями переменных. (См. рис. 7).

Присмотримся внимательнее к процессу умножения чисел k и $(k-1)!$. Умножение одного многоразрядного числа на другое сводится к умножению и сложению цифр. При этом мы хорошо различаем цифру младшего разряда и цифры старших разрядов.

Приостановим обсуждение задачи в целом и научимся решать небольшую, но важную для дальнейшего задачу: научим вычислительную машину выполнять выделение цифры десятков и цифры единиц в двухразрядном десятичном числе. Необходимо составить программу, работая по которой, ЭВМ после предъявления ей двухразрядного числа переменной a_1 присвоит значение, равное числу единиц, а переменной a_2 — значение, равное числу десятков.

Пусть переменная A имеет значение, равное двухразрядному десятичному числу (например, 73). Разделим значение A на 10 и от результата возьмем целую часть — это и будет цифра десятков.

$$\text{LET } A(2) = \text{INT}(A/10) \quad (a_2 = [7,3] = 7).$$

Если теперь из значения A вычесть $a_2 \cdot 10$, то получим цифру единиц в исходном числе A .

$$\text{LET } A(1) = A - A(2) \cdot 10 \quad (a_1 = 73 - 7 \cdot 10 = 3).$$

Вернемся к задаче получения числа $5!$. Мы помним, что для этого придется число $4! = 24$ умножить на 5. Помним мы и о том, что число $4! = 24$ хранится необычно: цифра 4 есть значение переменной a_1 , а цифра 2 — значение переменной a_2 . Можно образно сказать, что цифры 2 и 4 хранятся в ячейках a_2 и a_1 соответственно.

Умножив содержимое ячейки a_1 на 5, мы получим двухразрядный результат: $4 \times 5 = 20$. Цифру младшего разряда (это в данном случае ноль) будем считать новым значением переменной a_1 , а цифру 2 удержим пока «в уме». После этого следует выполнить умножение цифры, хранящейся в a_2 , на 5 и к полученному произведению прибавить цифру 2, предварительно взяв ее из нашей памяти. Результат действий: $5! = 5 \cdot 4! = 5 \cdot 24 = 120$.

Схематически все выполненные действия можно изобразить так, как показано на рисунке 8.

После выполнения всех действий переменные a_1 , a_2 и a_3 получили новые значения $a_1 = 0$, $a_2 = 2$ и $a_3 = 1$.

Если число $(k-1)!$ содержит не две, а три и более цифр, то схема получения цифр числа $k!$ остается прежней. Изобразим ее на рисунке 9.

Естественно, что число k может быть не только одноразряд-

ным, а и двухразрядным. В этом случае коэффициенты m_i могут быть многоразрядными числами.

Используя высказанные идеи, можно предложить текст программы, который в таблице 1 (см. вклейку) приводится с разъяснениями.

Эта же программа, распечатанная самой ЭВМ, имеет вид:

```

210 REM ВЫЧИСЛЕНИЕ ФАКТОРИАЛА ЧИСЛА N
20 DIM A(100)
30 INPUT N
40 FOR I=2 TO 100
50 LET A(I)=0:NEXT I
60 LET A(1)=1:LET L=1
70 FOR K=1 TO N
80 LET R2=0:LET R1=0:LET I=1
90 IF R2=0 THEN IF I>L THEN 160
100 LET R=A(I)*K+R2
110 LET R2=INT(R/(10^6))
120 LET R1=R-R2*(10^6)
130 LET A(I)=R1
140 LET I=I+1
150 GOTO 90
160 LET L=I-1
170 NEXT K
180 REM ВЫВОД ФАКТОРИАЛА ЧИСЛА N
190 FOR J=1 TO I-1
200 PRINT A(I-J);
210 NEXT J
220 END

```

Задав ручную значение $N=100$, мы поручим ЭВМ вычислить число $100!$ и вправе ожидать, что она выведет результат, сохранив все цифры. Ответ впечатляет, ниже приводится число $100!$, вычисленное и выданное на печать ЭВМ.

```

93 326215 443944 152681 699238 856266 700490
715968 264381 621468 592 963 895217 599993 229915
608941 463976 156518 286253 697920 827223 758 251
185210 916864 0 0 0 0

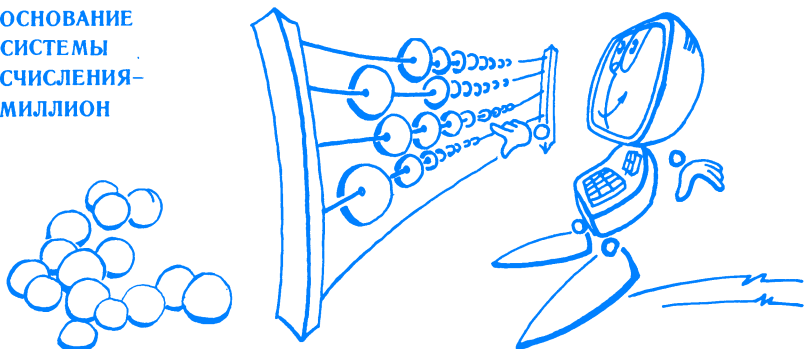
```

Сколько же цифр в числе $100!$? Ответ получить уже нетрудно — достаточно их посчитать. При этом нужно учесть, что 0 в конце числа — это 0 в разряде, поэтому при подсчете цифр каждый 0 надо считать шесть раз. Всего цифр 158. Это гигантское число.

Рассматривая результат, нетрудно заметить, что цифры числа при выводе оказались сгруппированными по 6 в группе. Кроме этого, необычно напечатаны последние четыре нуля. В чем же дело? Что заставило ЭВМ так выводить результат своей работы?

Ответ дается в следующем параграфе.

ОСНОВАНИЕ
СИСТЕМЫ
СЧИСЛЕНИЯ—
МИЛЛИОН



Читателю, конечно, известно, что при работе с ЭВМ используются системы с основанием $q=2$, $q=8$, $q=10$ и $q=16$. Каждая из систем имеет свои достоинства и наиболее полезное применение. О некоторых применениях будет рассказано.

Для ведения вычислений в каждой из этих систем счисления можно изготовить такие счеты, как показано на рисунке 10.

Теперь представим себе, что каждые счеты соединены со счетчиками так, как показано на рисунке 11. В окошках счетчиков демонстрируется количество одинарных косточек на каждой проволоке.

Спишем полученные числа с каждого счетчика и получим:

$$111101_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2 + 1 = 61_{10},$$

$$623_8 = 6 \cdot 8^2 + 2 \cdot 8 + 3 = 512 + 16 + 3 = 531_{10}.$$

Более необычно будет выглядеть счетчик, соединенный со счетами для системы счисления с основанием $q=16$ (рис. 12). Наибольшее число, которое может появиться в каком-нибудь из окошек счетчика, равно пятнадцати.

$$(9) \cdot 16^3 + (13) \cdot 16^2 + (5) \cdot 16 + 12 = 4096 + 6028 + 80 + 12 = 10216_{10}.$$

Представьте себе, что у нас есть счеты, на каждой проволоке которых не 16, а миллион косточек. Эти необычные счеты соединены со счетчиком, в каждом окошке которого могут появляться числа x , удовлетворяющие условию $0 \leq x \leq 999999$. Пусть результат, который показывает счетчик, изображен на рисунке 13.

Этот результат можно записать в виде многочлена:

$$(596213) \cdot (10^6)^3 + (123) \cdot (10^6)^2 + (57684) \cdot (10^6) + (691027).$$

Легко сообразить, что это запись числа в системе с основанием $q=10^6$. Числа, демонстрируемые счетчиком такого типа, можно схематически представить так:

$$(a_n) \cdot (10^6)^n + (a_{n-1}) \cdot (10^6)^{n-1} + (a_{n-2}) \cdot (10^6)^{n-2} + \dots + (a_2) \cdot 10^6 + (a_1).$$

В этой записи $a_1, a_2, a_3, \dots, a_{n-2}, a_{n-1}, a_n$ — «цифры» системы с основанием $q=10^6$. В роли цифр выступают числа $0 \leq x \leq 999999$. Еще раз подчеркнем, что цифра — это количество сдвинутых на одной проволоке косточек. *Цифры* в дальнейшем мы будем брать в скобки или писать над ними черту.

Пример числа, записанного в системе с основанием (миллион):

$$\overline{252640} \cdot (10^6)^3 + \overline{002510} \cdot (10^6)^2 + \overline{000011} \cdot 10^6 + \overline{908864} = \\ = 252640002510000011908864_{10^6}.$$

Это пример четырехразрядного числа в системе с основанием $q=10^6$. При переходе от одного разряда к следующему старшему «вес» единицы разряда возрастает не в десять раз, как в десятичной системе, а в миллион раз. Не правда ли, необычная система счисления? Какой смысл в ее введении? Какое отношение имеет эта система к вычислению числа $100!$?

Разъясним суть дела.

Пусть мы продолжаем кем-то ранее начатое вычисление факториалов нескольких, следующих одно за другим, натуральных чисел. До нас уже были вычислены факториалы первых 8 натуральных чисел, наша задача — продолжить вычисления, продолжить заполнение таблицы (рис. 14).

При занесении числа $10! = 3628800$ в «счетчик» пришлось использовать уже и второе окошко, в котором высвечиваются миллионы. Для этого потребовалось в числе 3628800 выделить единицу следующего разряда, а эта задача ранее уже рассматривалась. Напомним, что ранее для выделения «десятков» в числе A_{10} мы пользовались функцией $\text{INT}(A/10)$. Нетрудно сообразить, что теперь для выделения «миллионов» в числе A_{10} необходимо использовать эту же функцию, изменив в аргументе лишь знаменатель, мы должны теперь писать $\text{INT}(A/1000000)$ или $\text{INT}(A/(1E+6))$.

Число, демонстрируемое в каждом окошке счетчика, и есть цифра искомого числа $N!$. Теперь понятно, почему ЭВМ напечатала число $100!$, группируя цифры «по шесть» — дело в том, что перед вычислением в операторы программы вместо числа 10 были вписаны числа $1E+6$, и ЭВМ все вычисления осуществляла в «системе счисления с основанием миллион». Ясно, что это более быстрый алгоритм, ведь в каждом разряде без переноса единицы в следующие разряды ей удастся поработать во много раз больше, чем в десятичной системе.

Вот с каким необычным, но практически очень важным применением систем счисления может встретиться программист.

Любопытный читатель может, опираясь на рассмотренные идеи, попробовать написать программы сложения, умножения, деления и вычитания «длинных» чисел. Когда мы говорим «длинных» чисел, то имеем в виду, что число цифр в них велико, например, больше ста. Сами же числа могут быть очень маленькими

или ограниченными, но содержащими очень много цифр — речь идет о числах, в записи которых очень много цифр.

А сколько же все-таки цифр в числе 100!? Неужели придется вести счет цифрам вручную? Может быть, что-то можно изменить в программе или как-то ее дополнить с тем, чтобы ЭВМ сама сообщила, сколько цифр в записи числа 100!? Это сделать несложно. Напомним, что переменная L в программе есть счетчик числа использованных разрядов и поэтому достаточно только вывести на печать число $L = (L-1) \cdot 6$ и устно прибавить к нему число цифр старшего разряда (он может быть неполным, в старшем разряде результата может быть не шесть, а меньшее количество цифр. В числе 100! в старшем, 27-м, разряде всего две цифры — 9 и 3).

Узнать количество цифр в старшем разряде может и сама ЭВМ. Для этого достаточно дополнить программу такими операторами:

```
89 LET K=0
90 LET Q=INT(L/10^K)
91 IF Q>0 THEN LET K=K+1:GOTO 90
92 PRINT "K K-ВО ЦИФР В 100!";K
```

Для читателей, увлеченных программированием, рассказ о работе с числами, состоящими из большого количества цифр, может раскрыть некоторые «тайны» в искусстве программирования.

Рассказ о неожиданном использовании системы счисления с основанием $q=1000000$ увлекает нас в рассмотрение важной практической задачи, задачи создания дешифраторов. Под *дешифратором* далее мы будем понимать программу, работая по которой ЭВМ сумеет число, записанное в десятичной системе счисления, заменить равным ему числом в какой-то системе с другим основанием.

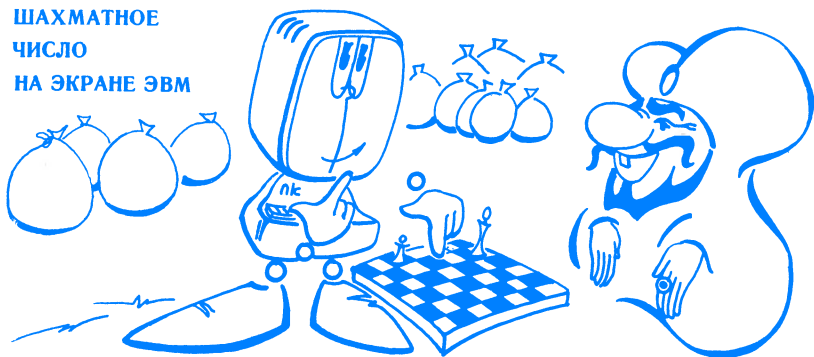
Использование системы счисления с необычным основанием $q=10^6$ позволило преодолеть технические трудности в применении ЭВМ.

Математики и программисты при решении задач давно пользуются возможностями перевода чисел из одной системы счисления в другую. В литературе можно встретить описание алгоритма игры «НИМ», в котором применяется система счисления с основанием $q=2$.

Особенно интересны задачи, в которых приходится переходить к системам с необычными основаниями. В практике применения ЭВМ, в программировании используются системы счисления не только с целым положительным, но и с целым отрицательным основанием, например $q=-2$, $q=-4$, и т. д., системы счисления с комплексным основанием $q=2i$, $q=1-i$ и другие.

Рекомендуем читателю расширить свои представления о десятичных системах счисления и их возможностях при разработке алгоритмов.

ШАХМАТНОЕ
ЧИСЛО
НА ЭКРАНЕ ЭВМ



Многим известна легенда о том, что изобретатель шахмат запросил за свое удивительное изобретение необычную плату: он попросил дать ему пшеничных зерен. Сколько же? Изобретатель попросил выдать ему зерна из следующего расчета: на первую клетку шахматной доски положить одно зерно, на вторую — два, на третью — четыре и так далее: на каждую следующую в два раза больше, чем на предыдущую.

Для решения этой задачи обычно применяют формулу *суммирования членов геометрической прогрессии*. В данном случае первый член прогрессии равен единице $b_1=1$, знаменатель прогрессии $q=2$, следовательно, искомая сумма зерен, «размещенных» на всех 64 клетках шахматной доски, равна:

$$S = \frac{b_1 \cdot (q^n - 1)}{q - 1}.$$

Обычно приводится результат в такой форме, иногда выполняются вычисления и дается полный результат. Попробуем его получить и мы: поручим ЭВМ вести суммирование шаг за шагом вычисляемых степеней числа 2.

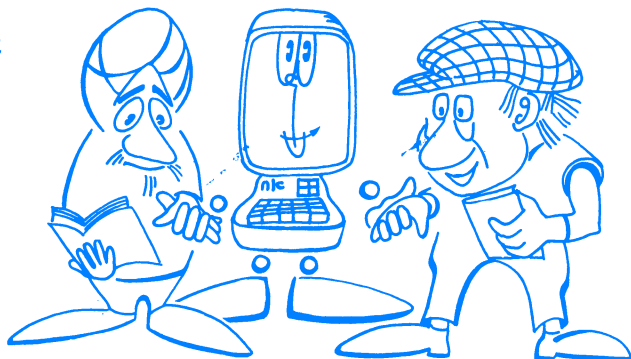
Вычислительная машина берется за работу, но через несколько секунд нас ожидает разочарование. Сумма зерен, расположенных на 46 клетках доски, $S_{46}=70368744177663$, и это самое большое число, которое может продемонстрировать ЭВМ, например, ЭВМ типа «Корвет». Если к этому числу прибавить все зерна, расположенные на 47-й клетке, то полученная сумма будет выдана уже в нормализованной форме: $S_{47}=1.4073748835533E+14$.

Часть цифр точного результата утрачены. Как же получить число $2^{64}-1$, сохранив все его цифры?

Нетрудно сообразить, что на помощь может прийти программа **«Вычисление факториала числа K »**. Степень числа есть произведение одинаковых множителей, в данном случае перемножаются двойки. Учитывая это, вносим в программу изменение: вместо переменной K записываем константу 2, а перед началом работы N полагаем равным 64.

В чем суть такого изменения? В ранее использованной программе число K сначала было равным 1, затем 2, затем 3 и так далее — перед каждым его использованием в качестве множителя мы увеличивали его на единицу. Внеся поправку, мы поручили машине вычислять не произведение вида: $1 \cdot 2 \cdot 3 \cdot 4 \cdot \dots \cdot n = n!$, а произведение вида: $1 \cdot 2 \cdot 2 \cdot \dots \cdot 2 = 2^n$, то есть вычислять степени числа 2. Вводим измененную программу и запускаем ЭВМ. Через несколько секунд ответ готов: $S_{64} = 18446744073709551615 = 2^{64} - 1$.

ЗАДАЧА О ДЕШИФРАТОРЕ



В производственной практике и в быту люди чаще всего пользуются десятичной системой счисления. Есть страны, где используются особые, рожденные в глубокой древности числовые системы. Такие национальные системы иногда обладают удивительными качествами.

Остановимся, для примера, на системе счисления, принятой в Японии. Выражая результаты вычислений, записывая числа, японцы пользуются символами:

единицы: 一 (1 — ичи); 二 (2 — ни); 三 (3 — сан); 四 (4 — си);

五 (5 — го); 六 (6 — року); 七 (7 — сити); 八 (8 — хати);

九 (9 — ку)

десяток: 十 (дзю)

сотня: 百 (хяку)

тысяча: 千 (сэн)

десять тысяч: 万 (ман).

Примеры

二百八十七 ($2 \cdot 10^2 + 8 \cdot 10 + 7 = 287_{10}$ — ни хяку хати дзю сити),

一千二百三十七 ($1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10 + 7 = 1237_{10}$ — ити сэн ни хяку сан дзю сити),

十二万二千六百 ($12 \cdot 10^4 + 2 \cdot 10^3 + 6 \cdot 10^2 = 122600_{10}$ —
(дзю ни ман ни сэн року хяку)).

Обращаем внимание на то, что все числа, записанные иероглифами, являются числами, записанными в многочленной форме. В качестве цифры, выписываемой перед какой-либо степенью десятки, можно использовать как отдельные символы (**三十** 3000), так и числа, составленные из предшествующих иероглифов. Например, число 673_{10} , записанное по-японски, имеет вид: **六百七十三**, число из пяти иероглифов следует считать «цифрой», если оно записано перед иероглифом **五** (ман), обозначающим 10^4 : **六百七十三五** $673 \cdot 10^4 = 6730000_{10}$.

Если возникает потребность записать число, большее чем десять тысяч в 10^4 раз, то используется символ **億** (оку).

Из сказанного ясно, что при записи чисел, меньших чем 10^4 , в Японии используется десятичная система, а для записи чисел, больших чем 10^4 , система, в которой основание равно 10^4 . Это пример *смешанной позиционной системы счисления*, одно из оснований системы очень большое ($q=10^4$). Как видно, система с большим основанием не такая уж редкость. Принятая в нашей стране система также является смешанной. Числа, большие 1000, мы считаем в системе с основанием $q=10^3$.

Конечно, особенности японской системы счисления сильно затрудняют перевод чисел с японского языка на какой-нибудь из европейских. Переводчик не только переводит слова (числительные одного языка заменяет числительными другого), но и выполняет вычисление значения многочлена, который записывает или произносит японец. Если переводчик переводит число, записанное в десятичной системе, то он сначала (если переводимое число большое) должен применить систему счисления с основанием $q=10^4$, а затем десятичную систему счисления.

Задача замены числа, данного и записанного в одной системе счисления, равным ему числом в другой системе счисления, встречается не только в общении с иностранцами. Эта проблема возникает и в контакте между человеком и ЭВМ. Дело в том, что арифметика многих вычислительных машин основана на использовании двоичной системы счисления, а люди в общении друг с другом используют чаще всего десятичную систему счисления. Появляется необходимость перевода чисел из десятичной системы в двоичную — так поступают при вводе числовых данных в ЭВМ. По окончании работы полученный результат (это чаще всего двоичное число) ЭВМ должна преобразовать в равное число десятичной системы и вывести его на экран или на печать. При написании программ широко используются системы счисления с основанием $q=8$ и $q=16$. И здесь необходимо заменять числа одной системы равными им числами другой. Поскольку замена

чисел есть преобразование информации, то и ее следует поручить решать ЭВМ.

Конечно, разрабатывая программы, следует стремиться к возможно более общему решению проблемы, например, составить программу, которая обеспечивала бы замену числа, записанного в системе с основанием q , равным числом системы счисления с основанием p , где p и q — допустимые в качестве оснований системы числа. Конечно, было бы интересно научиться заменять числа, записанные по-японски, на числа десятичной системы и наоборот.

Решение программистской задачи осуществим в три этапа:

1) составим программу замены любого данного натурального десятичного числа равным ему числом позиционной системы с основанием q ($q > 10$ и $q < 10$);

2) составим программу замены любого натурального числа, записанного в системе с основанием q , равным ему десятичным числом;

3) составим программу замены любого данного числа (натурального), записанного в системе с основанием q , равным ему числом, записанным в системе счисления с основанием p .

Решение первой задачи начнем с того, что вспомним, как человек решает такую задачу вручную. В школе изучается алгоритм замены чисел десятичной системы равными им числами двоичной и восьмеричной систем с помощью деления «углом» (рис. 15).

Алгоритм получения искомого числа состоит в том, что мы последовательно осуществляем деление данного десятичного числа на новое основание системы q и выделяем остатки. Выбранные остатки, взятые в обратном порядке, и есть цифры искомого числа в системе с основанием q .

Вновь мы обращаемся к уже рассмотренной задаче — задаче выделения целой части числа и остатка. В этот раз исходное число есть десятичное число, а число, на которое ведется деление, есть основание новой системы счисления. Цифры искомого числа будем размещать в массиве $O(N)$, отводя для каждой цифры один элемент массива. Такой прием уже использовался выше при вычислении $100!$.

Возможный вариант программы с краткими комментариями приводится ниже.

```
10 REM ДЕШИФРАТОР ИЗ 10 В Q
20 INPUT "ВВЕДИТЕ ИСХОДНОЕ ЧИСЛО A1 И ОСНОВАНИЕ НОВОЙ СИСТЕМЫ Q"; A1, Q
30 DIM A(100), Q(100)
40 LET I=0: LET K=0: LET A(I)=A1
50 LET A(I+1)=INT(A(I)/Q)
60 LET O(I)=A(I)-A(I+1)*Q
70 IF A(I+1)<>0 THEN LET I=I+1: GOTO 50
80 FOR R=0 TO I
90 PRINT O(I-R);
100 NEXT
110 END
```

Читателя не должно удивлять, что в некоторых случаях в роли цифры выступают «двухразрядные» числа. Это случается тогда, когда основание новой системы счисления $q > 10$.

Приступая к составлению программы замены чисел, данных в системе с основанием q , равными им числами десятичной системы, вспомним, как эта задача решается вручную. Начнем с примеров:

$$56704_8 = 5 \cdot 8^4 + 6 \cdot 8^3 + 7 \cdot 8^2 + 0 \cdot 8 + 4 = 5 \cdot 4096 + 6 \cdot 512 + 7 \cdot 64 + 4 = 20480 + 3072 + 448 + 4 = 24004_{10};$$

$$1010111_2 = 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2 + 1 = 87_{10};$$

$$A9BF_{16} = 10 \cdot 16^3 + 9 \cdot 16^2 + 11 \cdot 16 + 15 = 43455_{10}, \text{ где } A, B, F — \text{цифры системы с основанием } 16.$$

Все три примера на одну тему — вычисление значений многочлена вида $a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_1 x + a_0$ при некотором заданном значении x . Из сказанного заключаем, что если $a_n a_{n-1} a_{n-2} \dots a_1 a_0 q$ — есть цифровая запись некоторого числа в системе с основанием q , то получение равного ему десятичного числа сводится к вычислению значения многочлена:

$$a_n \cdot q^n + a_{n-1} \cdot q^{n-1} + a_{n-2} \cdot q^{n-2} + \dots + a_1 q + a_0.$$

Задача *вычисления значений многочлена* всегда (это традиция) рассматривается при обучении школьников программированию.

Горнер Вильямс Джордж (1786—1837) — английский математик, работал в области алгебры. Опубликовал способ приближенного вычисления вещественных корней многочлена.

Напомним алгоритм и программу. Перед вычислением исходный многочлен необходимо преобразовать по схеме Горнера, то есть записать его в виде

$$(\dots((a_n x + a_{n-1})x + a_{n-2})x + \dots + a_1)x + a_0. \quad (*)$$

Программирование вычисления значений многочлена свелось к задаче программирования последовательности действий, заданных формулой (*). Задать такого типа многочлен для ЭВМ — это значит ввести в ее память все коэффициенты $a_i (i=0, 1, 2, \dots, n)$. В память машины следует поместить и число n — степень многочлена и значение q (основание новой системы счисления).

Примем вариант, когда все коэффициенты будут храниться в DATA в той последовательности, в которой они будут использоваться в процессе шифрования. Программа может быть такой:

```

120 REM ДЕШИФРАТОР ИЗ Q В 10
130 DATA 4,3,2,1,2,0,1
140 READ N,Q,A
150 READ A1
160 LET A=A*Q+A1
170 LET N=N-1
180 IF N>0 THEN 150
190 PRINT A
200 END

```

Читатель вправе спросить, будут ли обе программы работать, если количество цифр в исходных и получаемых числах велико. Ответ дать нетрудно — нет, не будут. В обеих программах действуют ограничения, накладываемые разрядной сеткой, используемой ЭВМ. В программе «**дешифратор из 10 в Q**» исходное число не может содержать более 15 цифр, а в программе «**дешифратор из Q в 10**» результат, т. е. десятичное число, не может содержать более 15 цифр.

О том, что такие ограничения в принципе преодолимы, читатель уже знает. С такой проблемой мы сталкивались выше. Идеи, на основе которых можно разработать программу для шифрования чисел с большим количеством цифр, известны, дело за составлением программы — до успеха один шаг.

Но в программировании часто бывает так: имеется ясная идея алгоритма, кажется, что программу написать нетрудно, но вас может ожидать сюрприз — большая работа, работа, которая требует терпения и дополнительной изобретательности.

Если Вы вдумчиво (это значит Вам ясен каждый оператор и все взаимосвязи между операторами программы) изучили программу «**Вычисление факториала числа N**», то сумеете самостоятельно разработать программу, которую целесообразно называть «**Дешифратор для чисел с большим количеством цифр**».

Мы же ограничимся тем, что два программных дешифратора объединим в один. Это можно сделать, используя небольшой диалог. Обе разработанные программы заносятся в память ЭВМ до начала их применения.

```

1  REM ДЕШИФРАТОР ДЛЯ ЧИСЕЛ С БОЛЬШИМ ЧИСЛОМ ЦИФР
2  PRINT "ИСХОДНОЕ ЧИСЛО ДЕСЯТИЧНОЕ - ДА ИЛИ НЕТ?"
3  INPUT "A$="; A$
4  IF A$="ДА" THEN 10
5  GOTO 120
10 REM ДЕШИФРАТОР ИЗ 10 В Q
20 INPUT "ВВЕДИТЕ ИСХОДНОЕ ЧИСЛО A1 И ОСНОВАНИЕ НОВОЙ СИСТЕМЫ"; A1, Q
30 DIM A(100), O(100)
40 LET I=0: LET K=0: LET A(I)=A1
50 LET A(I+1)=INT(A(I)/Q)
60 LET O(I)=A(I)-A(I+1)*Q
70 IF A(I+1)<>0 THEN LET I=I+1: GOTO 50
80 FOR R=0 TO I
90 PRINT O(I-R);
100 NEXT R
110 END
120 REM ДЕШИФРАТОР ИЗ Q В 10
130 DATA 4,3,2,1,2,0,1
140 READ N,Q,A
150 READ A1
160 LET A=A*Q+A1
170 LET N=N-1
180 IF N>0 THEN GOTO 150
190 PRINT A

```

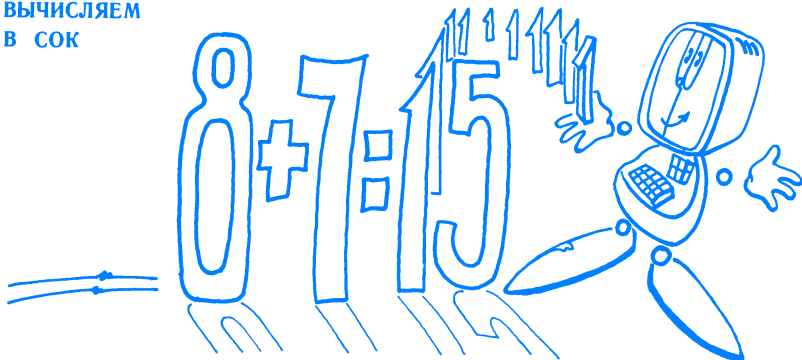
Для читателей, заинтересовавшихся проблемой создания дешифрирующих программ, укажем, что в практике используются

позиционные системы счисления с различными основаниями, в том числе и с основаниями, не являющимися натуральными числами. Существуют негепозиционные системы, основание в которых есть целое отрицательное число, например $q=-2$, $q=-4$, и другие. Существуют системы с основанием, выраженным комплексным числом, например, $q=2i$.

Для каждой из таких систем (и многих других, о которых мы не упоминали) необходимо иметь дешифратор, который любое число, записанное в недесятичной системе, заменит равным ему числом десятичной системы и наоборот.

Мы специально остановимся только на одном из дешифраторов, а именно, на программе, работая по которой ЭВМ данное десятичное число запишет в системе остаточных классов и на программе решения обратной задачи: замена числа, данного в системе остаточных классов, равным ему десятичным числом. Рассказ об этом — в следующем параграфе.

ВЫЧИСЛЯЕМ
В СОК



Общим достоинством позиционных систем счисления является компактность записи чисел (цифра за цифрой), простота алгоритмов выполнения основных арифметических действий над числами, возможность сведения операций над числами к действиям над цифрами. Так, сложение и умножение в позиционных системах счисления основываются на таблицах сложения и умножения цифр.

Может быть, самым неудобным и трудным при сложении цифр (или при умножении их) является необходимость запоминать единицы переноса: «восемь плюс семь — пятнадцать; пять пишем, один запоминаем», или «трижды восемь — двадцать четыре; четыре пишем, два запоминаем». Необходимость работать с цифрами переноса можно считать недостатком позиционных систем, и можно попробовать поставить задачу устранить этот «недостаток». Иначе говоря, можно заняться решением задачи по разработке новой специальной системы счисления, сложение

чисел в которой должно осуществляться поразрядно, но при этом без операции переноса единицы в старший разряд. Объявим сразу, что такая система найдена и названа *системой остаточных классов* — СОК. Изучению особенностей этой системы следует предпослать некоторые сведения о сравнениях.

Рассмотрим множество натуральных чисел. Зададим какое-нибудь небольшое число, например, $m=5$, и назовем его *модулем*. При делении любого натурального числа на число $m=5$ могут получиться остатки: 0, 1, 2, 3 и 4. Все множество натуральных чисел можно разбить на пять классов, включая в каждый класс числа, которые при делении на 5 дают один и тот же остаток. Числа, которые при делении на 5 дают одинаковые остатки, будем называть *сравнимыми по модулю* пять. Вот как выглядят упомянутые пять классов:

$$\div 1, 6, 11, 16, 21, \dots$$

$$\div 2, 7, 12, 17, 22, \dots$$

$$\div 3, 8, 13, 18, 23, \dots$$

$$\div 4, 9, 14, 19, 24, \dots$$

$$\div 0, 5, 10, 15, 20, \dots$$

Числа в каждом (горизонтальном) классе образуют арифметическую прогрессию с формулами «общего члена»:

$$a_n=5n-4,$$

$$a_n=5n-3,$$

$$a_n=5n-2,$$

$$a_n=5n-1,$$

$$a_n=5(n-1), \text{ где } n \in \mathbb{N}.$$

Ясно, что каждое натуральное число войдет только в один класс, будет членом только одной из этих прогрессий. Назовем такие прогрессии и множества чисел, в них входящих, *классами по модулю* пять.

Аналогично рассуждая, можно ввести понятие о классах по любому другому натуральному модулю p . Числа, входящие в классы, называют *вычетами*, например, по модулю 5. Если из каждого класса взять по одному вычету, то их совокупность называется: *полная система вычетов* по данному модулю. Обычно в полную систему вычетов берут из каждого класса наименьшие числа. Полная система вычетов по модулю p всегда содержит числа: 0, 1, 2, 3, ..., $(p-1)$.

Введем над числами, входящими в полную систему вычетов, операцию *сложения*: сложить два вычета из полной их системы — это значит *сложить их по правилам десятичной арифметики, но сумму заменить наименьшим вычетом этого же класса*.

Пример

Рассмотрим класс вычетов по модулю 5. Пусть к трем прибавляется четыре, в ответе семь. Найдем остаток от деления семи на пять — остаток равен двум, поэтому можем записать

$3 \oplus 4 = 2$. Знак «плюс» взят в кружок, чтобы подчеркнуть, что суммирование проводится по необычным правилам. Еще примеры: $6 \oplus 4 = 0$ (так как 10 при делении на 5 дает в остатке 0) или $3 \oplus 3 = 1$ (6 при делении на 5 дает в остатке 1).

Все сказанное дает основание составить таблицу сложения для вычетов по модулю 5 (табл. 2).

Таблица 2

\oplus	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Когда хотят сказать, что два числа принадлежат одному классу, то пишут:

$$3 + 4 \equiv 2 \pmod{5},$$

$$4 + 4 \equiv 3 \pmod{5}.$$

Таблица 3

\otimes	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Аналогично составляется таблица умножения вычетов по модулю 5 (табл. 3). При умножении вместо произведения следует писать остаток от деления полученного произведения на 5:

$$3 \otimes 4 = 2,$$

$$4 \otimes 4 = 1.$$

Обычно записывают так:

$$3 \times 4 \equiv 2 \pmod{5} \text{ и } 4 \times 4 \equiv 1 \pmod{5}.$$

Сложение и умножение обладают *переместительным* и *сочетательным* свойствами, а операция умножения — и *распределительным* свойством относительно сложения

$$\begin{aligned} 3 + 4 &\equiv 2 \pmod{5} & \text{и} & & 4 + 3 &\equiv 2 \pmod{5}, \\ 2 \times 3 &\equiv 1 \pmod{5} & \text{и} & & 3 \times 2 &\equiv 1 \pmod{5}, \\ 2(3 + 4) &\equiv 4 \pmod{5} & \text{и} & & (2 \times 3 + 2 \times 4) &\equiv 4 \pmod{5}. \end{aligned}$$

Все сказанное подготовило введение понятия о системе счисления остаточных классов (СОК). Задать СОК — это значит указать набор модулей, то есть несколько взаимно простых натуральных чисел. Набор может быть, например, таким: $p_1=2$, $p_2=3$, $p_3=5$ и $p_4=7$.

Располагая заданным набором модулей, можно приступить к записи чисел в конкретной системе, определяемой этим набором модулей. Делается это так: данное десятичное число A_{10} делим на первый модуль, на p_1 , и остаток считаем первой цифрой искомого числа. Затем это же число A_{10} делим на p_2 и остаток считаем второй цифрой образуемого числа, третья и четвертая цифры искомого числа есть остатки от деления A_{10} на $p_3=5$ и $p_4=7$.

Пример

Пусть $A_{10}=183_{10}$. Делим число 183 сначала на 2, получаем в остатке 1 ($a_1=1$), затем 183 делим на 3 и получаем 0 ($a_2=0$), затем 183 делим на 5 и получаем $a_3=3$ и на 7 и имеем $a_4=1$. Следовательно, $183_{10}=1031_c$ (буква «с» внизу означает, что число записано в системе остаточных классов).

Это же число $A_{10}=183_{10}$ при другом наборе модулей будет записано другими цифрами. Заметим, что цифр в каждом числе столько же, сколько и модулей, и поэтому нуль, стоящий слева, отбрасывать нельзя. Подчеркнем, что если набор модулей задан, то этим самым определен и диапазон чисел, которые могут быть записаны в этом наборе. В приведенном примере $p_1=2$, $p_2=3$, $p_3=5$ и $p_4=7$. Следовательно, наибольшим числом, которое может быть записано в системе с таким набором модулей, является число 210_{10} , равное произведению всех модулей.

Формула числа, записанного в СОК, привычна, число записано «цифра за цифрой»: $a_1a_2a_3...a_k$.

Рассмотрим сложение чисел в системе счисления остаточных классов. Примеры будем брать для системы вышеуказанных модулей. Напомним, что сложение цифр осуществляется по каждому модулю отдельно и по вышерассмотренному правилу.

Пример

Пусть требуется вычислить сумму чисел $A_1=89_{10}$ и $A_2=76_{10}$, записанных в системе остаточных классов: $89_{10}=1245_c$ и $76_{10}=0116_c$.

Суммируем:

$$\begin{array}{r} + 1245 \\ 0116 \\ \hline 1004 \end{array}$$

Сложение цифр «старших» разрядов осуществляется по модулю два (в сумму берется остаток от деления десятичной суммы цифр на 2), сложение цифр в следующем столбце ведется по модулю 3, затем цифры 4 и 1 складываются по модулю 5 ($4+1 \equiv 0 \pmod{5}$) и, наконец, $5+6 \equiv 4 \pmod{7}$.

Проверим результат так. Найдем десятичную сумму: $89_{10} + 76_{10} = 165_{10}$, этот результат запишем в рассматриваемой системе остаточных классов и получим: $165_{10} = 1004_c$. Это значит, что суммирование чисел 1245_c и 0116_c проведено верно.

Подчеркнем, что суммирование чисел, записанных в СОК, свелось к поразрядному сложению цифр; никакого переноса единиц в следующий разряд не использовалось. Это означает, что система, в которой числа суммируют поразрядно, без использования операции переноса единиц из младших разрядов в старшие, найдена — это система остаточных классов.

Большое впечатление на тех, кто впервые знакомится с системой остаточных классов, производит то обстоятельство, что умножение чисел осуществляется поразрядно, без переноса единиц в старшие разряды.

Пример

Найдем произведение $13_{10} \times 11_{10}$. Запишем оба множителя в системе СОК:

$$\begin{array}{r} \times 13_{10} = \times 1136_c \\ \times 11_{10} = \times 1214_c \\ \hline + 13 \\ \hline 143_{10} = 1233_c \end{array}$$

Разъясним только одно умножение:
 $6 \times 4 \equiv 3 \pmod{7}$.

Подведем итоги: любое положительное целое число десятичной системы счисления можно записать в виде числа в какой-нибудь системе остаточных классов. С такими числами необычно выполняются и сложение, и умножение. Никаких проблем типа «пять пишем, четыре в уме».

Остался один вопрос: как осуществляется замена чисел, записанных в конкретной СОК, равными им десятичными числами?

Мы подошли к еще одной задаче на тему «Создание программных дешифраторов», перед нами возникла задача — создать программу, работая по которой ЭВМ сумеет десятичное число заменить числом, записанным в СОК, и наоборот.

Первая часть задачи оказывается простой — в самом деле, что значит число A_{10} записать в системе счисления остаточных классов с набором модулей $p_1, p_2, p_3, \dots, p_k$? Это значит, число A_{10} поочередно разделить на модули и каждый раз в качестве цифры брать полученный при делении остаток.

Если исходное число A и модули (p_1, p_2, \dots, p_k) поместить в хранилище DATA, то нижеприводимая программа «заставит» ЭВМ выступить в роли дешифратора «Из 10 в СОК».

```
10 REM ДЕШИФРАТОР ИЗ 10 В СОК
30 DIM A(5)
40 INPUT "ВВОД ИСХОДНОГО ЧИСЛА В"; В
50 DATA 2, 3, 5, 7, 11
60 FOR I=1 TO 5
```



```

70 READ P
80 LET A(I) = B - INT(B/P) * P
90 NEXT I
100 FOR I=1 TO 5
110 PRINT A(I);
120 NEXT I
130 END

```

Как же составить программу «обратного» дешифратора? Эта задача значительно труднее и разработка программы требует обсуждения некоторых теоретических вопросов. .

Сформулируем задачу так, как это делали китайские математики более 2000 лет назад, открывшие алгоритм ее решения.

Задача. Восстановить число по данным остаткам от деления его на известные числа.

Искомый алгоритм задается формулой:

$$A_{10} \equiv (a_1 B_1 + a_2 B_2 + a_3 B_3 + \dots + a_k B_k) \pmod{N}.$$

Здесь $N = p_1 \cdot p_2 \cdot p_3 \cdot \dots \cdot p_k$; a_i — цифры числа, записанного в системе СОК с набором модулей $p_1, p_2, p_3, \dots, p_k$; коэффициенты $B_1, B_2, B_3, \dots, B_k$ находят из условия: $B_i = \frac{m_i N}{p_i} \equiv 1 \pmod{p_i}$, а m_i находят из условия $\frac{m_i N}{p_i} \equiv 1 \pmod{p_i}$.

Рассмотрим задачу вычисления коэффициента m_i на примере системы с модулями $p_1=2, p_2=3, p_3=5, p_4=7$. Найти m_1 — это значит число $N=210$ разделить на $p_1=2$ и найти m_1 из условия: $m_1 \cdot 105 \equiv 1 \pmod{2}$, то есть найти такое m_1 , чтобы $(m_1 \cdot 105 - 1)$ делилось нацело на 2. Найти m_2 — это значит число $N=210$ разделить на $p_2=3$ и найти m_2 из условия $m_2 \cdot 70 \equiv 1 \pmod{3}$, то есть найти такое m_2 , чтобы $(m_2 \cdot 70 - 1)$ делилось без остатка на 3. И далее: m_3 ищем из условия $(m_3 \cdot 42 - 1) \div 5$, а m_4 — из условия $(m_4 \cdot 30 - 1) \div 7$.

Можно ли такую работу поручить ЭВМ? Да, можно! Вариант программы приводится ниже.

```

10 REM ДЕШИФРАТОР ИЗ СОК В 10
20 REM РАБОТАЕТ С ЧИСЛАМИ X <= 2310
30 REM ФОРМИРОВАНИЕ МАССИВА P(5) МОДУЛЕЙ
40 DIM P(5), R(5)
50 DATA 2, 3, 5, 7, 11
60 N=2*3*5*7*11
70 FOR I=1 TO 5: READ P(I): NEXT I
80 REM ФОРМИРОВАНИЕ МАССИВА КОЭФ-ТОВ В
90 DIM B(5): LET M=1
100 FOR K=1 TO 5
110 LET V=N/P(K)*M
120 LET C=(V-1)/P(K): LET F=C-INT(C)
130 IF F=0 THEN 150
140 LET M=M+1: GOTO 110
150 LET B(K)=V
160 LET M=1
170 NEXT K
180 REM "ВВОД ИСХОДНОГО ЧИСЛА"
190 FOR I=1 TO 5

```

```

200 INPUT R(I): NEXT I: LET S=0
210 REM ВЫЧИСЛЕНИЕ ИСКОМОГО ЧИСЛА
220 FOR I=1 TO 5
230 LET S=S+R(I)*B(I): NEXT I
240 PRINT "РЕЗУЛЬТАТ -"; INT((S/N - INT(S/N))*N+.5)
250 END

```

Получен программный дешифратор «Из СОК в 10».

Если теперь две эти программы объединить, то получим дешифратор, который позволит быстро готовить числа десятичной системы для работы с ними по правилам систем остаточных классов и по окончании вычислений этот же дешифратор заменит результат счета в СОК равным ему десятичным числом.

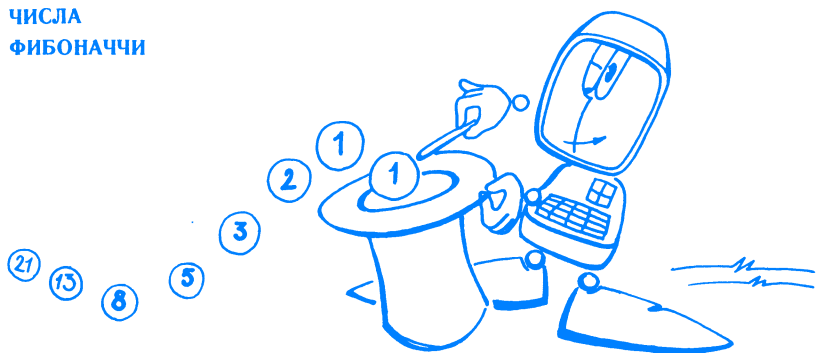
Дело за немногим — научить ЭВМ работать по-новому, вести вычисления «без переноса единиц в старшие разряды». Иначе говоря, необходимо составить программу сложения и умножения цифр по правилам арифметики СОК. Такие программы предлагаем составить читателям самостоятельно. Свою задачу — задачу создания программы-дешифратора, работающего в столь необычной системе счисления, считаем выполненной. Как всегда подчеркнем, что предложенные варианты программ не являются единственными и не претендуют на звание лучших.

Естественно задаться вопросами, которые мы обычно ставим, рассматривая числа десятичной системы счисления.

Пусть модули конкретной системы счисления остаточных классов есть числа: $p_1=2$, $p_2=3$, $p_3=5$, $p_4=7$. Как следует рассуждать, чтобы выяснить, является ли данное число 1000_c , записанное в этой системе: четным, кратным 3, кратным 5 или кратным 7? Делится ли число 1030_c на 21? Или более общий вопрос: какое из чисел: 1030_c и 1000_c больше? Еще один вопрос: делится ли нацело число 1000_c на число 1001_c ? Как следует сформулировать признак делимости числа на 4 в рассматриваемой системе остаточных классов?

В заключение немного пофантазируем: представьте себе, что у Вас имеется сконструированный выше дешифратор и что Вы разработали систему суммирования и умножения чисел, записанных в СОК. Это значит, что Вы можете эффективно суммировать и умножать, ведь работа во всех разрядах может осуществляться одновременно — вычисления можно вести параллельно. Эта подкупающая идея привлекла внимание инженеров-конструкторов вычислительных машин, и ЭВМ, в основе работы арифметических узлов которых лежит СОК, были созданы.

Все, о чем мы рассказали, можно назвать созданием модели такой необычной ЭВМ.



Числа, о которых пойдет речь, хорошо известны математикам. Есть немало серьезных и занимательных задач, в которых главными действующими лицами и исполнителями являются эти числа, названные в честь замечательных ученых.

Леонардо Пизанский (Фибоначчи) (1180—1240) — итальянский математик. В 1202 г. издал «Книгу об абак». По этой книге европейские математики изучили индийскую позиционную систему счисления.

Рассказ о каждом из этих числовых множеств может составить книгу. Остановимся на этих числах с тем, чтобы взглянуть на них глазами программиста. Наша задача не в том, чтобы изучить свойства этих чисел, а научить ЭВМ генерировать их. Мы не будем рассматривать многочисленные применения названных чисел, а обсудим особенности создания программ, действуя по которым ЭВМ сможет вычислять эти числа.

Выше мы уже имели дело с этими числами. Напомним, что *числа Фибоначчи* есть члены последовательности 1, 1, 2, 3, 5, 8, 13, 21, Каждый член этой последовательности, начиная с третьего, равен сумме двух ему предшествующих, а первые два члена одинаковы, они равны единице:

$$\begin{aligned}a_3 &= a_2 + a_1 = 1 + 1 = 2, \\a_4 &= a_3 + a_2 = 2 + 1 = 3, \\a_5 &= a_4 + a_3 = 3 + 2 = 5, \\a_6 &= a_5 + a_4 = 5 + 3 = 8.\end{aligned}$$

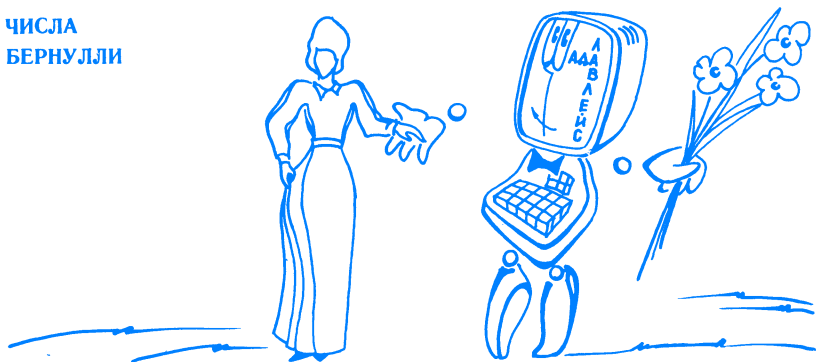
Составим программу получения N первых чисел Фибоначчи и вывода их на печать.

Текст программы может быть таким:

```
5 REM ГЕНЕРАТОР ЧИСЕЛ ФИБОНАЧЧИ
10 DIM F(100)
15 LET F(1)=1:LET F(2)=1
20 INPUT N
25 FOR I=3 TO N
30 LET F(I)=F(I-1)+F(I-2)
35 NEXT I
40 FOR K=1 TO N
45 PRINT F(K);
50 NEXT K
55 END
```

Для вычисления очередного, n -го члена последовательности Фибоначчи использовалась рекуррентная формула $a_n = a_{n-1} + a_{n-2}$. Напомним, что термин «рекуррентный» происходит от латинского слова „гесигго“, что означает: «бегу назад», или «возвращаюсь». Рекуррентные формулы широко используются в математике. Часто эти формулы совсем не просты и разработка программы для ЭВМ, которая будет ими пользоваться, требует немалой смекалки.

ЧИСЛА БЕРНУЛЛИ



Для получения чисел Бернулли можно использовать следующую рекуррентную формулу:

$$B_0 = 1; C_k^0 B_0 + C_k^1 B_1 + C_k^2 B_2 + C_k^3 B_3 + \dots + C_k^{k-1} B_{k-1} = 0.$$

Бернулли Иоганн (1667—1748) — швейцарский математик из династии швейцарских ученых. Достиг больших результатов в разработке дифференциального и интегрального исчисления, вариационном исчислении, геометрии и механике.

Научимся вычислять числа Бернулли вручную. Найдем B_1 , для этого нужно решить уравнение: $C_2^0 B_0 + C_2^1 B_1 = 0$, или $1 + 2 \times B_1 = 0$, или $B_1 = -1/2$. Зная B_1 , найдем B_2 . Решаем уравнение: $1 + C_3^1 B_1 + C_3^2 B_2 = 0$, или $1 + 3(-1/2) + 3B_2 = 0$ и отсюда получаем $B_2 = 1/6$.

Вычислим число Бернулли B_3 , для этого решим уравнение: $C_4^0 B_0 + C_4^1 B_1 + C_4^2 B_2 + C_4^3 B_3 = 0$, или $1 + 4(-1/2) + 6(1/6) + 4B_3 = 0$, или $4B_3 = 0$ и отсюда $B_3 = 0$. Таким образом, действуя шаг за шагом, вычислив B_n , можно переходить к вычислению B_{n+1} .

Обратимся к примерам и попробуем получить возможно более общую формулу, которую можно будет положить в основу программы:

$$B_4 = -\frac{1 + C_5^1 B_1 + C_5^2 B_2 + C_5^3 B_3}{C_5^4}$$

и

$$B_5 = -\frac{1 + C_6^1 B_1 + C_6^2 B_2 + C_6^3 B_3 + C_6^4 B_4}{C_6^6},$$

или более общий вид формулы для вычисления

$$B_n = -\frac{C_{n+1}^0 B_0 + C_{n+1}^1 B_1 + C_{n+1}^2 B_2 + \dots + C_{n+1}^{n-1} B_{n-1}}{C_{n+1}^n}. \quad (*)$$

Формулу (*) можно положить в основу программы. Возможный вариант программы — «Генератор чисел Бернулли» — приведен на вклейке.

При составлении программы в качестве подпрограммы использовалась ранее разработанная программа «Вычисление биномиальных коэффициентов».

Об одном применении чисел Бернулли мы уже рассказывали, эти числа входят в общую формулу вычисления суммы:

$$\sum_{k=1}^n k^N = 1^N + 2^N + 3^N + \dots + n^N, \text{ где } k, N \in \mathbb{N}. \quad (*)$$

В заключение привлекаем внимание читателя к формуле, описанной Б. Паскалем в 1819 г. Это формула для вычисления сумм вида (*).

Заметим, что $S_0 = 1^0 + 2^0 + 3^0 + \dots + n^0 = n$,

$$S_1 = 1^1 + 2^1 + 3^1 + \dots + n^1 = \frac{n(n+1)}{2}.$$

Для отыскания S_2 воспользуемся выкладками:

$$(n+1)^3 = n^3 + 3n^2 + 3n + 1,$$

$$(n+1)^3 - n^3 = 3n^2 + 3n + 1,$$

$$\text{при } n=1: 2^3 - 1^3 = 3 \cdot 1^2 + 3 \cdot 1 + 1,$$

$$\text{при } n=2: 3^3 - 2^3 = 3 \cdot 2^2 + 3 \cdot 2 + 1,$$

$$\text{при } n=3: 4^3 - 3^3 = 3 \cdot 3^2 + 3 \cdot 3 + 1,$$

$$\begin{aligned} (n+1)^3 - n^3 &= 3(n+1)^2 + 3(n+1) + 1, \\ (n+1)^3 - 1^3 - S_2 + 3S_1 + S_0 &= 0. \end{aligned}$$

Отсюда легко получить искомую сумму S_2 как функцию от n (при $k=2$), $S_2 = \frac{n \cdot (n+1) \cdot (2n+1)}{2}$. Обобщая, получают формулу:

$$(n+1)^{k+1} = n^{k+1} + C_{k+1}^1 n^k + C_{k+1}^2 n^{k-1} + C_{k+1}^3 n^{k-2} + \dots + 1$$

и отсюда выводят искомую формулу, которую можно положить в основу вычислений:

$$(n+1)^{k+1} - 1 = (k+1)S_k + C_{k+1}^2 S_{k-1} + S_{k+1}^3 S_{k-2} + \dots + S_0.$$

В этой формуле S_k можно выразить через все предшествующие суммы S_i ($i \leq k-1$):

$$S_k = ((n+1)^{k+1} - C_{k+1}^2 \cdot S_{k-1} - S_{k+1}^3 \cdot S_{k-2} - \dots - S_0 - 1) / (k+1).$$

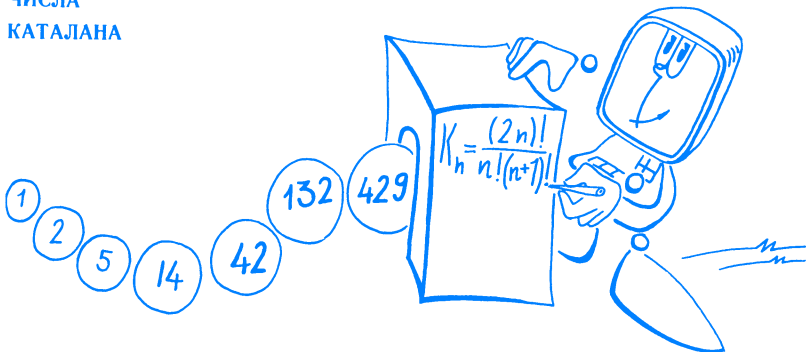
В завершение рассказа о генераторе чисел Бернулли упомянем о любопытном факте.

В первой половине XIX века английский математик Чарльз Бэббедж (1792—1871) работал над созданием Универсальной, или Аналитической, машины. «Очерк аналитической машины, изобретенной Бэббеджем», написанный итальянским военным инженером Л. Менабреа, был переведен на английский язык и прокомментирован леди Адой Лавлейс (1843).

Дочь поэта Дж. Байрона леди Ада Лавлейс (1815—1852) оказалась первой программисткой в мире. Свою программу для решения задачи на машине она назвала «Список операций для вычисления чисел Бернулли». Программа включала условную передачу управления, циклы и другие приемы программирования, которые мы сегодня используем очень широко.

Заслуги Ады Лавлейс перед программированием увековечены тем, что в ее честь один из языков программирования высокого уровня носит имя Ада.

ЧИСЛА КАТАЛАНА



Первые семь чисел Каталана: 1, 2, 5, 14, 42, 132, 429, ... бросают вызов любознательному школьнику. Какая формула может выступить генератором этих чисел? Есть ли общая формула, пользуясь которой и зная номер числа Каталана, можно вычислить число Каталана? Может быть, есть подходящая рекуррентная формула?

Каталан Эжен Шарль (1814—1894) — бельгийский математик, написал более 200 мемуаров по различным вопросам математики. Высказал предположение, что уравнение $x^2 - y^2 = 1$ не имеет решений для x, y, z и t — натуральных, больших 1, кроме одного $3^2 - 2^3 = 1$. Эта проблема до сих пор еще не решена.

Оказывается, для вычисления чисел Каталана есть и формула общего члена:

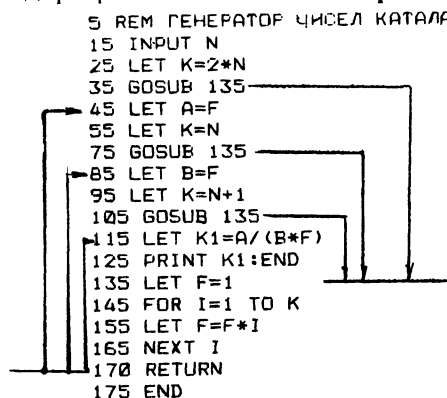
$$K_n = \frac{(2n)!}{n!(n+1)!}$$

и рекуррентная формула:

$$\text{для } n \geq 3 \quad K_n = \frac{K_{n-1}(4n-6)}{n} \text{ и } K_1 = K_2 = 1.$$

Найдены математиками и другие формулы.

Каждую из этих формул можно положить в основу программы для вычисления чисел Каталана. В первой программе вновь используется подпрограмма «Вычисление факториала числа К».



ВЫВОД ПЕРВЫХ 20 ЧИСЕЛ КАТАЛАНА

1	2	5	14				
42	132	429	1430	4862			
16796	58786	208012	742900	2674440			
9694845	35357670	129644790	477638700	1767263190			

Проще разработать программу для вычисления чисел Каталана, если в ее основу положить рекуррентную формулу:

$$K_n = \frac{K_{n-1}(4n-6)}{n} \text{ для } n \geq 3, \text{ при } K_1 = 1 \text{ и } K_2 = 1.$$

Программа в этом случае может быть, например такой, как приведенная ниже:

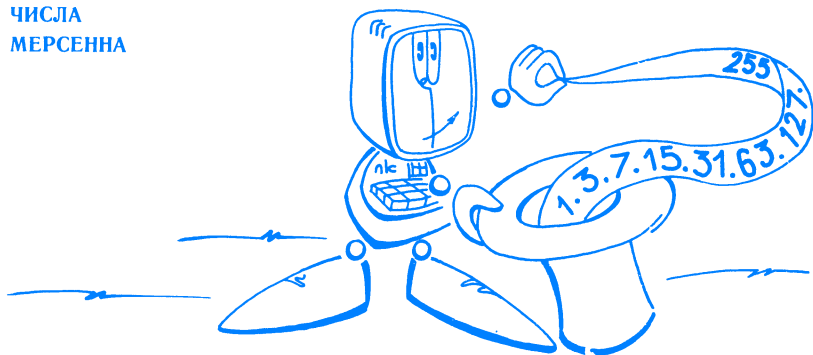
```

10 REM ГЕНЕРАТОР ЧИСЕЛ КАТАЛАНА
15 INPUT N: DIM K(N): LET K(2)=1
20 PRINT K(2):
25 FOR I=3 TO N
30 LET K(I)=(K(I-1)*(4*I-6))/I
40 PRINT K(I);
45 NEXT I
50 END

```

Действуя по такой программе, ЭВМ каждое найденное число Каталана сразу же будет выводить на экран (на печать).

Составление такой программы можно считать небольшим упражнением по теме: «Программирование вычислений по рекуррентным формулам». Программисты часто обращаются к таким приемам, ниже будет рассказано, как «сведение к рекуррентной формуле» облегчает решение задачи. Оказывается, иногда очень важно получать результат не сразу (например, вычислить по формуле общего члена), а «шаг за шагом» — новый результат полезно выводить из предыдущего.



Числа вида $M_n = 2^n - 1$ необычайно популярны в математике. С одним из примеров мы уже встречались — шахматное число оказалось числом именно такого типа: $M_{64} = 2^{64} - 1$. О некоторых применениях чисел вида $2^n - 1$ речь пойдет ниже.

Мерсенн Марен (1588—1648) — французский физик, математик и философ. Переписывался с Б. Паскалем, Р. Декартом, П. Ферма и другими замечательными учеными. Изучал совершенные числа и получил серьезные результаты.

Выпишем несколько первых членов последовательности, составленной из этих чисел:

$$1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, \dots \quad (*)$$

Среди членов последовательности (*) встречаются как простые, так и составные числа. Простыми числами из чисел вида $2^n - 1$ интересовался Мерсенн, в его честь они и называются *числами Мерсенна*.

Поставим программисту задачу — составить программу «Генератор чисел Мерсенна». Как всегда, необходимо изучить проблему и найти основную идею будущего алгоритма и программы. В данном случае используем известный теоретический результат, который обычно формулируется так: число $2^n - 1$ может быть простым, если n — число простое. Отсюда не следует, что если n — число простое, то $2^n - 1$ всегда простое. При некоторых простых n число $2^n - 1$ оказывается составным. Например, при $n=11$ (11 — число простое, $2^{11} - 1 = 2047$ оказывается составным: $2047 = 23 \cdot 89$). Ясно, однако, что если n — составное, то и $2^n - 1$ также составное. (Докажите это утверждение самостоятельно).

Из сказанного сделаем вывод: числа Мерсенна следует искать только среди чисел вида $2^n - 1$, если n — простое. Наша задача — составить программу, работая по которой ЭВМ будет генерировать одно за другим числа Мерсенна.

Учтем еще одно обстоятельство: при выяснении, является ли число $M = 2^n - 1$ числом простым, будем делить это число на

простые числа, не превышающие \sqrt{M} , и если исследуемое число не делится ни на одно предшествующее \sqrt{M} простое число, то оно само является простым. Пусть, например, исследуемое число $2^5 - 1 = 31$. Для выяснения, является ли оно простым, будем делить его на 2, затем на 3, на 5, на 7, а на числа простые, но большие, чем $\sqrt{31}$, делить уже не будем. В данном примере число 31 не делится нацело ни на одно из предшествующих $\sqrt{31}$ простых чисел, следовательно, число 31 — простое и при этом является числом Мерсенна.

Приведенных соображений достаточно, чтобы начать разработку программы. Предлагаем изучить следующий вариант программы «Генератор чисел Мерсенна»:

```

5 REM ГЕНЕРАТОР ЧИСЕЛ МЕРСЕННА
8 INPUT "КОЛ-ВО ИСКОМЫХ ЧИСЕЛ МЕРСЕННА N";N
10 DIM P(90),M(15):LET L=1
15 REM P(90) - МАССИВ ПРОСТЫХ ЧИСЕЛ
20 REM M(15) - МАССИВ МЕРСЕННА
25 DATA 2,3,5,7,11,13,17,19,23,29,31,37,41,43
30 DATA 47,53,59,61,67,71,73,79,83,89,97,101
35 DATA 103,107,109,113,127,131,137,139,149
40 DATA 151,157,163,167,173,179,181,191,193
45 DATA 197,199,211,223,227,229,231,233,239
50 DATA 241,251,257,263,269,271,277,281,283
55 DATA 293,307,311,313,317,331,337,347,349
60 DATA 353,359,367,373,379,383,389,397,401
65 DATA 409,419,421,431,433,439,443,457,461,463
70 REM ФОРМИРОВАНИЕ МАССИВА P(150)
75 FOR I=1 TO 90
80 READ P(I):NEXT I
85 REM ФОРМИРОВАНИЕ ЧИСЛА - КАНДИДАТА В
90 REM ЧИСЛА МЕРСЕННА
95 FOR I=1 TO N-1
100 LET C=2^P(I+1)-1
105 LET Q=SQR(C):LET G=INT(Q/LOG(Q))+1
110 IF C=3 THEN LET G=1
115 FOR K=1 TO G
120 LET D=C-INT(C/P(K))*P(K)
125 IF D=0 THEN 140
130 NEXT K
135 LET M(L)=C:LET L=L+1
140 NEXT I
145 REM ВЫВОД МАССИВА M(15)
150 FOR I=1 TO L-1
155 PRINT M(I);:NEXT I
160 END

```

В этом варианте сначала формируется рабочий массив простых чисел p . В нашем примере в этот массив входят числа 2, 3, ..., 463 — всего 90 простых чисел. ЭВМ сама из этих чисел, заранее внесенных в DATA, формирует массив P . Этих чисел достаточно, чтобы искать число Мерсенна в интервале от 3 до 90 000.

Кандидаты в числа Мерсенна формируются в виде: $C = 2^{P(I+1)} - 1$. Важное значение имеет переменная G , которая показывает, сколько простых чисел предшествует числу $Q = \sqrt{C}$.

Число G задает количество шагов при проверке числа C «на простоту».

В программе используются четыре цикла:

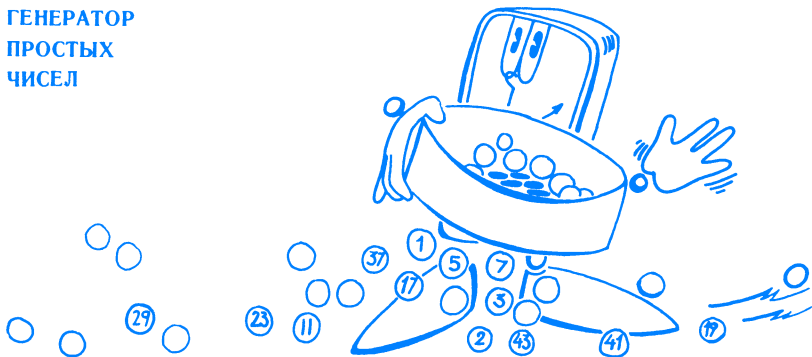
цикл формирования вручную массива из первых N простых чисел;

цикл вывода на печать массива полученных чисел Мерсенна;

цикл, содержащий в теле цикла еще один цикл. Внешний цикл (параметр I) для формирования чисел кандидатов в числа Мерсенна, внутренний цикл (параметр K) для проверки кандидатов на качество «быть простым».

Подчеркнем, что мы не ставим задачу достижения рекордов в поиске чисел Мерсенна. Для нас главное — получить работающий генератор этих чисел. Заметим только, что именно с помощью ЭВМ были установлены числа Мерсенна — «рекордсмены». Назовем несколько из них: в 1978 году получено 25-е число: $2^{21701}-1$, в 1979 году получено 26-е и 27-е числа: $2^{23209}-1$ и $2^{44497}-1$. Это огромные числа, не каждая ЭВМ в состоянии осуществить все вычисления, связанные с достижением таких «рекордов».

ГЕНЕРАТОР ПРОСТЫХ ЧИСЕЛ



дает только простые числа. Вторая — $P(n)=n^2+n+41$ — дает 40 простых чисел, если $0 \leq n \leq 39$.

Эйлер Леонард (1707—1783) — швейцарский математик, физик, механик и астроном. В 1726—1741 гг. работал в России, был академиком. В 1741—1766 гг. работал в Берлине. В 1766 году вернулся в Россию. Эйлер — автор более 860 научных трудов, из них более 140 посвящено теории чисел.

Какие же из простых чисел дают эти формулы? Ответ на этот вопрос может дать ЭВМ, работая по программе:

```
5555 REM ГЕНЕРАТОР ПРОСТЫХ ЧИСЕЛ ПО ЭЙЛЕРУ
10 DIM P1(40)
20 FOR N=0 TO 39
30 LET P1(N) = N^2 + N + 41
40 NEXT N
50 FOR I=0 TO 39
70 PRINT P1(I);
85 NEXT I
90 END
```

Результаты своей работы ЭВМ выдает в виде таблицы, содержащей 40 простых чисел.

```
41  43  47  53  61  71  83  97
113 131 151 173 197 223 251 281
313 347 383 421 461 503 547 593
641 691 743 797 853 911 971 1033
1097 1163 1231 1301 1373 1447 1523 1601
```

Если посмотреть на первую строку таблицы, то может показаться, что генерируются простые числа подряд, без пропусков: 41, 43, 47, 53, Однако это не так. Продвинувшись по таблице, выданной ЭВМ, мы обнаружим, что простые числа 67, 73, 79, 101, 103, 109 по формуле Эйлера n^2+n+41 не вычисляются. Генерируя только простые числа, ЭВМ генерирует их не по порядку, а с пропусками. В таблице 4 показана часть полной

Таблица 4

41	43	47	53		61		71
		83		97			
	113		131				151
			173				
197			223				
	251						281
				313			
347							383

таблицы простых чисел, начинающихся с числа 41. Записаны те числа, которые вычисляет ЭВМ по приведенной программе.

Как же получить все простые числа, меньшие 41, большие 1601 (последнее простое число, вычисленное машиной) и как найти те, которые были пропущены ЭВМ?

Ответы на некоторые вопросы можно получить, если поручить ЭВМ вести вычисления по другой производящей формуле Эйлера или по формуле Лежандра: $P(n)=2n^2+29$ для $0 \leq n \leq 29$, или по формуле Эскотта: $P(n)=n^2-79n+1601$ для $0 \leq n \leq 79$, или по формуле: $\frac{2^{n+1}+1}{3}$ для $1 \leq n \leq 79$.

Лежандр Адриен Мари (1752—1833) — французский математик. Двухтомный труд «Теория чисел» выдержал несколько изданий и был самым полным изложением теории чисел того времени.

Такими методами мы найдем многие простые числа, но удовлетворения не получим. Это не систематическая генерация всех простых чисел подряд и без пропуска. Математики не имеют единой формулы $F(N)$ для генерирования чисел подряд, начиная с 1. Нет в распоряжении математиков и рекуррентной формулы, аналогичной формуле для вычисления чисел Фибоначчи или Бернулли. Раз так, то остается создавать алгоритм, приемлемый для реализации на ЭВМ. В основе создаваемого алгоритма будет лежать только определение простого числа.

Из сказанного видно, что создание генератора простых чисел — это программистская задача, заметно отличающаяся от ранее рассмотренных. Обсудим некоторые подходы. Рассмотрим сначала *решето Эратосфена*, знакомое школьникам с пятого класса.

Эратосфен Киренский (около 276—194 до н. э.) — древнегреческий ученый. Занимался хронологией, астрономией, филологией, философией, музыкой. Построил прибор для решения задачи об удвоении куба; первый измерил дугу меридиана.

Прежде всего создадим остов решета — одномерный массив, в который поместим натуральные числа 1, 2, 3, 5, 7, 9, 11, 13, 15, Начиная с числа 3, все остальные числа будут нечетными. Если мы намерены искать простые числа среди натуральных, не превышающих, например, 5000, то последнее число в создаваемом массиве будет число 4999. Почему мы так делаем, понятно, ведь среди четных чисел простое число одно — это 2, других нет. Зачем же загружать память ЭВМ посторонней информацией? Далее вспомним, что простые числа, это числа, которые делятся только на себя и на единицу. Отсевать в решете Эратосфена будем числа составные, простые будут в нем оставаться.

Примем во внимание важное для составления программы обстоятельство: выясняя, является ли рассматриваемое число N простым, мы будем делить его на предшествующие ему простые

числа. И, если ни на одно из предшествующих простых чисел N не делится, то оно само является простым. Кажется, все правильно, однако в проводимой таким образом работе есть лишние шаги. Проверку на делимость числа следует заканчивать после того, как проверена делимость на простое число, не превышающее \sqrt{N} , вернее, «целую часть от \sqrt{N} ».

Следует учесть, что простые числа в процессе отсеивания будут обнаруживаться одно за другим, и из них целесообразно формировать массив простых чисел P . Сразу возникает вопрос: на какое количество чисел следует его планировать? Иначе говоря, мы должны узнать, сколько простых чисел предшествует числу 5000. Ответ на такой вопрос в общем виде дается в теории простых чисел. Если x — данное натуральное число, то $\pi(x) = x / \ln x$ (вернее, целая часть от $x / \ln x$) есть количество предшествующих ему простых чисел. Это приближенное равенство становится все более точным по мере увеличения числа x . Открыл эту формулу в 1792 году пятнадцатилетний мальчик Карл Гаусс — он увидел такую закономерность, изучая таблицы простых чисел в подаренной ему книге о логарифмах. Чуть более ста лет прошло, когда эту формулу доказали строго.

Гаусс Карл Фридрих (1777—1855) — немецкий математик, астроном, физик и геодезист. Работы Гаусса оказали большое влияние на все дальнейшее развитие высшей алгебры, теории чисел, дифференциальной геометрии, теории притяжения, классической теории электричества и магнетизма, геодезии, многих отраслей теоретической астрономии.

Французский математик А. Лежандр дал уточненную формулу:

$$\pi(x) = \frac{x}{\ln x - 1,0837}.$$

Сам К. Гаусс позднее определил функцию так:

$$\pi(x) = \frac{1}{\ln 2} + \frac{1}{\ln 3} + \dots + \frac{1}{\ln x}.$$

Учитывая сказанное, можно достаточно точно определить объем памяти для массива отсортировываемых простых чисел.

Напомним, что ответ на вопрос «делится ли a нацело на b » дается с помощью функции «дробная часть аргумента»: если a делится нацело на b , то значение функции равно нулю.

На рисунке 16 приведена схема алгоритма решения задачи. В схеме два цикла. Во внутреннем цикле идет проверка делимости «кандидата» в простые числа $P(J+1)$ на все предшествующие ему простые числа $P(J)$. Во внешнем цикле формируются «кандидаты» в простые числа и формируется массив найденных простых чисел P . Обращаем внимание на оператор $P(I+1) = P(I)$, здесь $P(I)$ всегда простое число, а $P(I+1)$ далее увеличивается на 2 и становится обследуемым «кандидатом» в простые числа.

Текст программы может быть таким:

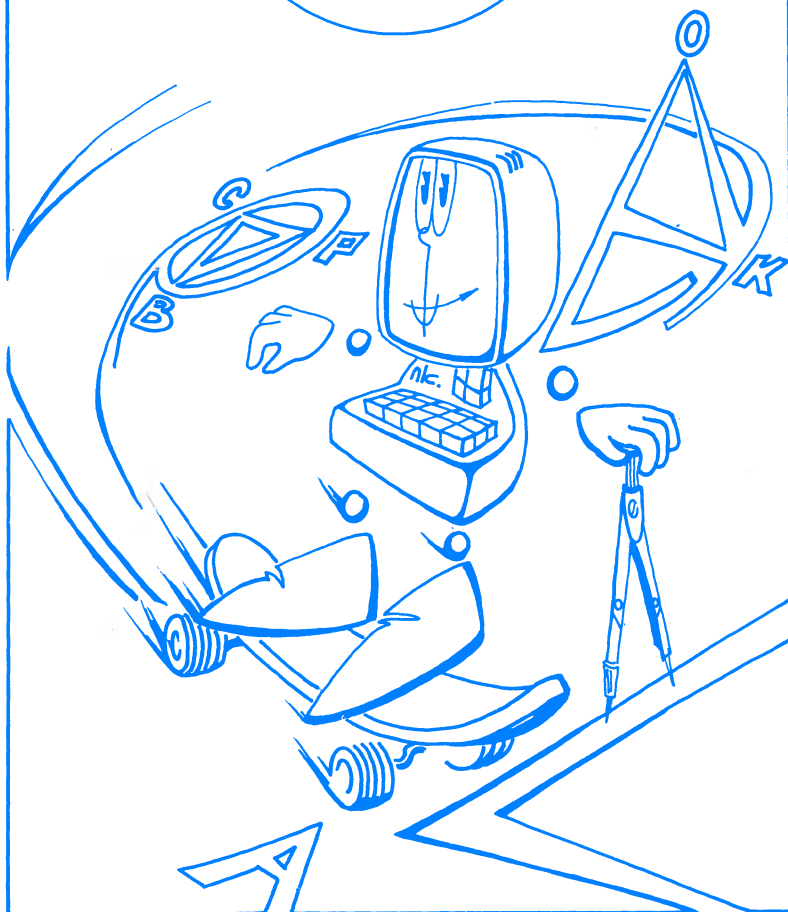
```
5 REM ГЕНЕРАТОР ПРОСТЫХ ЧИСЕЛ
10 INPUT N: LET P1=INT(N/LOG(N)):PRINT P1: PRINT
20 DIM P(P1+1): LET P(1) = 1: LET P(2) = 2: LET P(3) = 3
30 FOR I=3 TO P1
40 LET P(I+1) = P(I)
50 LET P(I+1) = P(I+1) + 2
60 FOR J=3 TO I
70 LET S = P(I+1) - INT(P(I+1)/P(J))*P(J)
80 IF S=0 THEN 50
90 NEXT J
100 NEXT I
110 REM ВЫВОД РЕЗУЛЬТАТА
120 FOR I=1 TO P1
130 PRINT P(I);
140 NEXT I
150 END
```

ГЛАВА

II.

В ГЕОМЕТРИЮ

С ЭВМ



ПЕРВЫЕ ШАГИ ЭВМ В ГЕОМЕТРИИ. ЕЩЕ ОДИН ШАГ В ГЕОМЕТРИЮ. ВСТРЕЧА НА ОРБИТЕ. ФОРМУЛА КАРДАНО НА ЭВМ.

*Помните: программы пишутся
для машин, а читаются людьми.*

Из фольклора
программистов

Метод координат — мост между геометрией и алгеброй. Без ЭВМ при использовании алгебраических моделей геометрического мира не обойтись. В возможностях ЭВМ применять циркуль, линейку, а также новые необычные геометрические инструменты для решения геометрических задач и задач графических.

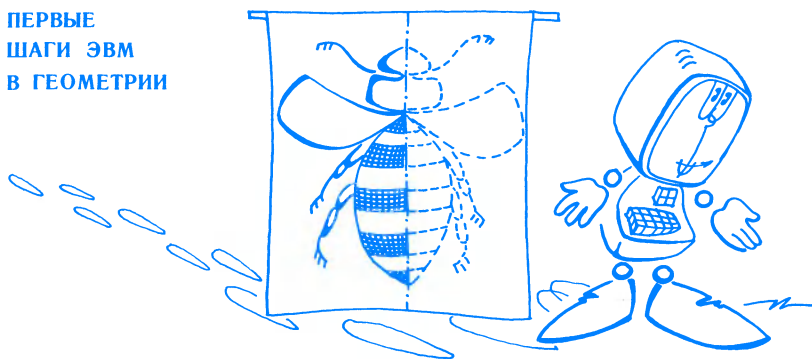
Черчение вручную — утомительно, непроизводительно. С появлением ЭВМ все изменилось, ЭВМ — дисциплинированный, аккуратный и неутомимый помощник.

Рассказ о методе координат — одна из целей этой главы. Вы убедитесь в необычайной эффективности метода, в огромном значении связи алгебры с геометрией, узнаете, как усилиями математиков метод координат приобрел современное звучание.

В этой же части книги рассказывается о том, как ЭВМ выполняет основные геометрические преобразования: центральную и осевую симметрии, параллельный перенос, поворот вокруг точки и другие. На базе таких умений строится решение непростых геометрических задач. Например, задачу «Шар на бильярде» совсем непросто решить вручную, особенно когда бильярд имеет форму не прямоугольника, а правильного выпуклого n -угольника.

Отдельно рассматриваются графические средства языка Бейсик, приводятся тексты программы, действуя по которым, ЭВМ изображает фигуры на экране. При этом удастся моделировать движение фигур.

ПЕРВЫЕ
ШАГИ ЭВМ
В ГЕОМЕТРИИ



Начало рассказа — в далеком прошлом, можно сказать, что все началось 8 июня 1637 года. Именно в этот день увидело свет произведение замечательного французского математика Рене Декарта «Рассуждения о методе, позволяющем направлять свой разум и отыскивать истины в науках. Кроме того, диоптрика, метеоры, геометрия, которые являются применением этого метода». Сегодня этот научный труд называют кратко: «Метод». Своей работой Декарт совершил переворот в математике, а миру, науке подарил метод координат, аналитическую геометрию и сделал первый шаг в осознании и продвижении идеи моделирования. Декартовы алгебраические модели геометрического мира потрясли ученых — его современников.

Все это происходило в то время, когда литературный герой А. Дюма мушкетер Д'Артаньян совершал свои подвиги, а коварный кардинал Ришелье плел паутину своих интриг. Можно было бы и не говорить о кардинале, но именно он волею случая сыграл удивительную роль в истории, о которой идет речь: Ришелье дал Декарту привилегию публиковать во Франции и за границей все, что он сочтет нужным. Так появился на свет «Метод». После смерти Декарта церковь запретила его труды как еретические и опасные для государства.

Декарт Рене (1596—1650) — французский философ, математик, физик, физиолог. Основные произведения Декарта: «Правила для руководства ума», «Трактат о свете», «Рассуждения о методе», «Начала философии».

В работе Декарта для нас главным является введение метода координат. Сегодня этот метод шагнул за пределы математики. Где только ни встретишься с координатами: в географии, в мореплавании, в космических полетах. Даже шахматисты ход поединка описывают, используя координаты, связанные с шахматной доской. Е2 — Е4 — такая запись ясно показывает, какой ход и какой фигурой осуществил шахматист. С методом координат знакомятся на уроках семиклассники и быстро разбираются в его деталях, учатся решать простые задачи.

Заслуга Декарта в том, что он нашел путь, связывающий геометрию с алгеброй. Как мост между двумя областями математики работает метод координат. Добавим, что с появлением ЭВМ движение по этому мосту стало более оживленным.

Вернемся к нашей истории. Древние математики, занимавшиеся геометрией, в качестве инструмента для построений пользовались только циркулем и линейкой без делений. Все построения, всякий геометрический синтез, выполненный с помощью других инструментов, объявлялся «механическим», рассматривался как недостойный и строго запрещался в пользовавшейся уважением геометрии. Такой взгляд на геометрию утверждал философ Платон. Лишь только когда Декарт опубликовал свою аналитическую геометрию, классическая, или, как ее часто называют, синтетическая, геометрия освободилась от столь жестких ограничений Платона.

Платон (ок. 427 — 347 гг. до н. э.) — древнегреческий философ, ученик Сократа. Платон многое сделал в становлении логики и элементарной математики.

Благодаря методу координат сегодня к решению задач по геометрии привлекаются такие «инструменты», как ЭВМ.

Как это делается? Покажем на примере. Пусть дана задача: построить прямоугольный треугольник по двум его медианам, проведенным к катетам.

Попросим решить эту задачу геометра, который не будет выходить за рамки синтетической геометрии, и математика, который знает, как для решения таких задач следует применять ЭВМ.

Геометр может предложить такое решение.

Пусть отрезки AB и CD — данные медианы. Построение начинаем с того, что на прямой l откладываем отрезок AB и делим его точкой K в отношении $3:1$, то есть $AK=3 \cdot KB$ (рис. 17). Затем на отрезке AK , как на диаметре, строим окружность. После этого из точки B радиусом $r=1/2 \cdot CD$ делаем засечку на построенной окружности (получаем точку F). Затем из точки B на прямую AF опускаем перпендикуляр BM (рис. 18).

После этого на прямой MB откладываем отрезок BC , равный отрезку MB . В завершение построения соединяем точки A и C . Треугольник AMC — искомый (рис. 19).

Не правда ли, очень интересное и, кажется, совсем несложное решение? Однако додуматься до него не так-то просто, нужно иметь подготовленное к такой работе и тренированное воображение. Каким образом человек догадывается о таких своеобразных и неожиданных решениях, мы не знаем. Это удивительные проявления мысли.

Как эта задача будет решаться с помощью ЭВМ?

Работу начнем с анализа задачи. Сначала треугольник размещаем в системе координат так, как показано на рисунке 20. Отрезки AB и CD известны; задача в том, чтобы найти координаты

наты вершин A и C . Обозначим $AB=a$, $CD=b$. Рассматривая $\triangle OAC$, замечаем, что точка A имеет координаты $(0; y_A)$, точка B имеет координаты $((x_C)/2; 0)$.

Напомним, что сегодня в средней школе используется формула для вычисления расстояния между двумя точками. Рисунок 21 напоминает суть дела. На рисунке 20 медиана AB задана точкой $A(0; y_A)$ и точкой $B((x_C)/2; 0)$, следовательно $((x_C)/2)^2 + y_A^2 = a^2$ (a — длина AB). Аналогично и для другой медианы $x_C^2 + ((y_A)/2)^2 = b^2$ (b — длина CD).

В полученных уравнениях неизвестными являются координаты y_A и x_C (далее индексы опускаются). Имеем:

$$\begin{cases} x^2 + 4y^2 = 4a^2, \\ 4x^2 + y^2 = 4b^2. \end{cases}$$

Решаем систему:

$$\begin{array}{ll} \begin{cases} x^2 + 4y^2 = 4a^2, \\ 4x^2 + y^2 = 4b^2, \end{cases} & \begin{cases} x^2 + 4y^2 = 4a^2, \\ \frac{16x^2 + 4y^2 = 16b^2}{15x^2 = 16b^2 - 4a^2}, \end{cases} \\ \begin{cases} x^2 + 4y^2 = 4a^2, \\ 4x^2 + y^2 = 4b^2, \end{cases} & \begin{cases} 4x^2 + 16y^2 = 16a^2, \\ \frac{4x^2 + y^2 = 4b^2}{15y^2 = 16a^2 - 4b^2}. \end{cases} \end{array}$$

Вычисления завершаются по формулам:

$$x = \sqrt{\frac{16b^2 - 4a^2}{15}} \quad \text{и} \quad y = \sqrt{\frac{16a^2 - 4b^2}{15}}.$$

Эти вычисления и вычерчивание искомого треугольника на экране дисплея можно поручить ЭВМ, работа которой осуществляется в два этапа:

1 этап — вычислительный, по программе:

```
5 INPUT "ВВОДЯТСЯ ДЛИНЫ МЕДИАН А,В"; А,В
10 LET X=SQR((16*(В^2)- 4*(А^2))/15)
20 LET Y=SQR((16*А^2 - 4*В^2)/15)
30 PRINT "X=";X, "Y=";Y
40 END
```

2 этап — изображение искомого треугольника (рис. 22) на экране по программе:

```
5 INPUT X,Y
10 COLOR 1,15,3: SCREEN 2
20 LINE (30,40)-(30+X,40),3
30 LINE (30,40)-(30,40+Y),3
40 LINE (30+X,40)-(30,40+Y),3
50 GOTO 50
```

Вторая программа нуждается в разъяснениях. Прежде всего поясним оператор `SCREEN 2`. С его помощью заказывается так называемый графический экран — именно на таком экране ведется

изображение геометрических фигур. Экран позволяет обращаться к каждой его точке. На экране 255 точек по горизонтали в каждом из 191 ряда, что позволяет демонстрировать на экране очень четкие чертежи. В операторе COLOR можно задать цвет линии основного чертежа, цвет фона и цвет рамки, окаймляющей весь рисунок на экране. Цвета линий, фона и рамки выбираются программистом из предлагаемых 15 цветов. Оператор 160 GO TO 160 удерживает рисунок на экране.

Обе вышеприведенные программы могут быть объединены в одну, и окончательные тексты программы и чертеж на экране могут быть, например, такими, как показано ниже:

```

10 REM ПОСТРОЕНИЕ ПРЯМОУГОЛЬНОГО ТРЕУГОЛЬНИКА
20 INPUT "ВВОДЯТСЯ ДЛИНЫ МЕДИАН"; A,B
30 LET X=SQR((16*B^2-4*A^2)/15)
40 LET Y=SQR((16*A^2-4*B^2)/15)
50 PRINT "X="X"Y="Y
60 LET K=6
70 COLOR 1,15,8:SCREEN 2:FOR I=10 TO (15+X)*3 STEP 15+X
80 LINE (I,40)-(I+X,40),K
90 LINE (I,40)-(I,40+Y),K
100 LINE (I+X,40)-(I,40+Y),K
110 K=K+2.5:NEXT I
120 PAINT (11,41),6:PAINT (2*X+41,41),11
130 LINE (25+1.5*X,40)-(25+X,40+Y),8
140 LINE (25+X,40+.5*Y)-(25+2*X,40),8
150 GOTO 150

```

На рисунке 23, который вывела ЭВМ на экран, не один, а три треугольника. Один из них закрашен в красный цвет, во втором проведены дополнительно обе данные медианы. Медианы синего цвета, а третий треугольник закрашен в синий цвет.

Построение треугольников и их закрашивание велось в цикле. Подчеркнем, что в приведенной программе геометрическая часть заняла всего четыре строки, остальные операторы обеспечивают демонстрацию результата на экране.

Среди основных графических средств Бейсика есть оператор CIRCLE, предназначенный для рисования на экране окружностей, дуг и овалов различного цвета и размера. Используя такой оператор и некоторые программистские «находки», можно составить программу, работая по которой ЭВМ продемонстрирует «геометрический мультфильм». На экране шаг за шагом будут вычерчиваться все отрезки и дуги, в завершение на экране будет оставлен только искомый треугольник.

Заканчивая рассказ о простейших геометрических умениях ЭВМ, приведем программу, работая по которой ЭВМ сумеет провести две касательные к окружности из точки, лежащей вне ее. В этой программе также две части, в первой ведутся вычисления, во второй осуществляется вычерчивание всех линий. И в этой программе чертеж «живет» — все построения следуют одно за другим так, как это делает геометр, решающий задачу.

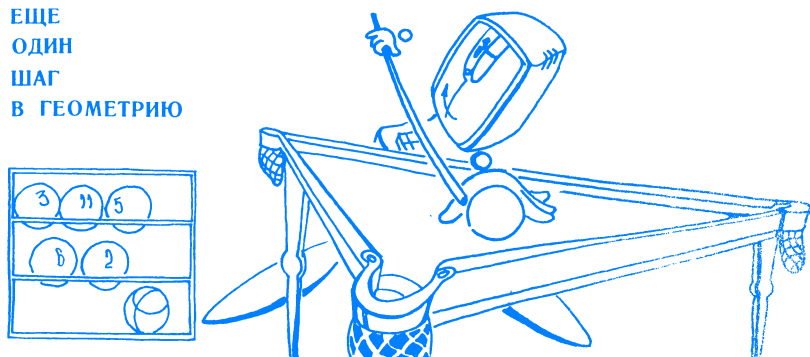
```

5  REM  ПОСТРОЕНИЕ КАСАТЕЛЬНОЙ К ОКРУЖНОСТИ
      ИЗ ТОЧКИ ВНЕ ЕЕ
10  INPUT 'ВВОД КООРДИНАТ ЦЕНТРА ОКРУЖНОСТИ
      И ЕЕ РАДИУСА' X0,Y0,R
15  INPUT 'ВВОД КООРДИНАТ ТОЧКИ' XA,YA
20  LET X1=(X0+XA)/2:LET Y1=(Y0+YA)/2
25  LET Q=(X0-X1)**2+(Y0-Y1)**2:LET L=R*R
30  LET A=X0-X1:LET B=Y0-Y1:LET M=X1*X1-X0*X0+Y1*Y1-Y0*Y0
35  LET N=Q-L:LET C=N-M:LET D=C/(2*B)-Y1:LET K=A/B
40  REM  ВЫЧИСЛЕНИЕ КООРДИНАТ ТОЧЕК КАСАНИЯ
45  LET X2=(F+SQR(F*F-P*Z))/P
50  LET X3=(F-SQR(F*F-P*Z))/P
55  LET Y2=(C-2*A*X2)/(2*B)
60  LET Y3=(C-2*A*X3)/(2*B)
65  PRINT X2;Y2;X3;Y3
67  REM  ПРОВЕДЕНИЕ ИСКОМЫХ КАСАТЕЛЬНЫХ
70  SCREEN 2:COLOR 1,15,8
75  LINE (XA,YA)-(X2,Y2),1
80  LINE (XA,YA)-(X3,Y3),1
85  GOTO 85
90  END

```

Решение рассмотренных геометрических задач велось по единой схеме. Сначала геометрическая задача на основе использования метода координат «пересказывалась» как алгебраическая. От задачи геометрической осуществлялся переход к ее алгебраической модели. В работе с алгебраической моделью ЭВМ выступала главным действующим лицом. Основное — перенесение задачи из геометрии в алгебру, создание хорошей алгебраической модели — это дело человека. Придумать модель — дело для ЭВМ непосильное.

ЕЩЕ
ОДИН
ШАГ
В ГЕОМЕТРИЮ



Любой школьник легко построит точку A_1 , симметричную точке A относительно некоторой данной точки O , называемой центром симметрии. Рисунок 24 напоминает, как это делается. Все три точки A , A_1 и O лежат на одной прямой и $OA=OA_1$.

Как эту задачу может решить ЭВМ? Остановимся на решении, в котором вновь будет использован метод координат. Сформули-

руем задачу иначе: по известным координатам точек $A(x_A; y_A)$ и точки $O(x_0; y_0)$ найти координаты точки $A_1(x_{A_1}; y_{A_1})$, симметричной точке A относительно точки O . После вычисления координат точки A_1 ЭВМ должна показать на экране три точки: A , A_1 и O . Точки A и A_1 соединить отрезком.

Рисунок 25 показывает, как следует вести вычисление координат точки A_1 . Ясно, что $x_0 = \frac{1}{2}(x_A + x_{A_1})$ и $y_0 = \frac{1}{2}(y_A + y_{A_1})$.

Отсюда получаем: $x_{A_1} = 2 \cdot x_0 - x_A$; $y_{A_1} = 2 \cdot y_0 - y_A$.

Программа может быть такой, как приведенная ниже.

```

5  ' ПОСТРОЕНИЕ ТОЧКИ СИММЕТРИЧНОЙ
15 ' ТОЧКЕ А ОТНОСИТЕЛЬНО Т.О
25 INPUT "ВВОД КООРД. Т. А"; XA, YA
35 INPUT "ВВОД КООРД. Т. О"; XO, YO
45 LET X=2*XO-XA:LET Y=2*YO-YA
55 PRINT "КООРД. ИСКОМОЙ Т. А1"; X, Y
65 SCREEN 2:COLOR 1, 15, 3
75 LINE (XA, YA)-(X, Y), 8
85 PSET (XA, YA), 1:PSET (XO, YO), 1:PSET (X, Y), 1
95 GOTO 95

```

Первая часть программы, вычислительная, завершается выводом на экран координат точки A_1 (см. строку 55). Во второй части программы выписаны операторы, задающие все геометрические построения. На экране белого цвета проведен красным цветом отрезок AA_1 , точки A , A_1 и O выделены черным цветом (рис. 26).

Опираясь на составленную программу, можно разработать программу построения треугольника, симметричного треугольнику ABC относительно точки $O(x_0; y_0)$.

Текст программы может быть таким, как приводимый ниже. И в этой программе есть часть вычислительная и часть графическая. Вычисления координат вершин искомого треугольника проводятся в цикле (строки 65—95).

```

5  ' ПОСТРОЕНИЕ ТР-КА А1 В1 С1
15 ' СИММЕТРИЧНОГО ТР-КУ А В С
25 ' ОТНОСИТЕЛЬНО Т.О
35 DIM X(3), Y(3)
45 DATA 80, 80, 10, 10, 60, 40, 40, 70
55 READ XO, YO
65 FOR I=1 TO 3
75 READ X, Y
85 LET X(I)=2*XO-X:LET Y(I)=2*YO-Y
95 NEXT I
105 SCREEN 2:COLOR 1, 15, 3
115 LINE (10, 10)-(60, 40), 1
125 LINE (60, 40)-(40, 70), 1
135 LINE (40, 70)-(10, 10), 1
145 LINE (X(1), Y(1))-(X(2), Y(2)), 5
155 LINE (X(2), Y(2))-(X(3), Y(3)), 5
165 LINE (X(3), Y(3))-(X(1), Y(1)), 5
175 PSET (80, 80), 8
185 GOTO 185

```

Исходный треугольник (рис. 27) изображен синим цветом, искомый треугольник — красным. Выделен и центр симметрии. Точка O — белого цвета, цвет экрана — голубой (у нас — белый).

Вычислительную часть программы построения точки, симметричной точке A относительно точки O , можно рассматривать как подпрограмму и применять для решения конкретных задач.

Задача. На плоскости расположены середины сторон многоугольника, имеющего нечетное число сторон. Координаты этих точек известны. Построить многоугольник.

Решение. Для примера на рисунке 28 изображены середины сторон некоторого треугольника: $A_1(x_1; y_1)$, $A_2(x_2; y_2)$ и $A_3(x_3; y_3)$. Алгоритм решения задачи прост. В плоскости треугольника выбирают произвольную точку $P_0(x_0; y_0)$ и затем последовательно выполняют построения: строят точку P_1 , симметричную точке A_1 относительно точки P_0 , затем строят точку P_2 , симметричную уже точке P_1 относительно точки A_2 , затем строят точку P_3 , симметричную точке P_2 относительно точки A_3 . После этого находят точку B_1 — середину отрезка P_0P_3 . Далее вновь осуществляется построение симметричных точек. К этому моменту ход работы показан на рисунке 28.

Теперь ищем вершины искомого треугольника. Первая уже найдена — это вершина B_1 . Вершина B_2 симметрична B_1 относительно A_1 , вершина B_3 симметрична B_2 относительно A_3 и, конечно, должна оказаться симметричной точке B_1 относительно A_2 . Рисунок 29 отражает заключительную часть работы.

Текст программы для построения конкретного треугольника приводится ниже.

```

10 REM ПОСТРОЕНИЕ ТРЕУГОЛЬНИКА ПО СЕРЕДИНАМ
20 REM ЕГО СТОРОН
30 CLS: INPUT X1, Y1
40 DATA 40, 40, 85, 60, 50, 80
50 LET X0=X1: Y0=Y1
60 FOR I=1 TO 3
70 READ X, Y
80 LET XP=2*X-X0: LET YP=2*Y-Y0: PRINT "XP="; XP, "YP="; YP
90 LET X0=XP: LET Y0=YP
100 NEXT I
110 DIM X(3), Y(3)
120 LET X(1)=(X0+XP)/2: LET Y(1)=(Y0+YP)/2
130 PRINT "X(1)="; X(1), "Y(1)="; Y(1)
140 RESTORE 40
150 FOR I=1 TO 2
160 READ X, Y
170 LET X(I+1)=2*X-X(I): LET Y(I+1)=2*Y-Y(I)
180 PRINT "X("; I+1; ")="; X(I+1), "Y("; I+1; ")="; Y(I+1)
190 NEXT I
200 STOP
210 CLS: COLOR 4, 15, 15: SCREEN 2
220 LINE (X(1), Y(1))-(X(2), Y(2)), 4
230 LINE (X(2), Y(2))-(X(3), Y(3)), 4
240 LINE (X(3), Y(3))-(X(1), Y(1)), 4
250 LINE (0, 0)-(250, 0), 1
260 LINE (0, 0)-(0, 192), 1
270 GOTO 270

```

В тексте программы использован оператор RESTORE, с его помощью координаты данных точек, помещенные в DATA, восстановлены и их удается использовать еще раз при вычислении координат вершин B_1 , B_2 и B_3 искомого треугольника. В завершение своей работы ЭВМ на экране демонстрирует искомый треугольник и оси координат. В тексте арифметический оператор присваивания выписан без служебного слова LET, в ряде версий языка Бейсик опускать слово LET можно.

Завершая рассказ, приводим более общую программу для построения $(2n+1)$ -угольника по серединам его сторон.

```

10 REM "ПОСТРОЕНИЕ (2N+1)- УГОЛЬНИКА ПО КООРД.
20 REM СЕРЕДИН ЕГО СТОРОН
30 INPUT "L - К-ВО СТОРОН";L
40 DIM X(L),Y(L),U(L+1),V(L+1)
50 INPUT "КООРД. ВСПОМОГАТЕЛЬНОЙ ТОЧКИ";X0,Y0
60 PRINT "ВВОД КООРДИНАТ СЕРЕДИН СТОРОН"
70 FOR I=1 TO L
80 INPUT X(I),Y(I)
90 NEXT I
100 LET X=X0:LET Y=Y0
110 FOR I=1 TO L
120 LET X=2*X(I)-X
130 LET Y=2*Y(I)-Y
140 NEXT I
150 REM ВЫЧИСЛЕНИЕ КООРДИНАТ ПЕРВОЙ ВЕРШИНЫ
160 REM ИСКОМОГО МНОГОУГОЛЬНИКА
170 LET U(1)=(X+X0)/2:LET V(1)=(Y+Y0)/2
180 REM ВЫЧИСЛЕНИЕ КООРД. ОСТАЛЬНЫХ ВЕРШИН
190 FOR K=2 TO L
200 LET U(K)=2*X(K-1)-U(K-1)
210 LET V(K)=2*Y(K-1)-V(K-1)
220 NEXT K
230 REM ИЗОБРАЖЕНИЕ ИСКОМОГО МНОГОУГОЛЬНИКА
240 SCREEN 2:COLOR 1,15,3:CLS
250 FOR J=1 TO L-1
260 LINE (U(J),V(J))-(U(J+1),V(J+1)),2
270 NEXT J
280 LINE (U(1),V(1))-(U(L),V(L)),2
290 GOTO 290
300 END

```

Задача. Построить точку, симметричную данной точке A относительно данной прямой MN .

Сначала рассмотрим задачу, как задачу на построение. Разработаем программу, действуя по которой ЭВМ покажет, как эта точка будет найдена графически. Для этой цели будут использованы операторы LINE и CIRCLE.

Уточним условия задачи: $A(x_A; y_A)$ — координаты точки, для которой разыскивается симметричная точка $A'(x; y)$. Осью симметрии будет прямая, задаваемая двумя точками $M(x_M; y_M)$ и $N(x_N; y_N)$.

Решение задачи в элементарной геометрии разъясняется рисунком 30.

Метод прост, не сложен и соответствующая программа. Прежде чем записать программу, напомним, что

$$R_1 = \sqrt{(x_A - x_M)^2 + (y_A - y_M)^2},$$

и

$$R_2 = \sqrt{(x_A - x_N)^2 + (y_A - y_N)^2}.$$

Текст программы может быть таким, как приведенный ниже.

```

REM ПОСТРОЕНИЕ ТОЧКИ, СИММЕТРИЧНОЙ
5 REM ДАННОЙ ОТНОСИТЕЛЬНО ПРЯМОЙ
5 INPUT "КООРДИНАТЫ ДАННОЙ ТОЧКИ А"; XA, YA
5 PRINT "КООРДИНАТЫ ТОЧЕК ЗАДАЮЩИХ ОСЬ
5 INPUT "СИММЕТРИИ"; XM, YM, XN, YN
5 REM ВЫЧИСЛЕНИЕ РАДИУСОВ R1 И R2
5 LET R1=SQR((XA-XM)^2+(YA-YM)^2)
5 LET R2=SQR((XA-XN)^2+(YA-YN)^2)
5 SCREEN2
5 CIRCLE(XM, YM), R1, 1
Ø5 CIRCLE(XN, YN), R2, 1
15 LINE(XM, YM)-(XN, YN), 15
25 GOTO 125

```

В этой программе вычисление координат искомой точки не осуществляется, вся программа решает задачу построения искомой точки. ЭВМ на экране дисплея демонстрирует отрезок MN и две окружности, проходящие через данную точку A и через искомую точку A_1 .

Еще раз подчеркнем, что работая по программе, приведенной выше, ЭВМ находит искомую точку геометрическими средствами. Как же получить координаты точки A_1 , симметричной данной точке A относительно оси MN ? Сделать это можно двумя способами: вычислить, используя известные координаты точек $A(x_A; y_A)$, $N(x_N; y_N)$, $M(x_M; y_M)$; снять координаты точки A_1 с экрана «вручную».

Остановимся на первом методе, то есть научимся вычислять координаты искомой точки $A_1(x; y)$. Воспользуемся еще раз рисунком и, учитывая, что $AM = A'M$ и $AN = A'N$, получим систему из двух уравнений, относительно неизвестных координат точки:

$$\begin{cases} (x_A - x_M)^2 + (y_A - y_M)^2 = (x - x_M)^2 + (y - y_M)^2, \\ (x_A - x_N)^2 + (y_A - y_N)^2 = (x - x_N)^2 + (y - y_N)^2. \end{cases}$$

Решение этой системы выполняем в общем виде. Опуская детали преобразований, получаем формулы для вычисления искомых координат точки:

$$x = \frac{K}{K^2 + 1} \cdot (y_A - y_M + K \cdot x_M + \frac{x_A}{K}),$$

$$y = K \cdot (x - x_M) + y_M,$$

$$y_{A_1} = 2 \cdot x - x_A; \quad y_{A_1} = 2 \cdot y - y_A,$$

где $K = \frac{y_N - y_M}{x_N - x_M}$.

Эти формулы кладутся в основу программы вычисления координат искомой точки A' . Следует иметь в виду, что прямая MN , высту-

пающая в роли оси симметрии, может оказаться расположенной горизонтально и в этом случае $K=0$, и поэтому вышеприведенные расчетные формулы применять нельзя. Эти формулы дают отказ и в случае, когда MN — вертикальная линия.

Из сказанного следует, что программист должен позаботиться о вычислении значения K в самом начале процесса решения задачи. И, если ось симметрии (прямая MN) вертикальна и ее уравнение есть $x=m$, то координаты искомой точки $A'(x; y)$ вычисляются по формулам:

$$\begin{aligned} y &= y_A, \\ x &= 2 \cdot x_M - x_A = 2 \cdot m - x_A. \end{aligned}$$

Если же ось симметрии (прямая MN) горизонтальна и ее уравнение $y=n$, то координаты искомой точки $A'(x; y)$ вычисляются по формулам:

$$\begin{aligned} x &= x_A, \\ y &= 2 \cdot y_M - y_A = 2 \cdot n - y_A. \end{aligned}$$

Текст программы может быть таким, как приведенный ниже.

```

10 ' ВЫЧИСЛЕНИЕ КООРДИНАТ Т. А1
20 ' СИММЕТРИЧНОЙ Т. А, ОТНОСИТЕЛЬНО
30 ' ПРЯМОЙ, ПРОХОДЯЩЕЙ ЧЕРЕЗ Т. М И N
40 INPUT XM, YM, XN, YN, XA, YA
50 IF XM=XN THEN 130
60 IF YM=YN THEN 160
70 LET K=(YN-YM)/(XN-XM)
80 LET X=(K/(K^2+1))*(YA-YM+K*XM+XA/K)
90 LET Y=K*(X-XM)+YM
100 ' ВЫВОД ИСКОМЫХ КООРДИНАТ Т. А1
110 PRINT "X1="; 2*X-XA; "Y1="; 2*Y-YA
120 END
130 PRINT " ПРЯМАЯ MN - ВЕРТИКАЛЬНА "
140 PRINT "X1="; 2*XM-XA; "Y1="; YA
150 END
160 PRINT " ПРЯМАЯ MN - ГОРИЗОНТАЛЬНА "
170 PRINT "X1="; XA; "Y1="; 2*YM-YA
180 END

```

В этой программе отсутствует графическая часть. Это позволяет рассматривать предлагаемую программу как подпрограмму для решения различных задач. Решим одну из таких задач.

Задача. На бильярдном столе, имеющем треугольную форму, установлен шар (см. рис. 31). Шар расположен в точке A_0 . В какую точку стороны A_1A_2 следует направить шар, чтобы он, последовательно отразившись от всех трех сторон «бильярда», вновь попал в точку своего старта? (Удар шара о стенку будем считать идеальным ударом, и угол «падения» шара равен углу «отражения».)

Геометрическое решение задачи очень красиво и основано на умении строить точки, симметричные друг другу относительно прямой.

Разъясним это решение, рисунок 32 поможет понять детали. Сначала строим точку P_1 , симметричную точке A_0 относительно

стороны A_1A_3 (эту сторону мы обозначили цифрой 3). Затем строим точку P_2 , симметричную точке P_1 уже относительно стороны (2), а в завершение строим точку P_3 , симметричную точке P_2 относительно стороны A_1A_2 , то есть стороны (1).

После этого через точки A_0 и P_3 проводим прямую, которая пересечет сторону A_1A_2 в точке M_1 (может и не пересечь — в этом случае задача решений не имеет). Далее через точки M_1 и P_2 проводим прямую, которая пересечет сторону A_2A_3 в точке M_2 , затем находим точку M_3 — точку пересечения стороны треугольника A_1A_3 и прямой M_2P_1 . Точкой прицеливания является точка M_1 ; после удара об эту точку шар пройдет через точки M_2 , M_3 и через исходную точку A_0 .

Описанный геометрический алгоритм можно положить в основу программы, которую условно назовем «Шар на бильярдном столе». Конечно, такая программа должна быть применимой для управления движением шара по бильярду не только с тремя сторонами, но и с числом сторон бильярда более трех, например 7 или « n ». Возможный вариант программы такой:

```

10 'ШАР НА БИЛИАРДЕ
20 INPUT "УКАЖИТЕ К-ВО СТОРОН БИЛИАРДА L";L
30 DIM X(L),Y(L),HX(L+1),HY(L+1)
40 FOR I=1 TO L
50 PRINT I;"-Я ВЕРШИНА ":INPUT X(I),Y(I)
60 NEXT I
70 INPUT "ВВОД КООРДИНАТ ШАРА";X,Y
80 LET K3=X:LET K4=Y
90 SCREEN 3:COLOR 1,15,3:CLS
100 LINE (X(1),Y(1))-(X(L),Y(L)),7
110 FOR I=1 TO L-1
120 LINE (X(I),Y(I))-(X(I+1),Y(I+1)),7
130 NEXT I
140 CIRCLE (X,Y),3,7
150 LET M=X:LET N=Y
160 LET HX(L+1)=X:LET HY(L+1)=Y
170 FOR I=L TO 1 STEP -1
180 LET J=I:IF J=L THEN J=0
190 LET J=J+1
200 'ОПРЕДЕЛЕНИЕ КООРДИНАТ ВСПОМОГАТЕЛЬНЫХ ТОЧЕК
210 LET A=X(J)-X(I):LET B=Y(J)-Y(I)
220 LET T=(A*(HX(I+1)-X(I))+B*(HY(I+1)-Y(I)))/(A^2+B^2)
230 LET HX(I)=2*(A*T+X(I))-HX(I+1):HY(I)=2*(B*T+Y(I))-HY(I+1)
240 NEXT I
250 'ОПРЕДЕЛЕНИЕ КООРД-Т ТОЧЕК УДАРА (XU,YU) О СТОРОНЫ
260 FOR I=1 TO L
270 IF I=L THEN J=1 ELSE J=I+1
280 LET A=X(J)-X(I):LET B=Y(J)-Y(I):LET C=M-HX(I):LET D=N-HY(I)
290 LET K1=((HX(I)*D-HY(I)*C)*A-(X(I)*B-Y(I)*A)*C)/(D*A-B*C)
300 IF A=0 THEN 320
310 LET P1=D:LET P2=M:LET P3=N:LET P4=C:GOTO 330
320 LET P1=B:LET P2=X(I):LET P3=Y(I):LET P4=A
330 LET K2=(P1*(K1-P2)+P3*P4)/P4
340 IF (K1-X(I))*(K1-X(J))+(K2-Y(I))*(K2-Y(J))<0 THEN 360
350 PRINT "РЕШЕНИЯ НЕТ":END
360 LINE (M,N)-(K1,K2),6
370 LET M=K1:LET N=K2
380 NEXT I
390 LINE (K3,K4)-(M,N),6
400 END

```

Дадим краткие пояснения к тексту программы.

В строках 20—70 билиард задается координатами вершин. Билиард может быть произвольным многоугольником: треугольный билиард, пятиугольный билиард или с каким-то иным числом сторон. Здесь же указывается точка, в которую устанавливается шар.

В строках 100—140 приведены операторы, обеспечивающие демонстрацию на экране формы билиарда и шара на билиарде в виде маленького кружочка.

В строках 200—240 приведена вычислительная часть программы, вычисляются координаты вспомогательных точек P1, P2, P3, P4.

В строках 250—350 вычисляются координаты точек удара шара о стороны билиарда и проверяется, лежат ли эти точки в пределах той стороны билиарда, о которую шар должен удариться. Если рассчитанная точка удара о следующую сторону лежит за пределами стенки билиарда, то ЭВМ выдает сообщение «РЕШЕНИЙ НЕТ» и останавливается.

В строках 360—390 на экран выводится искомая траектория шара.

На рисунке 33 приведен пример движения шара по треугольному билиарду.

Несколько слов о снятии вручную координат точки, размещенной на экране. Программист с помощью клавиатуры ведет по экрану вспомогательную точку, и когда она совмещается с точкой, координаты которой нужно найти, он вызывает координаты вспомогательной точки. Примерный вариант программы приводится ниже.

```
5 REM СНЯТИЕ КООРДИНАТ ТОЧКИ ВРУЧНУЮ
10 COLOR 1,15,8:SCREEN 2
20 SPRITE$(0)="_":PSET(100,100),1
30 X=0:Y=0
40 T=STICK(0):D=STRIG(0):IF D=-1 GOTO 110
50 PUT SPRITE 1,(X,Y),8,0
60 IF T=1 THEN Y=Y-1
70 IF T=3 THEN X=X+1
80 IF T=5 THEN Y=Y+1
90 IF T=7 THEN X=X-1
100 GOTO 40
110 CLOSE:OPEN"GRP:"AS#1:PRINT #1,X;Y
120 FOR I=1 TO 15:GOTO 40
```

Такое использование ЭВМ может показаться искусственным, однако при работе в диалоге с ЭВМ иногда такие прикидки могут быть полезны. Представьте себе, что прямая и эллипс, лежащие в одной плоскости, пересекаются, и точки их пересечения видны на экране дисплея — в этом случае можно координаты этих точек найти вручную.

Продолжим обсуждение вопросов взаимодействия с ЭВМ через экран. Ниже приводится программа, имеющая необычный диало-

говый характер. Машина используется как циркуль и линейка. Для примера рассматривается задача построения биссектрисы угла.

```

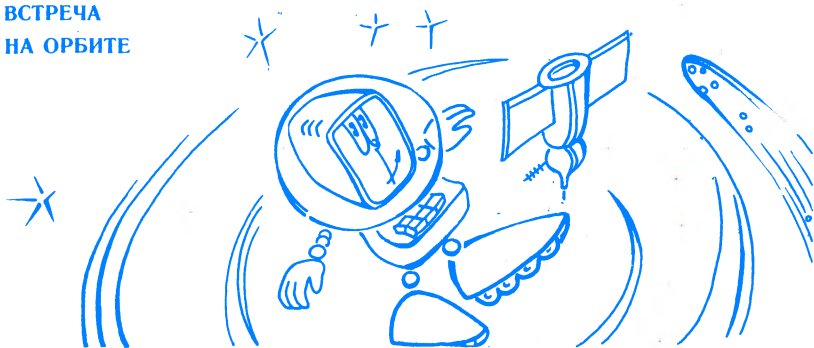
10 REM ПОСТРОЕНИЕ БИСSEKTPPCЫ
20 CLS:SCREEN 2:SPRITES(0)="_":X=10:Y=10
30 CLS:T=0:LINE(10,100)-(200,10),1
40 LINE(10,100)-(200,180),1
50 CIRCLE(10,100),70,2,3.14*372,3,1472
60 PUT SPRITE 0,(X,Y),H,0
70 X$=INKEY$:D=STICK(0):M=SRIG(0):IF X$="+" GOTO 30
80 IF M=-1 GOTO 140
90 IF D=1 THEN Y=Y-1
100 IF D=3 THEN X=X+1
110 IF D=5 THEN Y=Y+1
120 IF D=7 THEN X=X-1
130 GOTO 60
140 CLOSE:OPEN"GRP:"AS#1:PRINT #1,X,Y
150 IF T=2 THEN LINE(X,Y)-(10,100),8:GOTO 180
160 IF T=1 THEN CIRCLE(X,Y),70,1,(3/2)*3.14,3.14/3:T=T+1:GOTO 180
170 CIRCLE(X,Y),70,1,(5/3)*3.14,3.14/3:T=1
180 GOTO 60

```

Несколько рисунков разъясняют все этапы работы. На рисунке 34 показано, что, выполняя операторы 30, 40 и 50, ЭВМ построила угол и провела дугу вспомогательной окружности.

После этого она приостановила работу и стала ждать до тех пор, пока человек с помощью четырех клавиш, управляющих движением курсора, не переместил рабочую «красную» точку из ее начального положения в точку (10;10) в точку «1». После этого ЭВМ сама (оператор 160) проводит окружность с центром в точке «1» тем же радиусом, что и ранее. Вновь пауза в работе ЭВМ до тех пор, пока человек не переместит «красную» точку в точку «2». ЭВМ после этого проводит дугу из точки «2» и получает точку пересечения двух дуг — точку «3». Вновь возникает пауза в работе ЭВМ — в это время человек перемещает «красную» точку в точку «3» и тем самым «сообщает координаты» точки «3» ЭВМ, которая может теперь провести искомую биссектрису (оператор 150) (рис. 35).

На этом примере было показано, как моделируются геометрические инструменты, как с помощью необычных циркуля и линейки можно осуществлять самые разнообразные построения в цвете. Рассмотрим очень простой пример — такое применение ЭВМ может быть обогащено путем расширения запаса инструментов, которыми может человек вооружить ЭВМ.



Представим ситуацию: Наблюдается редкое явление — в пределах околоземного пространства вошло небольшое небесное тело (метеор) и движется в сторону Земли. В первые минуты полета были отмечены несколько точек, через которые прошло тело, и специалисты установили, что оно движется по параболической орбите. При этом возникло опасение, что тело может пройти в опасной близости от большой научно-исследовательской станции, находящейся в космосе вблизи Земли (рис. 36). Возникла задача уточнения траектории полета метеора и вычисления наибольшего сближения его с космической станцией.

Была построена математическая модель ситуации, и с помощью ЭВМ специалисты нашли ответ на оба вопроса. Они определили все коэффициенты уравнения параболы и вычислили, в какой точке пространства M метеор окажется ближе всего к станции.

Задачу в такой постановке, когда следует учесть и скорость движения станции, и характер движения метеора, и то, что они движутся в различных плоскостях, решить непросто. Ниже будет рассмотрена задача более простая, она аналогична упомянутой выше. При решении этой задачи будут подчеркнуты особенности решения таких задач.

Задача. Пусть известно, что в прямоугольной системе координат xOy по параболической орбите вида $y = Ax^2 + Bx + C$ движется точка M (коэффициенты A , B и C пока неизвестны). В этой же плоскости находится точка $S(x_0; y_0)$. В результате наблюдений за точкой M обнаружено, что она прошла через три точки $T_1(x_1; y_1)$, $T_2(x_2; y_2)$ и $T_3(x_3; y_3)$. Найти значения всех коэффициентов A , B и C и записать уравнение траектории движения точки M , а также вычислить, в какой точке $Q(x_4; y_4)$ произойдет наибольшее сближение точек M и S .

Одно из решений может быть таким. Так как при $x = x_1$ значение $y = y_1$, то можно записать $Ax_1^2 + Bx_1 + C = y_1$. Учитывая, что $y(x_2) = y_2$ и $y(x_3) = y_3$, можно записать аналогичные уравнения

$$Ax_2^2 + Bx_2 + C = y_2,$$

$$Ax_3^2 + Bx_3 + C = y_3.$$

Получена система трех линейных уравнений с тремя неизвестными A , B и C .

$$\begin{cases} Ax_1^2 + Bx_1 + C = y_1, \\ Ax_2^2 + Bx_2 + C = y_2, \\ Ax_3^2 + Bx_3 + C = y_3. \end{cases} \quad (1)$$

Систему трех уравнений с тремя неизвестными можно решать различными способами, например, применяя формулы Крамера.

Крамер Габриэль (1704—1752) — швейцарский математик. Профессор математики и философии. Основные работы относятся к высшей алгебре и аналитической геометрии. В алгебре установил и опубликовал (1750 г.) правила решения систем n линейных уравнений с n неизвестными.

В этом случае предварительно необходимо вычислить четыре определителя третьего порядка:

$$\Delta = \begin{vmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{vmatrix}; \Delta_A = \begin{vmatrix} y_1 & x_1 & 1 \\ y_2 & x_2 & 1 \\ y_3 & x_3 & 1 \end{vmatrix}; \Delta_B = \begin{vmatrix} x_1^2 & y_1 & 1 \\ x_2^2 & y_2 & 1 \\ x_3^2 & y_3 & 1 \end{vmatrix}; \Delta_C = \begin{vmatrix} x_1^2 & x_1 & y_1 \\ x_2^2 & x_2 & y_2 \\ x_3^2 & x_3 & y_3 \end{vmatrix}.$$

Неизвестные коэффициенты вычисляются по формулам:

$$A = \frac{\Delta_A}{\Delta}; B = \frac{\Delta_B}{\Delta} \text{ и } C = \frac{\Delta_C}{\Delta}.$$

Существует простое правило вычисления определителей третьего порядка. Пусть дан определитель:

$$\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}.$$

Составим вспомогательную таблицу, приписав справа два первых столбца. Составим произведения из трех элементов так, как указывают стрелки (рис. 37). Произведения у стрелок, у которых написан «+», возьмем с тем же знаком, а произведения, у стрелок которых стоит «—», возьмем с противоположным знаком и получим:

$$\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = a_1 b_2 c_3 + b_1 c_2 a_3 + c_1 a_2 b_3 - a_3 b_2 c_1 - b_3 c_2 a_1 - c_3 a_2 b_1.$$

Для вычисления коэффициента A в нашей задаче придется вычислить значение дроби:

$$A = \frac{y_1 x_2 + x_1 y_3 + y_2 x_3 - y_3 x_2 - x_3 y_1 - y_2 x_1}{x_1^2 x_2 + x_1 x_3^2 + x_3 x_2^2 - x_3^2 x_2 - x_3 x_1^2 - x_2^2 x_1}.$$

Аналогично составляются выражения для вычисления коэффициентов B и C :

$$B = \frac{x_1^2 y_2 + y_1 x_2^2 + y_3 x_2^2 - x_3^2 y_2 - y_3 x_1^2 - x_2^2 y_1}{x_1^2 x_2 + x_1 x_2^2 + x_3 x_2^2 - x_3^2 x_2 - x_3 x_1^2 - x_2^2 x_1};$$

$$C = \frac{x_1^2 x_2 y_3 + x_1 y_2 x_3^2 + y_1 x_2^2 x_3 - x_3^2 x_2 y_1 - x_3 y_2 x_1^2 - y_3 x_2^2 x_1}{x_1^2 x_2 + x_1 x_2^2 + x_3 x_2^2 - x_3^2 x_2 - x_3 x_1^2 - x_2^2 x_1}.$$

Существует способ отыскания коэффициентов без использования системы трех линейных уравнений и без применения формул Крамера.

Лагранж Жозеф Луи (1736—1813) — французский математик и механик. Сочинения Лагранжа по математике, астрономии и физике составляют 14 томов. В математическом анализе Лагранж дал формулу остаточного члена ряда Тейлора, формулу конечных приращений и интерполяционную формулу.

Этот способ основан на использовании интерполяционной формулы Лагранжа, имеющей для данного случая вид:

$$\begin{aligned} f(x) = y = & y_1 \cdot \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} + y_2 \cdot \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} + \\ & + y_3 \cdot \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}. \end{aligned} \quad (2)$$

Легко проверить, что при $x=x_1$ второе и третье слагаемое формулы (2) будут равны нулю и $f(x_1)=y_1$, а это означает, что кривая, задаваемая уравнением (2), проходит через точку $T_1(x_1; y_1)$. Аналогично при $x=x_2$ $f(x_2)=y_2$ и при $x=x_3$ $f(x_3)=y_3$ парабола проходит и через точки $T_2(x_2; y_2)$ и $T_3(x_3; y_3)$.

Может возникнуть вопрос: действительно ли уравнение (2) является уравнением параболы? Убедимся, что это действительно так. Преобразуем правую часть, для чего сначала введем обозначения:

$$m = \frac{y_1}{(x_1-x_2)(x_1-x_3)}; \quad n = \frac{y_2}{(x_2-x_1)(x_2-x_3)};$$

$$k = \frac{y_3}{(x_3-x_1)(x_3-x_2)} \quad (3)$$

и перепишем уравнение (2) в виде:

$$y = m(x-x_2)(x-x_3) + n(x-x_1)(x-x_3) + k(x-x_1)(x-x_2).$$

Если теперь в правой части раскрыть все скобки и полученный многочлен записать, расположив его члены по убывающим степеням переменной x , то обнаружится, что этот многочлен есть квадратный трехчлен вида Ax^2+Bx+C , то есть является уравнением параболы:

$$y = (m+n+k)x^2 - (m(x_3+x_2) + n(x_1+x_3) + k(x_1+x_2))x + (mx_2x_3 + nx_1x_3 + kx_1x_2).$$

Искомые коэффициенты:

$$\begin{aligned} A = m+n+k; \quad B = - (m(x_3+x_2) + n(x_3+x_1) + k(x_1+x_2)); \\ C = mx_2x_3 + nx_1x_3 + kx_1x_2. \end{aligned} \quad (4)$$

Прежде чем продолжить решение задачи, составим программу отыскания коэффициентов квадратной параболы, зная координаты трех ее точек. В основу программы берем формулы (3), (4), предварительно выяснив, что вычисления по этим формулам включают значительно меньше арифметических операций, чем вычисления по формулам Крамера.

Программа может быть такой:

```

5  REM ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТОВ КВАДРАТНОЙ ПАРАБОЛЫ
10 INPUT -X1, Y1, X2, Y2, X3, Y3
15 LET M=Y1/((X1-X2)*(X1-X3)): LET N=Y2/((X2-X1)*(X2-X3))
16 LET K=Y3/((X3-X1)*(X3-X2))
20 LET A=M+N+K: LET B=-(M*(X3+X2)+N*(X3+X1)+K*(X2+X1))
21 LET C=M*X2*X3+N*X1*X3+K*X1*X2
25 PRINT A;B;C
30 END

```

После того как неизвестные коэффициенты уравнения траектории полета точки найдены, можно приступить к решению второй части задачи: выяснить, не пройдет ли точка M в опасной близости от точки S . Воспользуемся известной формулой для вычисления расстояния между двумя точками, например, точками $A(x_A; y_A)$ и $B(x_B; y_B)$:

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}.$$

В этой формуле координаты точек A и B постоянны. В нашей задаче постоянные координаты имеет точка $S(x_0; y_0)$, а точка M движется и ее координаты x — абсцисса и $y = Ax^2 + Bx + C$ — ордината меняются.

Расположение движущейся точки M и неподвижной точки S схематически можно изобразить так, как показано на рисунке 38.

По мере перемещения точки M расстояние SM меняется, и это расстояние есть функция от x , задаваемая формулой:

$$SM = R(x) = \sqrt{(x - x_0)^2 + ((Ax^2 + Bx + C) - y_0)^2}. \quad (5)$$

Наибольшее сближение точек S и M есть минимум функции $R(x)$.

Возникла новая задача — найти минимум функции (5). Школьный учебник советует начать отыскание минимума с дифференцирования функции $R(x)$ и получения первой ее производной — $R'(x)$. Дифференцируем и имеем:

$$R'(x) = \frac{2(x - x_0) + 2(Ax^2 + Bx + C - y_0)(2Ax + B)}{2\sqrt{(x - x_0)^2 + ((Ax^2 + Bx + C) - y_0)^2}}.$$

В числителе открываем все скобки. Полученный многочлен располагаем по убывающим степеням переменной x и полученное выражение приравняем к нулю:

$$2A^2x^3 + 3ABx^2 + (B^2 + 2AC - 2Ay_0 + 1)x + (BC - By_0 - x_0) = 0,$$

или

$$A_1x^3 + B_1x^2 + C_1x + D_1 = 0,$$

где

$$A_1 = 2A^2; B_1 = 3AB; C_1 = B^2 - 2AC - 2Ay_0 + 1; \\ D_1 = BC - By_0 - x_0.$$

Получено общее уравнение третьей степени, необходимо найти все его действительные корни. Это можно сделать различными способами: вычислить по формуле Кардано, решить методом половинного деления отрезка, содержащего корень, или как-то иначе.

Кардано Джироламо (1501—1576) — итальянский математик, механик и врач. В математике с именем Кардано связывают формулу для решения кубического уравнения, которую он позаимствовал у Н. Тарталья. Эта формула была опубликована в его книге «Великое искусство, или о правилах алгебры» (1545).

Пусть мы выбрали вычисление корней кубического уравнения по формуле Кардано. Отметим, что программа для отыскания корней полного кубического уравнения по формуле Кардано имеет большое значение. Составление программы основано на ряде соображений и должно учитывать немало условий. Разработку программы осуществим отдельно, отвлекаясь от конкретной задачи, которая подвела нас к потребности решить кубическое уравнение.

В следующем параграфе рассказывается, как следует решать полные кубические уравнения с использованием формулы Кардано. Там же приводится программа и необходимые разъяснения. Если мы воспользуемся этой программой, то точки, в которых возможно наибольшее сближение, будут найдены — это либо одна точка, либо две, либо три.

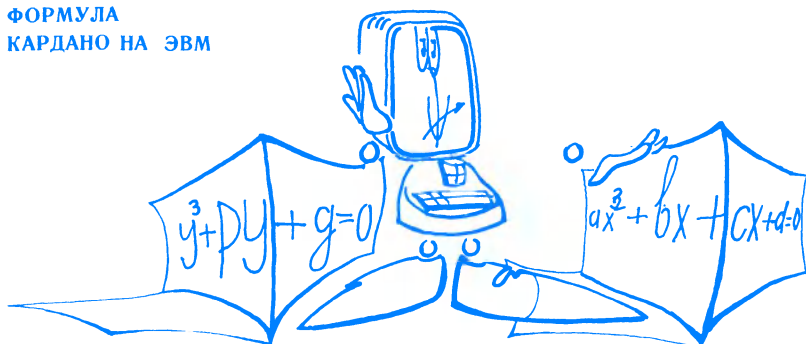
Нам остается вычислить значение функции $R(x)$ в найденных критических точках и сравнить между собой. Наименьшее значение $R(x)$ и даст наибольшее значение сближения. Зная абсциссу точки наибольшего сближения, можно вычислить и ее ординату: $y_4 = Ax_4^2 + Bx_4 + C$.

После этого задачу о встрече на орбите можно считать решенной — координаты точки наибольшего сближения движущейся точки M и неподвижной точки S найдены, вычислена и величина наибольшего сближения.

Читателю ясно, что общий подход к решению задачи останется прежним, если траектория движения небесного тела будет не параболой, а, например, гиперболой.

Вновь по нескольким точкам потребуется восстановить уравнение, на этот раз уравнение гиперболы, и вычислить минимум расстояния между пролетающим небесным телом и космической станцией.

Если станция находится в плоскости гиперболы, а уравнение гиперболы задано формулой $y = \frac{A_1x + B_1}{A_2x + B_2}$, то задачу может попробовать решить читатель.



Пусть дано полное кубическое уравнение: $ax^3 + bx^2 + cx + d = 0$. Применив подстановку $x = y - b/3a$, получим уравнение кубического вида: $y^3 + py + q = 0$, где

$$p = \frac{3ac - b^2}{3a^2} \quad \text{и} \quad q = \frac{3b^3 - 9abc + 27a^2d}{27a^3}.$$

Введем переменную $D = \left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^2$, теперь, если $D > 0$, то исходное уравнение имеет один действительный корень и два мнимых:

$$y_1 = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}.$$

Введем обозначения для выписанных выше кубических корней:

$$u = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}, \quad v = \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}.$$

И тогда действительный корень $y_1 = u + v$, а два мнимых имеют вид:

$$y_2 = -\frac{1}{2}(u + v) + i\frac{\sqrt{3}}{2}(u - v); \quad y_3 = -\frac{1}{2}(u + v) - i\frac{\sqrt{3}}{2}(u - v).$$

Если же $D = 0$, то все корни кубического уравнения действительные и два из них равны между собой:

$$y_1 = u + v = 2\sqrt[3]{-\frac{q}{2}}, \quad y_2 = y_3 = -\frac{1}{2}(u + v) = -\sqrt[3]{-\frac{q}{2}} = \sqrt[3]{\frac{q}{2}}.$$

Если же $D < 0$, то уравнение имеет три различных действительных корня:

$$\begin{aligned} y_1 &= 2\sqrt[3]{-\frac{p}{3}} \cdot \cos \frac{\varphi}{3}, \\ y_2 &= 2\sqrt[3]{-\frac{p}{3}} \cdot \cos \frac{\varphi + 2\pi}{3}, \\ y_3 &= 2\sqrt[3]{-\frac{p}{3}} \cdot \cos \frac{\varphi + 4\pi}{3}. \end{aligned}$$

Угол φ находится из условия $\cos y = \frac{q}{2\sqrt{\frac{p^3}{27}}}$. Найдя один или

несколько корней уравнения $y^3 + py + q = 0$, вычисляем корни исходного уравнения, для этого применим подстановку $x = y - \frac{b}{3a}$.

Прежде чем приступить к составлению программы на основе приведенных выше кратких теоретических соображений, следует рассмотреть отдельно задачу отыскания угла φ по известному значению функции косинус. Это значит решить уравнение вида: $\cos \varphi = a$.

Напомним, что в нашем распоряжении язык Бейсик, и вот оказывается, что в ряде версий языка среди встроенных функций нет функции $\arccos x$. Имеется только функция $\arctg x$, а функций, обратных функции синус и косинус, нет. Как же решить

уравнение $\cos \varphi = \frac{-q}{2\sqrt{-\frac{p^3}{27}}}$?

Рассмотрим два пути. Сначала выразим $\tg \varphi$ через $\cos \varphi$, то есть запишем:

$$\sin \varphi = \sqrt{1 - \cos^2 \varphi} \text{ и } \tg \varphi = \frac{\sqrt{1 - \cos^2 \varphi}}{\cos \varphi}.$$

И так как $\cos \varphi$ нам известен, то мы сможем вычислить и $\tg \varphi$. Для этого достаточно в программе использовать операторы:

```
LET C=-Q/(2*SQR(-P^3/27))
LET T QF=SQR(1-C^2)/C
LET F=ATN(TQF)
```

После того как будет вычислено значение $\varphi = F$, можно вычислить значения функции косинус в точках $\frac{\varphi + 2\pi}{3}$ и $\frac{\varphi + 4\pi}{3}$. Для этого используются операторы:

```
LET C1=COS(F/3)
LET C2=COS((F+6.28318)/3)
LET C3=COS((F+12.56636)/3)
```

После этого вычисляются y_1 , y_2 и y_3 , а затем и три корня исходного уравнения x_1 , x_2 и x_3 .

Второй путь вычисления числа φ можно образно назвать «машинным». Предлагается поручить ЭВМ «в цикле» вести вычисления значений функции косинус от $\varphi = 0$ до $\varphi = \pi$, шаг принять малым, и эти вычисления должны сопровождаться проверкой. Машина должна проверять, не равно ли вычисленное

значение $\cos(\varphi_i)$ числу $\frac{-q}{2\sqrt{-\frac{p^3}{27}}}$. Если вычисленное значение и

это число отличаются достаточно мало, то соответствующее значение аргумента φ_i и следует считать искомым углом φ .

Фрагмент программы в этом случае может быть таким:

```
10 LET C1=-Q/(2*SQR(-P^3/27)):LET E=.001
15 FOR F=0 TO 3.14159 STEP .01
20 LET C=COS(F)
25 IF ABS(C1-C) < E THEN 35
30 NEXT F
35 PRINT F
40 END
```

Таким образом, с помощью машинного подбора значения F мы вычислили искомое значение угла φ .

Рассказанного достаточно для написания программы решения полного кубического уравнения $ax^3+bx^2+cx+d=0$ на ЭВМ с помощью формул Кардано.

Программа может быть такой, как приведенная ниже. Конечно, при окончательной отладке текста программы были применены некоторые «маленькие хитрости».

```
5 REM РЕШЕНИЕ ПОЛНЫХ КУБИЧЕСКИХ УРАВНЕНИЙ
10 DEF FNF(X)=SGN(X)*ABS(X)^(1/3)
15 INPUT "ВВЕДИТЕ КОЭФФИЦИЕНТЫ УРАВНЕНИЯ";A,B,C,D
20 LET P=(3*A*C-B^2)/(3*A^2)
25 LET Q=(2*B^3-9*A*B*C+27*A^2*D)/(27*A^3)
30 LET D=(P/3)^3+(Q/2)^2
35 IF D<0 THEN 90
40 LET U=-Q/2+SQR(D):LET V=-Q/2-SQR(D)
45 IF D>0 THEN 70
50 REM ПРИ D=0 УРАВНЕНИЕ ИМЕЕТ НЕ БОЛЕЕ ДВУХ КОРНЕЙ X1,X2
55 LET X1=U+V-B/(3*A)
60 LET X2=-(U+V)/2-B/(3*A)
65 PRINT X1;X2:END
70 REM ПРИ D>0 УРАВНЕНИЕ ИМЕЕТ ОДИН ДЕЙСТВИТЕЛЬНЫЙ КОРЕНЬ
75 LET U=FNF(U)
80 LET V=FNF(V)
85 LET X=U+V-B/(3*A):PRINT X:END
90 REM ПРИ D<0 УРАВНЕНИЕ ИМЕЕТ ТРИ КОРНЯ X1,X2,X3
95 LET C=-Q/(2*SQR(-P^3/27))
100 LET F=-ATN(C/SQR(-C*C+1))+1.5708
105 LET R=SQR(-P/3)
110 LET X1=2*R*COS(F/3)-B/(3*A)
115 LET X2=2*R*COS((F+6.28318)/3)-B/(3*A)
120 LET X3=2*R*COS((F+12.5663)/3)-B/(3*A)
125 PRINT X1;X2;X3
130 END
```

На первый взгляд программа кажется большой и громоздкой, но ведь и задача непростая.

Обращаем внимание школьников, что необходимость решать полные и неполные кубические уравнения встречается при решении многих задач и разработанная программа может включаться в программы решения этих задач как подпрограмма.

В качестве примера читатель может попробовать свои силы в разработке программы для решения неполного кубического уравнения вида: $ax^3 + cx = 0$.

При составлении программы он должен учесть несколько обстоятельств: рекомендуем, например, обратить внимание на знаки коэффициентов a и c . Возможно, что при некоторых значениях a и c возникнут особенности в решении уравнения.

В начале программы возможно следует включить небольшое исследование коэффициентов и лишь после этого давать основные вычисления.

Аналогичная задача — составление программы для решения неполного кубического уравнения вида: $ax^3 + bx^2 = 0$. В этом случае проблем меньше, так как это уравнение легче исследовать.

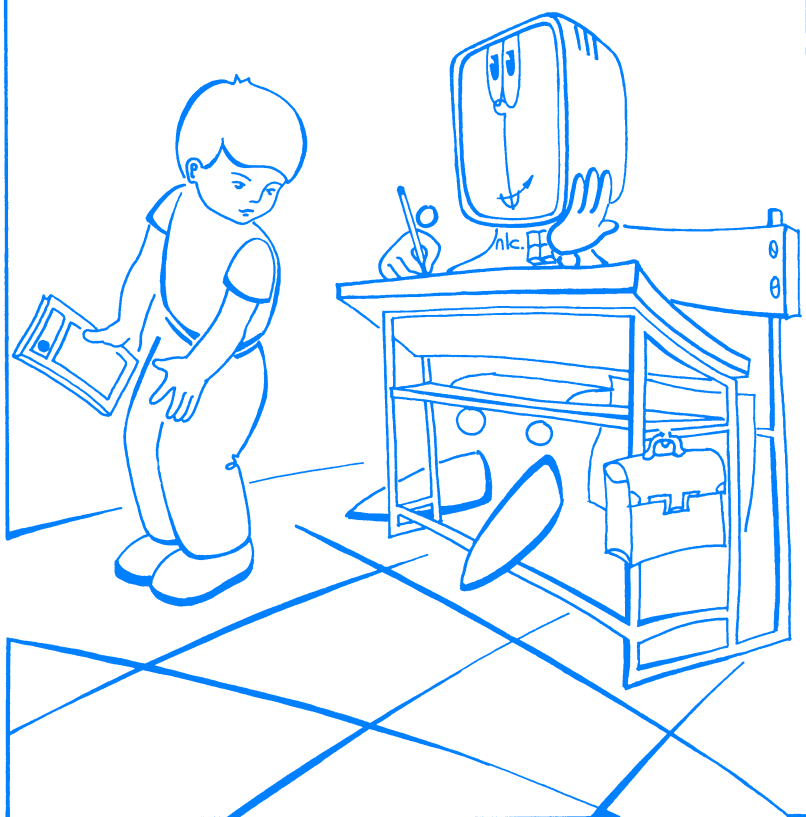
Можно предложить читателю составить единую программу для решения этих двух уравнений.

Наконец, читатель может попробовать свои силы в решении неполного кубического уравнения вида: $ax^3 + d = 0$. В этом случае ему придется познакомиться с тем, как с помощью ЭВМ извлекают кубические корни.

ГЛАВА

III.

ЭВМ УЧИТСЯ
ЛОГИКЕ



ЭВМ ВЫПОЛНЯЕТ ЛОГИЧЕСКИЕ ОПЕРАЦИИ. ПО ЕДИНОМУ АЛГОРИТМУ. АРИФМЕТИКА СЛУЖИТ ЛОГИКЕ.

*Пишите программы просто —
не делайте их слишком хитро-
умными.*

Из фольклора программистов

Логические задачи — твердый орешек. Чтобы ЭВМ могла помочь в их решении, ее научили выполнять операции алгебры высказываний: логическое сложение, умножение и отрицание.

В центре читательского внимания особенности подготовки решения логических задач на ЭВМ. В этой части читатель приглашается к ознакомлению с основными логическими операциями, выполняемыми над высказываниями. Высказывания выражают логическую сущность задачи.

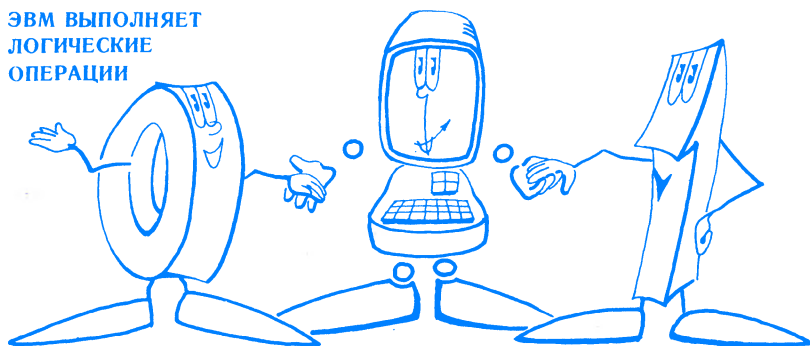
Мир формул алгебры высказываний — мир необычайно интересный. Начиная изучать школьную алгебру, учащиеся овладевают умением выполнять тождественные преобразования: учатся выносить множители за скобки, осуществлять разложение на множители, пользоваться формулами сокращенного умножения и многим другим. Чрезвычайно важно уметь составлять формулы сложных высказываний, понимать, что значит тождественно преобразовать формулу алгебры высказываний. Об этом и идет речь.

Неожиданным для школьника является тот факт, что огромное количество логических задач решается по единому алгоритму. Это означает, что можно разобраться в том, как ЭВМ составляет расписание, как машина распределяет продукты, учитывая множество различных, подчас противоречивых условий. Задача о спортивном состязании как раз и позволяет подчеркнуть все этапы подготовки к решению задачи на ЭВМ, увидеть суть единого алгоритма и оценить трудности в решении таких задач, даже на ЭВМ.

Работа читателю-школьнику предстоит серьезная, но дело того стоит. Разобраться в том, как логика действует в сотрудничестве с ЭВМ — необходимо.

Подчеркнем, что при решении сложных задач умение создавать из простых условий передачи управления составные условия, формулируемые с помощью операций алгебры высказываний, позволяет делать программы более простыми и эффективными. Это значит, что алгебра высказываний необходима для программиста, потребность в ней может возникнуть при решении не только логической задачи, а и задачи вычислительной, графической или задачи символьной обработки.

**ЭВМ ВЫПОЛНЯЕТ
ЛОГИЧЕСКИЕ
ОПЕРАЦИИ**



ЭВМ может выполнять логические операции, и это дает возможность поручать ей решение сложных логических задач. В развитых версиях языка Бейсик в числе основных функций имеются и логические функции. Напомним, что эти функции задаются таблицами (табл. 5—7).

Таблица 5

Логическое умножение

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 6

Логическое сложение

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Таблица 7

Логическое отрицание

A	\bar{A}
0	1
1	0

Умение выполнять каждую отдельную операцию позволяет ЭВМ выполнять вычисления весьма сложных логических функций. Рассмотрим пример, в котором ЭВМ вычислит и напечатает все значения нижеприведенной логической функции F от трех логических переменных: A , B и C : $F(A, B, C) = A + B \cdot C$.

В записи формулы использованы обозначения: знак «+» — для обозначения логического сложения; знак « \cdot » — для обозначения логического умножения; знак «черта» — для обозначения логического отрицания. В тексте программы эти символы будут заменены на обозначения логических операций, принятых в языках программирования.

Напомним, что каждая логическая переменная принимает только два значения: 0 и 1, и вычисленное выражение может быть также только 0 или 1. Решить поставленную задачу — это значит заполнить таблицу 8.

Таблица 8

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

На рисунке 39 приведена схема алгоритма для решения этой очень простой задачи, рядом выписан текст программы и итоговая таблица вычислений.

Приведем программу решения этой же задачи, в которой будут использованы операторы цикла.

```

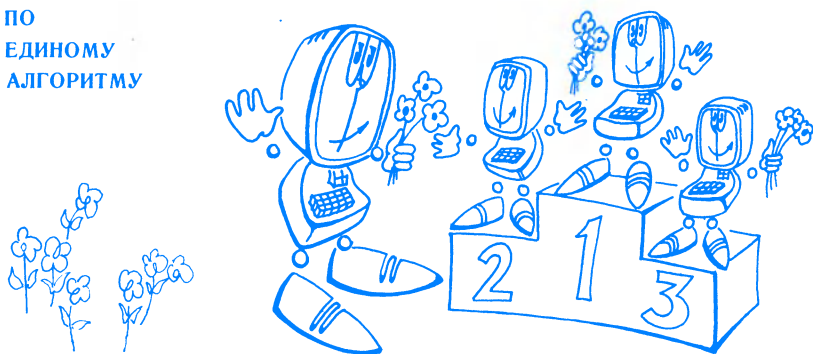
5 REM ТАБЛИЦА ЗНАЧЕНИЙ ЛОГ. ФУНКЦИИ
10 FOR A=0 TO 1
15 FOR B=0 TO 1
20 FOR C=0 TO 1
25 LET F=F(A,B,C)
30 PRINT A;B;C;F
35 NEXT C
40 NEXT B
45 NEXT A
50 END

```

Заметим, что вычисление значений логических функций является частью более сложных логических процедур. Укажем для примера, что решение многих логических задач, например, занимательных задач, основано на умении вычислять все значения логической функции, выражающей логическую сущность задачи.

Приведенная выше программа «Таблица значений логической функции» часто входит в более сложные программы. Конечно, число переменных, входящих в выражение, может быть различным. Однако алгоритм остается единым и для различных выражений, и для различного числа переменных.

Сегодня ЭВМ могут работать и в более сложной области математической логики — могут действовать в рамках логики предикатов. Для взаимодействия человека и ЭВМ в этой области созданы специальные средства, среди которых упомянем о языке программирования Пролог. Сегодня язык Пролог можно применять на персональных ЭВМ, это значит, что решение сложных логических задач становится делом обыденным.



Вычислительные задачи — это только часть того, что можно поручить ЭВМ. Схема применения ЭВМ при решении логических задач, приводимая ниже, поможет увидеть возможности в решении неарифметических задач.

Задача. Шесть спортсменов — Адамов, Белов, Ветров, Глебов, Дронов и Ершов — в проходившем соревновании заняли первые шесть мест, причем ни одно место не было разделено между ними. О том, кто какое место занял, были получены такие высказывания:

1. «Кажется, первым был Адамов, а вторым — Дронов».
2. «Нет, на первом месте был Ершов, а на втором — Глебов».
3. «Вот так болельщики! Ведь Глебов был на третьем месте, а Белов — на четвертом».
4. «И вовсе было не так: Белов был пятым, а Адамов — вторым».
5. «Вы все перепутали: пятым был Дронов, а перед ним — Ветров».

Известно, что в высказывании каждого болельщика одно утверждение истинное, а второе — ложное. Определить, какое место занял каждый из спортсменов.

Решение задачи начинается с введения буквенных обозначений всех высказываний, задающих условие задачи:

A1 — «Адамов был первым»,

D2 — «Дронов — второй»,

E1 — «Ершов был первым»,

G2 — «Глебов — второй»,

G3 — «Глебов — третий»,

B4 — «Белов — четвертый»,

B5 — «Белов — пятый»,

A2 — «Адамов — второй»,

D5 — «Дронов — пятый»,

B4 — «Ветров — четвертый».

Высказывания каждого болельщика о двух спортсменах можно задать формулами:

$$\begin{aligned}
 A1 \cdot \overline{D2} + \overline{A1} \cdot D2 &= 1, \\
 E1 \cdot \overline{Г2} + \overline{E1} \cdot Г2 &= 1, \\
 Г3 \cdot \overline{Б4} + \overline{Г3} \cdot Б4 &= 1, \\
 Б5 \cdot \overline{А2} + \overline{Б5} \cdot А2 &= 1, \\
 Д5 \cdot \overline{В4} + \overline{Д5} \cdot В4 &= 1.
 \end{aligned} \tag{1}$$

Помним, что в условии задачи сказано: в высказывании каждого болельщика одно высказывание истинно, а другое — ложно. Следует учесть и то, что ни одно место не было разделено участниками. Это условие можно задать формулами:

$$\begin{aligned}
 A1 \cdot E1 &= 0, & \overline{A1} + \overline{E1} &= 1, \\
 A2 \cdot D2 &= 0, & \overline{A2} + \overline{D2} &= 1, \\
 B4 \cdot B4 &= 0, & \overline{B4} + \overline{B4} &= 1, \\
 B5 \cdot D5 &= 0, & \overline{B5} + \overline{D5} &= 1.
 \end{aligned} \tag{2}$$

И наконец, следует учесть, что ни один из спортсменов не может занять два разных места. Это обстоятельство выражается формулами:

$$\begin{aligned}
 A1 \cdot A2 &= 0, & \overline{A1} + \overline{A2} &= 1, \\
 D2 \cdot D5 &= 0, & \overline{D2} + \overline{D5} &= 1, \\
 B4 \cdot B5 &= 0, & \overline{B4} + \overline{B5} &= 1, \\
 Г2 \cdot Г3 &= 0, & \overline{Г2} + \overline{Г3} &= 1.
 \end{aligned} \tag{3}$$

Обращаем внимание на то, что все логические условия записаны в виде истинных высказываний.

Решить поставленную задачу — это значит учесть все логические условия, задающие содержание и суть взаимосвязей в задаче. Решить логическую задачу — это узнать, при каких значениях переменных $A1, D2, E1, Г2, Б4, A2, Б5, Д5, Г3$ и $В4$ все условия выполняются одновременно. Решить данную задачу — это значит узнать, при каких значениях указанных выше переменных логическое произведение условий (1), (2) и (3) принимает значение 1, то есть истинно.

Итак, решение задачи свелось к задаче вычисления значений сложного логического выражения:

$$\begin{aligned}
 F &= (\overline{A1} \cdot D2 + A1 \cdot \overline{D2}) (\overline{E1} \cdot \overline{Г2} + E1 \cdot Г2) (\overline{Г3} \cdot B4 + Г3 \cdot \overline{Б4}) \\
 &(\overline{Б5} \cdot A2 + Б5 \cdot \overline{А2}) (\overline{Д5} \cdot \overline{В4} + Д5 \cdot В4) (\overline{A1} + \overline{E1}) (\overline{A2} + \overline{D2}) \\
 &(\overline{B4} + \overline{B4}) (\overline{B5} + \overline{D5}) (\overline{A1} + \overline{A2}) (\overline{D2} + \overline{D5}) (\overline{B4} + \overline{B5}) (\overline{Г2} + \overline{Г3}).
 \end{aligned} \tag{4}$$

Решить задачу можно по рассмотренной выше программе «Таблица значений логической функции». Работая по этой программе, ЭВМ выдаст таблицу, в которой будет $n = 2^{10} = 1024$ строки и 11 столбцов. В одной из строк значение F будет равно единице, и можно будет узнать, чему при этом равно значение каждой из десяти переменных.

Процесс решения задачи на ЭВМ можно упростить. Приводимое ниже изменение схемы алгоритма и соответствующее

изменение программы можно осуществить только в том случае, если мы убеждены, что задача имеет единственное решение. Если это так, то вычисление значения F следует вести только до тех пор, пока F не станет равным 1. Конечно, и на печать следует выдавать только ту строку таблицы, в которой $F=1$.

Изменения, которые следует ввести в схему алгоритма, приведены на рисунке 40.

В тексте программы должна появиться строка

IF F=1 THEN PRINT A1; D2; E1; G2; G3; B4; D5; A2; B5; B4; F.

В заключение приведем полный текст программы на языке Бейсик и результат решения задачи.

```

10 REM РЕШЕНИЕ ЛОГИЧЕСКОЙ ЗАДАЧИ
20 FOR A1=0 TO 1
30 FOR D2=0 TO 1
40 FOR E1=0 TO 1
50 FOR G2=0 TO 1
60 FOR G3=0 TO 1
70 FOR B4=0 TO 1
80 FOR B5=0 TO 1
90 FOR A2=0 TO 1
100 FOR D5=0 TO 1
110 FOR W4=0 TO 1
120 REM ВВОД ЛОГИЧЕСКОГО ВЫРАЖЕНИЯ,
130 REM ЗАДАЮЩЕГО УСЛОВИЯ ЗАДАЧИ
140 LET F1=(A1 AND NOT D2) OR (NOT A1 AND D2)
150 LET F2=(E1 AND NOT G2) OR (NOT E1 AND G2)
160 LET F3=(G3 AND NOT B4) OR (NOT G3 AND B4)
170 LET F4=(B5 AND NOT A2) OR (NOT B5 AND A2)
180 LET F5=(D5 AND NOT W4) OR (NOT D5 AND W4)
190 LET F6=(NOT A1 OR NOT E1) AND (NOT A2 OR NOT D2)
200 LET F7=(NOT B4 OR NOT W4) AND (NOT B5 OR NOT D5)
210 LET F8=(NOT B4 OR NOT B5) AND (NOT G2 OR NOT G3)
220 LET F9=(NOT D2 OR NOT D5) AND (NOT A1 OR NOT A2)
230 LET F10=F1 AND F2 AND F3 AND F4 AND F5
240 LET F11=F6 AND F7 AND F8 AND F9:LET F=F10 AND F11
250 IF F=1 THEN 370
260 NEXT W4
270 NEXT D5
280 NEXT A2
290 NEXT B5
300 NEXT B4
310 NEXT G3
320 NEXT G2
330 NEXT E1
340 NEXT D2
350 NEXT A1
360 PRINT "ЗАДАЧА РЕШЕНИЙ НЕ ИМЕЕТ":GOTO 210
370 PRINT "ВЫВОД РЕЗУЛЬТАТА"
380 IF A1=1 THEN PRINT "1-Е МЕСТО - АДАМОВ"
390 IF D2=1 THEN PRINT "2-Е МЕСТО - ДРОНОВ"
400 IF E1=1 THEN PRINT "1-Е МЕСТО - ЕРШОВ "
410 IF G2=1 THEN PRINT "2-Е МЕСТО - ГЛЕБОВ"
420 IF G3=1 THEN PRINT "3-Е МЕСТО - ГЛЕБОВ"
430 IF B4=1 THEN PRINT "4-Е МЕСТО - БЕЛОВ "
440 IF B5=1 THEN PRINT "5-Е МЕСТО - БЕЛОВ "
450 IF A2=1 THEN PRINT "2-Е МЕСТО - АДАМОВ"
460 IF D5=1 THEN PRINT "5-Е МЕСТО - ДРОНОВ"
470 IF W4=1 THEN PRINT "4-Е МЕСТО - ВЕТРОВ"
480 END

```

2-Е МЕСТО - ДРОНОВ
 1-Е МЕСТО - ЕРШОВ
 3-Е МЕСТО - ГЛЕБОВ
 5-Е МЕСТО - БЕЛОВ
 4-Е МЕСТО - ВЕТРОВ

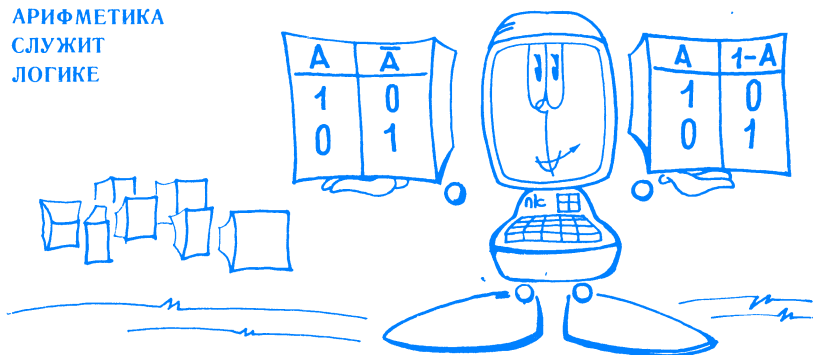
Обращаем внимание читателей на то, что ответ ЭВМ выдала в текстовой форме, она назвала каждого спортсмена и занятое им место. Это сделано с помощью небольшой дополнительной программы (она приводится ниже основной). Ответ на вопрос задачи помещен в таблице 9.

Таблица 9

Таблица результатов

Места	Фамилии спортсменов
1	Ершов
2	Дронов
3	Глебов
4	Ветров
5	Белов

АРИФМЕТИКА
 СЛУЖИТ
 ЛОГИКЕ



В некоторых упрощенных версиях языка Бейсик встроенных логических операций нет. И это, на первый взгляд, ограничивает возможности применения ЭВМ. Однако это не так. Решение логических задач возможно на любой ЭВМ. Дело в том, что можно построить арифметические модели основных логических операций. Создание таких моделей — это еще одна увлекательная задача для программиста.

Ниже приводятся очень простые модели трех операций: логического отрицания, логического умножения и логического сложения.

Таблица 10

A	\overline{A}
1	0
0	1

Таблица 11

A	$1-A$
1	0
0	1

Напомним таблицу, пользуясь которой, находят значения логического отрицания (табл. 10). В таблице 11 формула $1-A$ — уже формула арифметическая. Пользуясь этой формулой, можно получать такие же результаты, как и выполняя операцию логического отрицания над значением переменной A . Таким образом, формула $1-A$ есть арифметическая модель операции логического отрицания. И, если, например, следует вычислить \overline{B} (логическое отрицание B), то достаточно вычислить значение арифметического выражения $1-B$.

Иначе говоря, вместо оператора: 5 LET C=NOT B можно использовать оператор: 5 LET C=1-B. Читатель согласится, что логическое умножение и умножение арифметическое, выполняемые над значениями переменных, например A и B , дают одинаковые результаты. Иначе говоря, вместо оператора: N LET C=A.AND.B можно использовать оператор: N LET C=A*B.

Более сложной является модель операции логического сложения. Из таблиц 12—13 ясно, как это делается.

Таблица 12

A	B	$A+B$
1	1	1
1	0	1
0	1	1
0	0	0

Таблица 13

A	B	$A+B-A*B$
1	1	1
1	0	1
0	1	1
0	0	0

Вместо логического оператора: N LET C=A.OR.B можно использовать арифметический оператор: N LET C=A+B-A*B.

Из сказанного следует, что любую формулу, в записи которой применяются операции .NOT., .OR. и .AND., можно записать только с использованием арифметических операций. Это означает, что логические вычисления будут полностью заменены арифметическими.

Ограничимся небольшим примером. Пусть в текст конкретной программы следует включить вычисление значения логического выражения $A+B \cdot C$, результат необходимо присвоить переменной P .

Решение может быть таким:

$$\begin{aligned} 15 \text{ LET } P &= (1 - (A + (1 - B) - A * (1 - B))) * C = \\ &= (1 - A) * B * C. \end{aligned}$$

Решение может быть и таким:

15 LET P=.NOT.(A.OR..NOT.B).AND.C.

Второе решение чисто арифметическое, но P получит те же значения, что и при вычислении логического выражения.

Для читателей, интересующихся математической логикой, приведем арифметические модели логических операций «импликация» (табл. 14—15) и «эквивалентность» (табл. 16—17).

Арифметическая модель импликаций

Таблица 14

A	B	$A \rightarrow B$
1	1	1
1	0	0
0	1	1
0	0	1

Таблица 15

A	B	$1 - A + A * B$
1	1	1
1	0	0
0	1	1
0	0	1

Арифметическая модель эквивалентности

Таблица 16

A	B	$A \leftrightarrow B$
1	1	1
1	0	0
0	1	0
0	0	1

Таблица 17

A	B	$1 - (A - B)^2$
1	1	1
1	0	0
0	1	0
0	0	1

Все сказанное о связи между логическими и арифметическими операциями иллюстрируется рисунками 41—42.

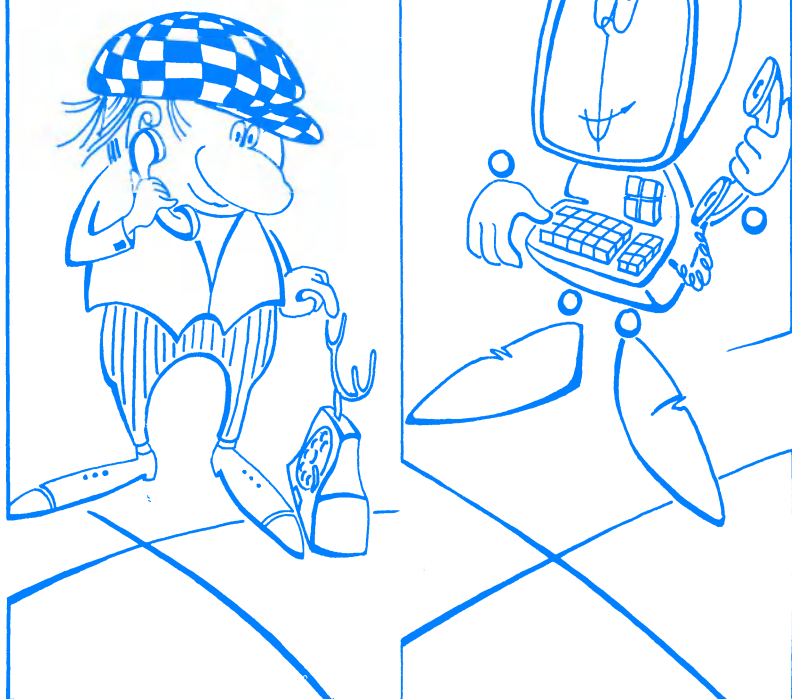
Отсюда следует важное заключение. При разработке программ можно использовать арифметические модели логических операций. В этом случае арифметика помогает логике. Начинающие программисты избегают использования в своих программах элементов логики, а зря — можно указать много примеров хорошего совмещения в одной программе арифметических и логических операций. Настоятельно рекомендуем ознакомиться с логическими возможностями условного оператора «IF...THEN...ELSE...». Для этого следует внимательно изучить версию языка Бейсик, которой Вы пользуетесь.

ГЛАВА

IV.

В ДИАЛОГЕ С

ЭВМ



АВТОМАТИЧЕСКИЙ ТЕЛЕФОННЫЙ СПРАВОЧНИК. В СОПЕРНИЧЕСТВЕ С ЧЕЛОВЕКОМ. ЗАДАЧА О ФАЛЬШИ- ВОЙ МОНЕТЕ.

Каждая программа — отражение достоинств и недостатков ее составителя.

Из фольклора программистов

Умение ЭВМ работать не только с числами, но и со словами — основа создания телефонных справочных бюро и более сложных информационно-поисковых систем.

ЭВМ — соперник человека в игре. Как машину учат играть и что значит научить «выигрывать»? Алгоритмические и программистские проблемы сплелись в один клубок.

Школьник познакомится с так называемым диалоговым режимом взаимодействия человека и ЭВМ. Автоматический телефонный справочник «понимает» о чем его спрашивают, чем интересуются: номером телефона или фамилией его владельца. ЭВМ точно отвечает на поставленный вопрос, ответу предшествует небольшой диалог.

Организуя соперничество человека с ЭВМ, нужно предусмотреть обмен репликами между партнерами. В начале игры ЭВМ может обратиться к человеку с вопросами, уточняющими условия поединка. В процессе игры следует проверять, не нарушают ли соперники правил игры, и в случае необходимости ЭВМ может указать человеку на его ошибку.

В конце игры, как правило, ЭВМ спрашивает, не желает ли человек сыграть еще одну партию, благодарит за участие в игре.

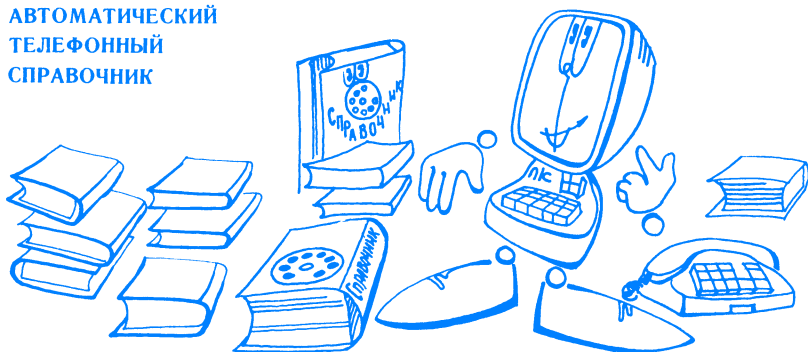
Диалоговый режим используется очень часто. В качестве примера можно назвать большую совокупность программ, предназначенную для решения систем линейных уравнений. Приступая к сотрудничеству с человеком, ЭВМ может запросить у него количество уравнений, число переменных. Получив такие сведения, машина может предостеречь человека от применения неподходящего метода. Так, если определитель системы близок к нулю, ЭВМ рекомендует не применять для решения метод Крамера, а предложит для выбора перечень иных способов решения системы уравнений. Программы всех методов хранятся в памяти ЭВМ.

Особенно распространен режим диалога в больших информационно-справочных системах. К таким системам следует отнести системы резервирования проездных билетов, мест в гостиницах, форм туристического обслуживания.

Создание хорошей диалоговой системы требует ясного понимания сущности задачи, осмысления возможных потребностей человека.

Предлагаемые задачи позволяют школьникам разобраться в особенностях конструирования диалога между ЭВМ и пользователем.

**АВТОМАТИЧЕСКИЙ
ТЕЛЕФОННЫЙ
СПРАВОЧНИК**



Поступил заказ на разработку автоматического телефонного справочника, центральным узлом которого следует сделать ЭВМ. Заказчики хотят, чтобы ЭВМ обеспечила возможность получения ответов на такие вопросы: какой номер телефона, например, у Иванова? У кого на квартире установлен телефон, например, 7-58?

Взаимодействие между человеком и ЭВМ они представляют себе так. Человек на пульте набирает номер автоматической справочной, для этого он вводит с клавиатуры число 09, и ЭВМ на экране дисплея в ответ высвечивает текст:

**ВЫ ХОТИТЕ УЗНАТЬ НОМЕР ТЕЛЕФОНА ПО ИМЕНИ
ВЛАДЕЛЬЦА?**

ВВЕДИТЕ ОТВЕТ «ДА» ИЛИ «НЕТ».

Если человек отвечает «ДА», то ЭВМ выдает на экран текст:
ВВЕДИТЕ ФАМИЛИЮ ВЛАДЕЛЬЦА ТЕЛЕФОНА.

Прочитав фамилию, ЭВМ должна найти соответствующий номер телефона и вывести на экран справку, например такую:
ИВАНОВ 562.

Если человек на первый вопрос отвечает словом «НЕТ», то ЭВМ должна на экране высветить текст:

**ВЫ ХОТИТЕ ПО НОМЕРУ ТЕЛЕФОНА УЗНАТЬ ФАМИ-
ЛИЮ ВЛАДЕЛЬЦА?**

ДА ИЛИ НЕТ?

Если человек отвечает «ДА», то ЭВМ выдает на экран сообще-
ние:

ВВЕДИТЕ НОМЕР ТЕЛЕФОНА.

Рассмотрев номер телефона, ЭВМ должна в своей памяти найти фамилию владельца и вывести на экран справку, например такую:

684 КОЛЕСОВ

Если на второй вопрос, поставленный машиной, человек отве-

чает «НЕТ», то ЭВМ должна на экране дисплея выдать сообщение: «ДРУГИХ СПРАВОК НЕ ДАЕМ».

Конечно, это очень и очень простое «справочное бюро»; оно дает ответы всего на два вопроса. Однако, если телефонов очень много, то быстро подготовить ответ совсем не просто. Применение ЭВМ для создания таких автоматических справочных вполне оправдано. Описанное «справочное бюро» можно считать простейшим примером так называемых информационно-справочных систем. Такие системы входят в нашу жизнь. Примером может быть информационно-справочная система «Сирена-2», которая не только хранит информацию о всех рейсах самолетов, о проданных и не проданных билетах на каждый рейс, но и может «продать» билет, указав сведения о номере рейса, месте и времени вылета. Все это ЭВМ напечатает на бланке билета с помощью своего устройства вывода.

Предлагаем попробовать свои силы в разработке программного обеспечения описанной выше простейшей информационно-справочной системы.

Начнем с того, что представим себе систему в виде очень крупных блоков (рис. 43).

С одной стороны система обращена к тем людям, которые задают вопросы, с другой — она подключена к рабочему месту инженера-математика, которому поручена эксплуатация системы.

Программное обеспечение, которым должна быть снабжена система, призвано работать «на два фронта»: обеспечивать диалог системы с человеком и помогать инженеру быстро вносить изменения в памяти. Изменения могут быть, например, такими: абонент выехал и номер его телефона стал теперь номером телефона другого человека или кому-то пришлось изменить номер телефона. Удобно, если номера телефонов расположены в памяти ЭВМ в каком-то порядке, например, в порядке возрастания, а фамилии — в соответствии с русским алфавитом. В связи с этим возникают оперативные задачи: появляющийся новый номер телефона следует включить в соответствующее место так, чтобы порядок в расположении номеров или в следовании фамилий владельцев телефонов не нарушался. Могут возникнуть и другие задачи.

Начнем с разработки программы, обеспечивающей диалог между человеком и информационно-справочной системой. Заметим сразу же, что в этом случае ЭВМ будет работать с числами (номера телефонов) и строковыми переменными (фамилии владельцев телефонов). Напомним, что простая переменная, например F, может принимать только числовые значения, если же ее снабдить значком \$, то переменная F\$ будет уже принимать символьные или строковые значения. Оператор LET F=15.2 присваивает переменной F значение числовое — 15.2, оператор LET F\$=«ИВАНОВ» присваивает строковой переменной F\$ значение «ИВАНОВ». ЭВМ может сравнивать значения строковых переменных и узнавать совпадающие. Это дает

возможность организовывать разветвления в программе, например, такого типа: IF F\$ > R\$ или IF F\$ = L\$.

Ниже приводится готовая программа автоматического справочного бюро:

```
10 REM АВТОМАТИЧЕСКИЙ ТЕЛЕФОННЫЙ СПРАВОЧНИК
20 DIM T(900), F$(900)
30 FOR I=111 TO 880
40 LET T(I)=I:NEXT I
50 LET F$(112)="БИРЮКОВ"
60 LET F$(263)="КОЗЛОВ"
70 LET F$(706)="РУСЛАНОВ"
80 PRINT "ВВЕДИТЕ НОМЕР 09"
90 INPUT N
100 IF N<>9 THEN 80
110 PRINT "ВЫ ХОТИТЕ УЗНАТЬ НОМЕР ТЕЛЕФОНА"
120 PRINT "ПО ИМЕНИ ЕГО ВЛАДЕЛЬЦА, ДА ИЛИ НЕТ"
130 INPUT S$
140 IF S$="НЕТ" THEN 220
150 PRINT "ВВЕДИТЕ ФАМИЛИЮ ВЛАДЕЛЬЦА"
160 INPUT L$
170 FOR I=111 TO 880
180 IF L$=F$(I) THEN G=I:GOTO 270
190 NEXT I
200 PRINT "ТЕЛЕФОНА НЕ ИМЕЕТ"
210 GOTO 80
220 PRINT "ВЫ ХОТИТЕ УЗНАТЬ ФАМИЛИЮ"
230 PRINT "ВЛАДЕЛЬЦА ПО НОМЕРУ ТЕЛЕФОНА"
240 PRINT "ВВЕДИТЕ НОМЕР ТЕЛЕФОНА "
250 INPUT G
260 IF F$(G)=" " THEN 300
270 PRINT T(G);F$(G);
280 PRINT " ВСЕГДА К ВАШИМ УСЛУГАМ"
290 GOTO 80
300 PRINT "ТЕЛЕФОН НЕ УСТАНОВЛЕН"
310 GOTO 80
```

Дадим некоторые пояснения. Операторы DIM T(900), DIM F\$(900) резервируют место в памяти ЭВМ для девятисот номеров телефонов и фамилий их владельцев. Каждый элемент массива T (это массив номеров телефонов) в цикле получает значение из чисел, принадлежащих интервалу от 111 до 880.

В приведенной программе с помощью трех операторов LET трем гражданам — Бирюкову, Козлову и Русланову — «установлены» телефоны с номерами 112, 263 и 706. Вся остальная часть программы обеспечивает диалог между ЭВМ и тем, кто обращается к автоматическому справочному бюро. По окончании краткого диалога ЭВМ вежливо обратится к человеку со словами: «ВСЕГДА К ВАШИМ УСЛУГАМ».

Завершая рассказ о справочном бюро, подчеркнем, что в данном случае мы работали с двумя массивами — элементами одного были числа (номера телефонов), элементами другого — символьные или строковые величины. Элементы массивов использовались парами — каждой фамилии соответствовал свой закрепленный номер.

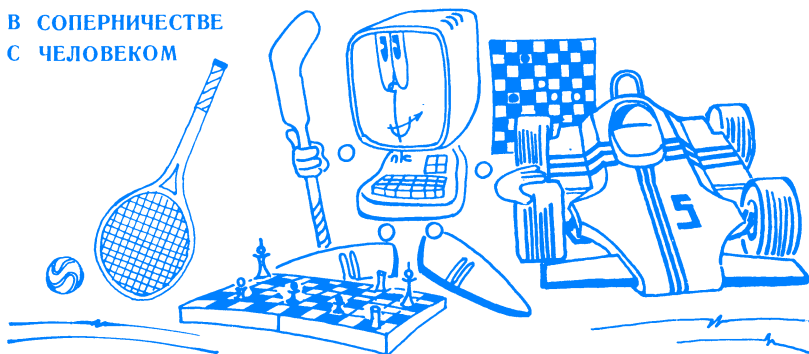
Идеи, на которых реализована система, можно развить дальше и создать, например, англо-русский и одновременно русско-

английский словарь. В этом случае будут использоваться два массива, находящиеся друг с другом во взаимно-однозначном соответствии. Оба массива будут состоять из строковых переменных; их значениями будут английские и русские слова.

Нетрудно представить себе, что справочники такого типа можно создавать по различным видам человеческой деятельности. Важным, например, может быть «автоматическое адресное бюро» — в этом случае в памяти ЭВМ будут храниться два массива: один из фамилий граждан, а другой — из их адресов, а также, как в рассмотренном выше «автоматическом телефонном справочнике», в этой системе будет возможность получать ответы на вопросы такого типа: Где проживает Иванов? Кто проживает в доме № 5, в квартире № 17 по улице Киевской?

Составить программу для того, чтобы ЭВМ могла выступить в роли такого адресного бюро, вполне по силам школьнику. Так постепенно в школу входят новые задачи, совсем недавно, казалось бы, лежащие в стороне.

В СОПЕРНИЧЕСТВЕ С ЧЕЛОВЕКОМ



Как только появились первые ЭВМ, так сразу же начались попытки научить их различным играм. Человека влекла проблема воспроизведения атмосферы игрового поединка. Людей всегда привлекали секреты великих шахматистов, шашкистов, игроков в игру «Го» и мастеров многих других национальных и признанных во всем мире игр. Программист, пытающийся научить ЭВМ хорошо играть в какую-нибудь игру, вынужден более глубоко анализировать всю деятельность игрока-человека. Написание игровых программ способствовало и развитию искусства программирования. Создание игровых программ требует от человека больших знаний о той игре, которая моделируется, изобретательности, настойчивости и воображения.

Ниже будет рассмотрен пример создания программы, работая по которой, ЭВМ выступит соперником человека в очень простой

игре. В ходе рассказа будут подчеркнуты наиболее важные для разработки таких программ детали.

Прежде всего о самой игре. Эта игра на тему: «Выбирание предметов». Она похожа на игру Баше, НИМ, Цзяншицзы, игру Гранди и другие. Во всех этих играх происходит выборание предметов из одной общей или нескольких групп. Победителем станет тот, кто сумеет взять последний предмет (в одних играх) или оставить его сопернику (в других). В любой момент игры каждый из соперников располагает полной информацией об игре: он знает, сколько предметов в общей группе, сколько их у каждого из играющих, какие ходы делали соперники и многое другое. Во всех таких играх существует алгоритм безошибочного ее проведения.

Условия игры. К началу игры в общей группе содержится некоторое количество предметов. Два игрока поочередно выбирают из общей группы любое количество предметов, не превышающее половины имеющихся в ней. Проигравшим считается тот, кто берет последний предмет.

Прежде чем начать разговор о программировании, необходимо изучить алгоритм безошибочной игры. Суть его проста: в игру следует вступать первым, если начальное количество предметов не является числом вида $N=2^k-1$. Любой свой ход следует делать так, чтобы после него в общей группе количество оставшихся предметов выражалось числом вида $N=2^k-1$. Если начальное количество предметов в общей группе выражается числом вида 2^k-1 , то первый ход следует предоставить сопернику, а каждый свой ход делать так, как указано выше.

Доказательство алгоритма приводить не будем, это может сделать и читатель. Упомянем только о том, что ему рекомендуется применить метод математической индукции.

Итак, мы знаем, как следует действовать, если мы будем с кем-то играть в такую игру. Как же научить этому ЭВМ? Чему ее следует научить?

Основных проблем две: первая — научить участвовать в игре. Это значит делать ходы, не нарушая правил, разбираться в возникающих ситуациях. Вторая проблема — научить выигрывать. Это значит уметь пользоваться безошибочным алгоритмом.

Приступая к решению этих больших проблем, определим для себя, как программиста, более простые задачи. Прежде всего научим ЭВМ следить за стратегической направленностью действий человека в игре. Что это значит? Пусть, по условиям поединка человека и ЭВМ, право выбора первого хода принадлежит человеку. При некотором начальном количестве предметов (N) человек может принять неверное решение о том, каким по очереди вступать в игру. Если это случится, то ЭВМ должна действовать строго в соответствии с безошибочным алгоритмом и выиграть. Как она будет это делать, рассмотрим ниже. Может, однако, случиться и так, что человек правильно ответит на вопрос, каким

по очереди вступать в игру. И, более того, он каждый следующий ход выполняет так, как предписывает беспроеигрышная стратегия. В этом случае ЭВМ должна «признать», что ее соперник — «крепкий орешек», и ей остается только выжидание, надежда «поймать» человека на ошибке при выполнении какого-либо очередного хода.

Как видно, дел предстоит немало. Совокупность действий ЭВМ, обеспечивающих контроль за стратегией, используемой человеком, образует блок, названный нами **блок контроля алгоритма (БКА)**. Если ЭВМ сумела распознать, что человек ошибся или не знает беспроеигрышного алгоритма, то вся ее работа подчиняется «командам», входящим в **блок выигрыша (БВ)**.

Конечно, в процессе игры ЭВМ должна следить за соблюдением человеком правил игры. В случае нарушения правил ЭВМ должна подавать сопернику соответствующие реплики. Все действия такого характера формируются в **блоке контроля за правильностью хода (БКПХ)**. И, наконец, попытки «поймать» человека на ошибке делаются с помощью команд, входящих в **блок проигрыша (БП)**.

Взаимосвязь перечисленных блоков можно схематически изобразить так, как показано на рисунке 44.

Несколько разъяснений. Если человек принял неверное решение о том, каким по очереди вступать в игру, то управление процессом поединка передается блоку выигрыша (БВ), который доводит игру до победы ЭВМ. Блок выигрыша «сотрудничает» только с блоком контроля за правильностью хода человека (БКПХ).

Блок проигрыша (БП) ведет игру в ситуации, трудной для ЭВМ. Задача этого блока — обеспечить терпеливое выжидание в надежде, что человек — соперник ЭВМ — допустит ошибку. Если ошибка произойдет, то управление немедленно будет передано блоку выигрыша (БВ). Конечно, блок проигрыша должен сотрудничать с БКПХ.

В блоке БКПХ проверяется, верно ли человек выполнил очередной ход. В блоке ведется сравнение числа выбранных человеком M предметов с половиной имевшихся предметов в общей группе до хода человека $\frac{N}{2}$. Если $M \leq \frac{N}{2}$, то ход выполнен верно, если же $M > \frac{N}{2}$, то человек допустил ошибку, и ЭВМ должна подать ему реплику. Реплика может быть, например, такой:

ВЫ НАРУШИЛИ ПРАВИЛА ИГРЫ, ПОВТОРИТЕ ХОД.

Приступая к составлению программы, продумаем приглашение к игре, с которым ЭВМ обратится к человеку. Здесь каждый программист может проявить свой вкус и изобретательность. Ниже предлагается один из возможных вариантов программы. Рассмотрим ее первую часть.


```

10 PRINT "ЗДРАВСТВУЙТЕ. ЭВМ ПРЕДЛАГАЕТ ВАМ ПОЕДИНОК"
20 PRINT "ПОЗНАКОМЬТЕСЬ С УСЛОВИЯМИ ИГРЫ."
30 PRINT "ДАНА ГРУППА ИЗ N ПРЕДМЕТОВ."
40 PRINT "ИГРАЮТ ДВОЕ, ХОДЯТ ПО ОЧЕРЕДИ."
50 PRINT "ЗА ХОД МОЖНО БРАТЬ ИЗ ИМЕЮЩЕЙСЯ ГРУППЫ"
60 PRINT "ЛЮБОЕ К-ВО ПРЕДМЕТОВ, НЕ БОЛЕЕ ПОЛОВИНЫ"
70 PRINT "ПРОИГРЫВАЕТ ТОТ, КТО БЕРЕТ ПОСЛЕДНИЙ ПРЕДМЕТ"
80 PRINT "ПРИНИМАЕТЕ ВЫЗОВ? ДА ИЛИ НЕТ?"
90 INPUT V$
100 IF V$="НЕТ" THEN PRINT "СОЖАЛЕЮ, ЧТО ИГРА НЕ СОСТОЯЛАСЬ"
110 PRINT "БЛАГОДАРИЮ. НАЧИНАЕМ ИГРУ"
120 PRINT "НАЗОВИТЕ N - ИСХОДНОЕ К-ВО ПРЕДМЕТОВ"
130 INPUT N
140 REM 'ВЫЯСНЕНИЕ, ЯВЛЯЕТСЯ ЛИ ЧИСЛО N
150 REM 'ЧИСЛОМ ВИДА 2^k-1
160 LET X=1
170 LET X=X*2
180 IF X<N THEN 170
190 LET X=X-1
200 IF X=N THEN LET V=1 ELSE LET V=3
210 PRINT "КАКИМ ПО ОЧЕРЕДИ ХОТИТЕ ВСТУПАТЬ В ИГРУ?"
220 PRINT "ПЕРВЫМ? ДА ИЛИ НЕТ?"
230 INPUT W$
240 IF W$="ДА" AND V=1 THEN 280
250 IF W$="ДА" AND V=3 THEN 490
260 IF W$="НЕТ" AND V=1 THEN 600
270 IF W$="НЕТ" AND V=3 THEN 350

```

В строках 10—100 ЭВМ знакомит человека с условиями игры и вызывает его на поединок. Если человек принимает вызов, то она запрашивает у него начальное количество предметов N (строки 110—130). После этого ЭВМ спрашивает человека, каким по очереди он намерен вступить в игру: первым или вторым.

Получив ответ, ЭВМ выясняет, является ли число N числом вида $2^k - 1$ (строки 140—200). Затем ЭВМ принимает решение о том, как вести игру. Решение базируется на знании двух фактов: известно, каким по очереди ходит человек и является ли число N числом вида $2^k - 1$.

Решение ЭВМ можно разъяснить таблицей 18.

Таблица 18

Ход человека	Тип числа N	Решение ЭВМ
1-й	$N = 2^k - 1$	Передать управление БВ — 280
1-й	$N \neq 2^k - 1$	Передать управление БП — 490
2-й	$N = 2^k - 1$	Передать управление БП — 600
2-й	$N \neq 2^k - 1$	Передать управление БВ — 350

Вторая часть программы (строки 280—470) содержит операторы блока выигрыша ЭВМ (БВ). Здесь же ведется расчет наилучшего хода ЭВМ (строки 340—400) (см. вклейку).

В третьей части программы содержится блок проигрыша ЭВМ. Здесь же имеются операторы, которые проверяют, следует ли человек алгоритму, иначе говоря выясняют: оставляет ли человек после каждого хода количество предметов, выражающееся числом вида $2^k - 1$ (строки 540—590).

```
480 REM ИСПОЛНЕНИЯ АЛГОРИТМА ЧЕЛОВЕКОМ
490 INPUT "ВАШ ХОД. БЕРИТЕ R ПРЕДМЕТОВ R=";R
500 LET N=N-R
510 IF N<=1 THEN 420
520 IF R>N/2 THEN PRINT "НАРУШЕНЫ ПРАВИЛА":GOTO 490
530 PRINT "ОСТАЛОСЬ N=";N;"ПРЕДМЕТОВ"
540 REM АНАЛИЗ ХОДА ЧЕЛОВЕКА. ОСТАВИЛ ЛИ ОН
550 REM КОЛ-ВО ПРЕДМЕТОВ N=2^K-1?
560 LET L=1
570 LET L=L*2
580 IF L<=N THEN 570
590 IF N<>L-1 THEN 350
595 LET M=1
600 PRINT "ДЕЛАЮ ХОД. БЕРУ M=";M;"ПРЕДМЕТОВ"
620 PRINT "ОСТАЛОСЬ N=";N-M;"ПРЕДМЕТОВ"
630 GOTO 490
```

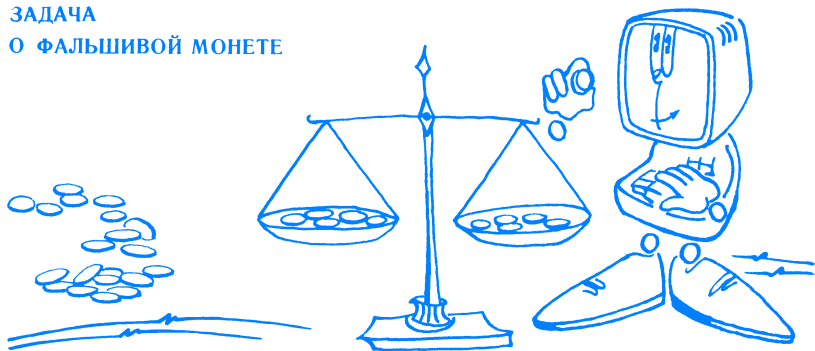
Такой может быть программа, реализующая игровой поединок между человеком и ЭВМ. Конечно, программа может быть составлена иначе, может включать другие или более развернутые реплики, подаваемые машиной человеку. Можно в программу ввести своеобразные «шахматные часы» (таймер). Эти часы будут включаться, когда ЭВМ выдаст человеку сообщение «ВАШ ХОД», затем будут вести отсчет времени, отведенного на один ход, например 30 секунд. Если человек просрочит время, то ЭВМ выдаст человеку сообщение такого типа: «ВЫ ПРОСРОЧИЛИ ВРЕМЯ. ВАМ ЗАСЧИТАЛИ ПРОИГРЫШ».

При желании в программу можно ввести счетчик количества нарушений правил игры человеком. Если, например, человек три раза нарушил правила, то ЭВМ может отказаться с ним играть.

Конечно, разработка таких программ требует настойчивости и собранности. Но дело стоит затрачиваемых усилий и времени. Ваши друзья с удовольствием попробуют сразиться с ЭВМ. Заметим, что такие интеллектуальные игры человека с ЭВМ следует приветствовать; эти игры способствуют развитию мышления. Мы против игр типа различных гонок на дорогах, космических бомбардировок и других игр, часто насаждаемых под предлогом привлечения школьников к ЭВМ. Сочетание занимательности с интеллектуальной нагрузкой принесет не только больше пользы школьнику, но и большее удовлетворение.

На рисунках 45 и 46 изображены схемы, которые помогут разобраться в том, как осуществляется передача управления из одной части к другой. Передача управления определяется конкретной ситуацией, складывающейся в ходе игры.

ЗАДАЧА О ФАЛЬШИВОЙ МОНЕТЕ



Пусть имеется 12 неотличимых по внешнему виду монет. Одна из монет фальшивая; она отличается от настоящих по весу, и при этом неизвестно — тяжелее она или легче настоящей монеты.

Можно ли за три взвешивания на весах без гирь: обнаружить фальшивую монету; определить, легче или тяжелее она настоящей? Оказывается, можно.

Делается это так. Все монеты пронумеровываются (им присваиваются номера от 1 до 12). Взвешивание проводится три раза. При первом взвешивании на левую чашу весов кладутся монеты: 1, 2, 3 и 4, а на правую — 5, 6, 7 и 8 (см. рис. 47). При втором взвешивании на левую чашу весов кладутся монеты: 1, 2, 3 и 5, а на правую — 4, 9, 10, 11 (рис. 48). При третьем взвешивании на левую чашу кладутся монеты: 1, 6, 9 и 12, а на правую — монеты: 2, 5, 7 и 10 (рис. 49).

При каждом отдельном взвешивании наблюдение ведется только за левой чашей весов. Может обнаружиться три случая: монеты на левой чаше легче, тяжелее или равны по суммарному весу монетам, размещенным на правой чаше. Каждое из трех взвешиваний дает три результата. Все возможные комбинации результатов можно изобразить в виде графа-дерева так, как показано на рисунке 50.

Если внимательно проследить за результатами по каждой из 27 ветвей дерева, то можно обнаружить, что в 12 случаях задуманная монета легче и в 12 — тяжелее. Можно догадаться (требуется настойчивость в анализе каждой ветви графа-дерева), какой номер имеет фальшивая монета.

Все эти результаты отражены на рисунке 50. По существу, этот рисунок задает таблицу соответствий троек типа «ЛЛР» паре типа «монета фальшивая». Располагая такой таблицей, можно приступить к созданию диалоговой программы. Действуя по программе, ЭВМ должна определить фальшивую монету.

Вариант программы приводим ниже.

```

5 REM ПОИСК ФАЛЬШИВОЙ МОНЕТЫ
10 DIM A$(12), B$(12), M$(3)
15 DATA ЛЛЛ, ЛЛТ, ЛЛР, ЛТР, ТЛТ, ТРЛ, ТРТ, ТРР, РТЛ, РТТ, РТР, РРЛ
20 DATA ТТТ, ТТЛ, ТТР, ТЛР, ЛТЛ, ЛРТ, ЛРЛ, ЛРР, РЛЛ, РЛР, РРТ
25 FOR I=1 TO 12:READ A$(I):NEXT I
30 FOR I=1 TO 12:READ B$(I):NEXT I
35 LET K=1
40 PRINT "ЗАДУМАЙТЕ ФАЛЬШИВУЮ МОНЕТУ"
45 PRINT "ЗАДУМАЛИ, ДА ИЛИ НЕТ"
50 INPUT L$
55 IF L$="НЕТ" THEN 40
60 PRINT "НА ЛЕВУЮ ЧАШУ ПОЛОЖИТЕ МОНЕТЫ:"
65 PRINT "1, 2, 3 И 4 , НА ПРАВУЮ :5, 6, 7 И 8"
70 GOSUB 140
75 PRINT "НА ЛЕВУЮ ЧАШУ ПОЛОЖИТЕ МОНЕТЫ:"
80 PRINT "1, 2, 3 И 5 , НА ПРАВУЮ :4, 9, 10 И 11"
85 GOSUB 140
90 PRINT "НА ЛЕВУЮ ЧАШУ ПОЛОЖИТЕ МОНЕТЫ:"
95 PRINT "1, 6, 9 И 12 , НА ПРАВУЮ :2, 5, 7 И 10"
100 GOSUB 140
105 LET M$=M$(1)+M$(2)+M$(3)
110 FOR I=1 TO 12
115 IF M$=A$(I) THEN PRINT "МОНЕТА ЛЕГКАЯ"; I:GOTO 130
120 IF M$=B$(I) THEN PRINT "МОНЕТА ТЯЖЕЛАЯ"; I:GOTO 130
125 NEXT I
130 END
135 '*****
140 PRINT "ТЯЖЕЛЕЕ ЛИ ЛЕВАЯ ЧАША, ДА ИЛИ НЕТ"
145 INPUT "R$="; R$
150 IF R$="ДА" THEN LET M$(K)="Т"
155 PRINT "ЛЕГЧЕ ЛИ ЛЕВАЯ ЧАША, ДА ИЛИ НЕТ"
160 INPUT "R$="; R$
165 IF R$="ДА" THEN LET M$(K)="Л" ELSE LET M$(K)="Р"
170 LET K=K+1
175 RETURN

```

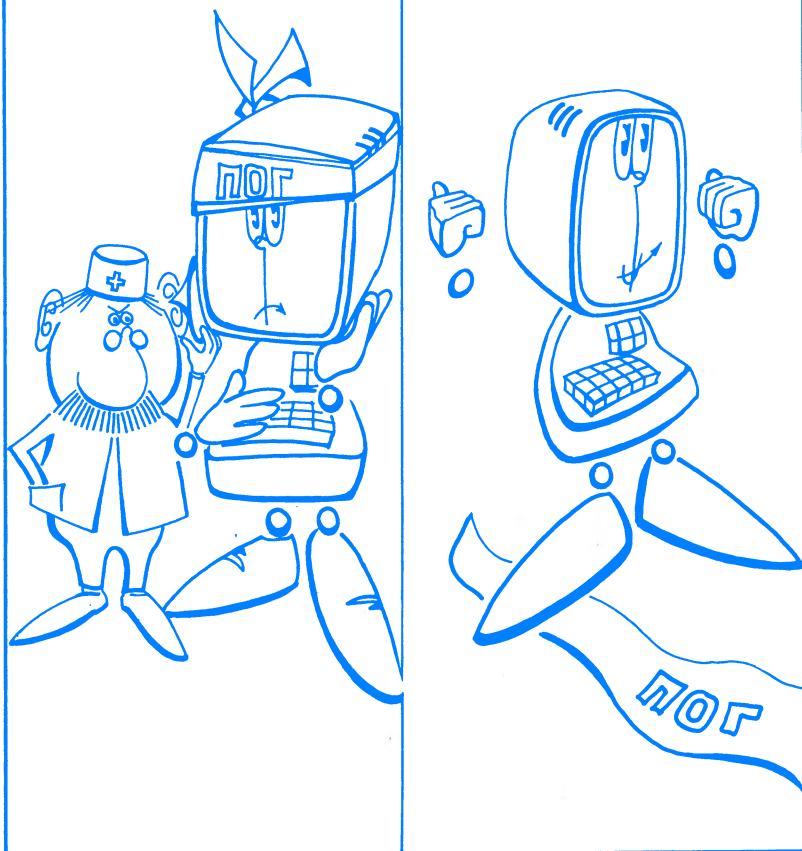
В программе можно выделить такие части:

- в строках 10—30 формируется массив слов, выражающих все комбинации результатов взвешиваний монет;
- в строках 35—100 ЭВМ ведет диалог с человеком, задумавшим монету, руководя его действиями при взвешивании монеты;
- в строках 140—175 подпрограмма формирования результата одного взвешивания. ЭВМ следит за одной, только левой, чашей весов и фиксирует результат взвешивания. Результат всегда один из трех: монета на левой чаше тяжелее (Т), легче (Л) или вес монеты на левой чаше равен весу монеты на правой (Р);
- в строках 105—125 формируется слово, выражающее результат всех трех взвешиваний, ведется поиск в таблице соответствий и печатаются результаты поиска фальшивой монеты.

ГЛАВА

V

И В ШУТКУ,
И ВСЕРЬЕЗ



УПОРЯДОЧЕНИЕ ТАБЛИЦЫ НАБЛЮДЕНИЙ. ПОЛУЧИТЕ
СДАЧУ. МАГИЧЕСКИЙ ЛИ КВАДРАТ? ЗАДАНИЕ «КВАНТА».
ТРЕУГОЛЬНИК ПАСКАЛЯ И ТРЕУГОЛЬНИК ЛЕЙБНИЦА.
ЧИСЛОВАЯ СПИРАЛЬ. ПАУЗА В РАБОТЕ ЭВМ. СЛОЖЕНИЕ
В Q-АРИФМЕТИКЕ. АРИФМЕТИКА ЛЬВА ТОЛСТОГО. ДЛЯ
ВСЕХ «СРЕДНИХ» ОДНА ПРОГРАММА. НЕИЗВЕСТНОЕ ПОД
ЗНАКОМ ФАКТОРИАЛА. ПАЛИНДРОМ ЛИ? ОТ ЦЕНТРА
К ЦЕНТРУ. ЛИНЕЙКА НА ЭКРАНЕ. ПРЕОБРАЗОВАНИЕ
ТРЕУГОЛЬНИКА. СОРТИРОВКА. ЭВМ, ПОРАБОТАЙТЕ
С КРОССВОРДОМ. ГЕОМЕТРИЧЕСКИЕ ИДЕИ РЕАЛИЗУЕТ
ЭВМ.

*ПОГ — программирование
очертя голову.*

*ПОГ — «детская» болезнь, воз-
никающая в начальный
период обучения про-
граммированию.*

Берегись ПОГа

Из фольклора программистов

Школьник, вооруженный ЭВМ, чувствует себя сильным. ЭВМ — замечательный инструмент, а если ведется работа по преобразованию информации, то ЭВМ — самое совершенное оружие труда.

Все программы для решения занимательных задач разработаны школьниками, членами Малой академии наук Крыма «Искатель». В каждой из задач читатель найдет какую-нибудь изюминку искусства программирования и, может быть, возьмет себе ее на вооружение.

Мы надеемся, что разнообразие задач еще раз подчеркнет могущество ЭВМ. Рассматривая предлагаемые шуточные задачи, Вы еще раз убедитесь в важности поиска основной идеи решения задачи. Найти идею, представить ее в виде алгоритма — это самое трудное, но и самое интересное. На этапе поиска идеи решения, подхода к алгоритму проявляется смекалка, умение сосредотачиваться на задаче. Все задачи этой главы как бы приглашают читателя к поиску, углублению затронутых тем. Небольшой рассказ «Арифметика Льва Толстого» — это дверь в десятичные системы счисления. Откройте эту дверь, и Вы увидите, что и рассказ о логическом квадрате, о секретах конструирования квадратов требует знания десятичных арифметик. Рассказ об упорядочении таблицы наблюдений — это пример задачи из производственной практики, пример задачи, открывающей дверь в элементарную статистику.

По всем темам читатель найдет немало сведений в статьях журнала «Квант», на страницах брошюр общества «Знание» (рекомендуем серию «Вычислительная техника и ее применение»).

УПОРЯДОЧЕНИЕ
ТАБЛИЦЫ
НАБЛЮДЕНИЙ



Очень часто результаты наблюдений оформляются в виде таблицы, имеющей две строки (табл. 19).

Если эта таблица наблюдений упорядочена, то есть значение признака A расположено в порядке возрастания и под каждым значением A_i указано соответствующее значение признака B , то такую таблицу называют *вариационным рядом*.

Таблица 19

A_i	A_1	A_2	A_3	\dots				A_n
B_i	B_1	B_2	B_3	\dots				B_n

В первой строке указаны значения признака A , например, размер обуви мальчика, а во второй строке — количество мальчиков, имевших такой размер обуви, если обследованию подвергалось N мальчиков. Естественно потребовать, чтобы числа A_i были расположены в порядке возрастания.

Требуется составить программу, работая по которой, ЭВМ данную таблицу обследования преобразует так, что значения переменной будут расположены в порядке возрастания. При этом соответствующие значения переменной B не будут перепутаны.

Программа может быть такой, как приведенная ниже. Заметим, что в программе был применен оператор SWAP — оператор обмена значений между переменными. В программе вслед за заменой значений переменных $A(I)$ и $A(J)$ осуществляется замена переменных $B(I)$ и $B(J)$.

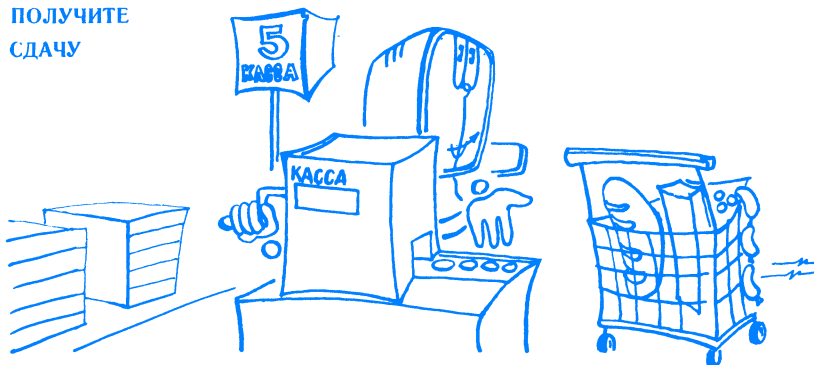
Такой программистский прием следует запомнить; он встречается часто.

```

5 PRINT"ВВЕДИТЕ ЧИСЛО ПАР ЧИСЕЛ В ТАБЛИЦЕ"
10 INPUT N
20 DIM A(N),B(N)
30 FOR I=1 TO N
35 PRINT"ВВЕДИТЕ A("I"),B("I")"
40 INPUT A(I),B(I)
50 NEXT
60 FOR I=1 TO N-1
70 FOR J=I+1 TO N
80 IF A(I)>A(J) THEN SWAP A(I),A(J):SWAP B(I),B(J)
90 NEXT J,I
100 FOR I=1 TO N
110 PRINT A(I);B(I)
120 NEXT
130 END
140 'SWAP - ОПЕРАТОР ОБМЕНА ЗНАЧЕНИЙ МЕЖДУ ПЕРЕМЕННЫМИ

```

ПОЛУЧИТЕ
СДАЧУ



В нашей практике используются монеты достоинством в 50, 20, 15, 10, 5, 3, 2 и 1 копейка. Какие монеты должен взять кассир, чтобы дать сдачу 79 копеек наименьшим количеством взятых монет? Ответ может быть таким: $79 = 50 + 20 + 5 + 3 + 1$.

Задача состоит в том, чтобы ЭВМ могла быстро подсказывать кассиру, какие монеты брать, если ему необходимо дать сдачу мелочью на сумму $S < 100$ копеек.

Идея решения задачи аналогична идее алгоритма замены десятичных чисел числами, записанными в римской нумерации.

Текст программы может быть таким:

```

10 'ПОЛУЧИТЕ СДАЧУ
20 INPUT "ВВЕДИТЕ ИСХОДНУЮ СУММУ S";S
25 PRINT "ПОЛУЧИТЕ СДАЧУ":PRINT
30 IF S>=50 THEN LET S=S-50:PRINT "50 КОП.-1 Т."
40 IF S>=20 THEN LET I=INT(S/20):PRINT "20 КОП.-";I;" Т.":
LET S=S-I*20
50 IF S>=15 THEN LET S=S-15:PRINT "15 КОП.-1 Т."
60 IF S>=10 THEN LET S=S-10:PRINT "10 КОП.-1 Т."
70 IF S>=5 THEN LET S=S-5:PRINT "5 КОП.-1 Т."
80 IF S>=3 THEN LET S=S-3:PRINT "3 КОП.-1 Т."
90 IF S>=2 THEN LET S=S-2:PRINT "2 КОП.-1 Т."
100 IF S>0 THEN PRINT "1 КОП.-1 Т."
110 END

```


Примеры решения других задач показаны в таблицах 20—21.

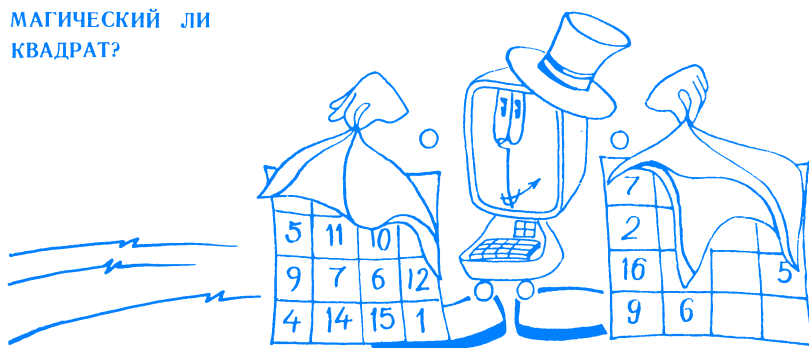
Таблица 20

?	99
50	коп. 1 шт.
20	коп. 2 шт.
5	коп. 1 шт.
3	коп. 1 шт.
1	коп. 1 шт.

Таблица 21

?	80
50	коп. 1 шт.
20	коп. 1 шт.
10	коп. 1 шт.

МАГИЧЕСКИЙ ЛИ КВАДРАТ?



Магическим квадратом называется квадрат из N^2 клеток, в каждой из которых записано одно из чисел от 1 до N^2 включительно и при этом суммы по любой горизонтали, вертикали и двум диагоналям равны одному и тому же числу. Число N называется *порядком квадрата*. На рисунке 51 приведены примеры двух квадратов 4-го порядка.

Левый из изображенных на рисунке квадратов называется *симметричным*. В этом квадрате сумма чисел, расположенных в одной строке, столбце или диагонали, равна 34. Утверждается, что первый и четвертый столбцы (или строки) квадрата можно поменять местами и его свойства не изменятся; не изменятся его свойства, если поменять местами второй и третий столбцы или эти же строки.

Второй из квадратов, изображенных на рисунке 51, называется *дьявольским*. Его свойства не изменятся, если его верхнюю строку поменять местами с нижней. Если в данном квадрате один из крайних столбцов стереть и затем приписать с противоположной стороны, то полученный квадрат будет, как и исходный, дьявольским.

Можно предложить составить программы проверки всех указанных особенностей квадратов. Такие программы составить несложно, но увлекательно.

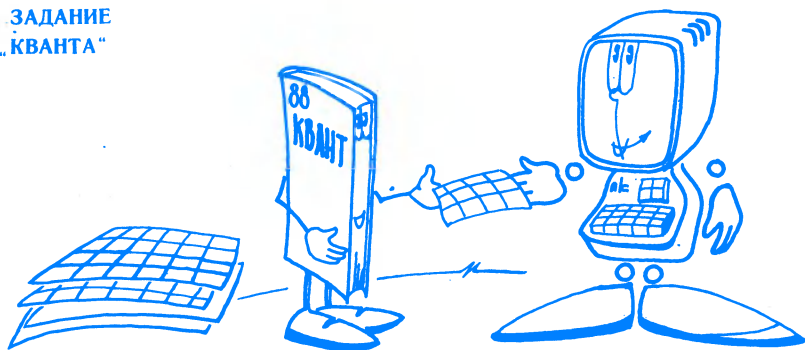
Более интересной является другая задача, которую решали школьники. На рисунке 52 дан необычный магический квадрат.

Это квадрат, составленный из 144 последовательных простых чисел. Постоянная этого квадрата $S=4514$. Число 2 — число простое, не используется, так как, входя в какую-то сумму чисел по горизонтали или вертикали, будучи четным, оно нарушало бы четность строки или столбца. Сейчас в квадрате по любой строке (столбцу и диагонали) суммируется 12 нечетных чисел.

Задача перед школьниками была сформулирована так: выяснить, является ли квадрат магическим; нет ли среди чисел, его образующих, чисел не простых, а составных?

Текст программы, составленной учащимися, выписан ниже.

```
5 REM ПРОВЕРКА МАГИЧЕСКОГО КВАДРАТА
10 DIM A(12,12),B(144)
20 FOR I=1 TO 12
30 FOR J=1 TO 12
40 PRINT I;J
50 INPUT A(I,J)
60 B=A(I,J):GOSUB 380
70 IF A=0 GOTO 100
80 PRINT A(I,J) "—ЧИСЛО НЕ ПРОСТОЕ"
90 GOTO 50
100 NEXT J, I
110 J=0
120 FOR I=1 TO 144
130 J=J+1
140 B=J:GOSUB 380
150 IF A<>0 GOTO 130
160 B(I)=J
170 NEXT I
180 FOR K=1 TO 144
185 A=0
190 FOR I=1 TO 12
200 FOR J=1 TO 12
210 IF B(K)=A(I,J) THEN A=1
220 NEXT J, I
230 IF A=0 THEN PRINT "НЕТ" K"—ГО ЧИСЛА"
240 NEXT K
250 A=0
260 K=A(1,1)+A(1,2)+A(1,3)+A(1,4)+A(1,5)+A(1,6)+A(1,7)+A(1,8)+
+A(1,9)+A(1,11)+A(1,12)
265 FOR I=2 TO 12
267 K1=0
270 FOR J=1 TO 12
280 K1=K1+A(I,J)
290 NEXT J
300 IF K1<>K THEN A=1
310 NEXT I:K1=0:K2=0
320 FOR I=1 TO 12:K1=K1+A(I,I):K2=K2+A(I,13-I)
330 NEXT I
340 IF K<>K1 OR K<>K2 THEN A=1
350 IF A=1 THEN PRINT "КВАДРАТ НЕ МАГИЧЕСКИЙ"
360 IF A=0 THEN PRINT "КВАДРАТ МАГИЧЕСКИЙ"
370 STOP
380 A=0
390 FOR D=2 TO INT(SQR(B))
400 IF B/D=INT(B/D) THEN A=1
410 NEXT D
420 RETURN
430 END
```



В журнале для школьников «Квант» (№ 5, 1980 г.) была опубликована задача: изучить таблицу, составленную из чисел. Таблица имела необычную форму (см. рис. 53). В левой верхней клетке таблицы помещено число 5. В каждом прямоугольнике суммировали числа. Число, записанное в правой нижней клетке любого прямоугольника с вершиной в «северо-западном» углу, есть сумма всех остальных чисел, записанных в клетках прямоугольника.

На рисунке 53 выделены два прямоугольника: в одном сумма чисел равна 10, в другом — 15. (Это квадрат, содержащий числа 5, 5, 5 и 15). Красным цветом выделен еще один прямоугольник — в правой нижней («юго-западной») клетке его записана сумма всех остальных пяти чисел — это сумма 40.

На основе рассказанного ставится задача на программирование: написать программу формирования таблиц такого типа. При этом требуется задавать предварительно число N , размещаемое в «северо-западном» углу, и указывать число клеток на стороне таблицы R . Результатом работы ЭВМ должна быть таблица, содержащая R^2 клеток; числа на экране и при печати должны изображаться так, как это показано на рисунке 54.

Текст программы, приводимой ниже, разработан школьником.

```
10 PRINT "ВВОД ИСХОДНОГО N"
20 INPUT "N=";N
30 INPUT "РАЗМЕР ТАБЛИЦЫ R=";R
40 DIM A(R,R)
50 FOR I=1 TO R
60 FOR J=1 TO R
70 FOR I1=I TO 1 STEP -1
80 FOR J1=J TO 1 STEP -1
90 LET A(I,J)=A(I1,J1)+A(I,J)
100 A(1,1)=N
110 NEXT J1
120 NEXT I1
130 NEXT J
140 NEXT I
150 LET X=15
```

```

160 CLS
170 FOR I=1 TO R
180 FOR J=1 TO R
190 PRINT TAB(X);A(I,J)
200 LET X=X+4
210 NEXT J
220 LET X=X-20
230 NEXT I

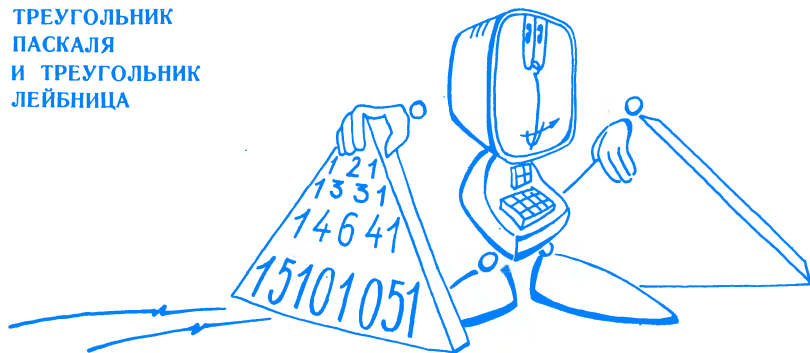
```

Рекомендуем изучить текст программы внимательно, в нем есть немало находок. Обращаем внимание на использование четырех встроженных циклов.

На рисунке 55 показано, как заполняются клетки таблицы числами, рассмотрен общий случай. Попробуйте составить рекуррентную формулу для получения числа, записываемого в текущую клетку таблицы, с координатами (I, J) . Может быть, удастся найти и формулу типа $a_{ij} = F(N, I, G)$, тогда, зная N и i, j , можно будет вычислить число, записываемое в клетку.

В этой задаче мы встретились с тем случаем, когда нас не привлекает алгоритм в виде программы (его трудно изучать), и нам хотелось бы получить алгоритм в более традиционной и знакомой форме, в виде формулы.

ТРЕУГОЛЬНИК
ПАСКАЛЯ
И ТРЕУГОЛЬНИК
ЛЕЙБНИЦА



Во многие книги по программированию включается задача на построение арифметического треугольника Паскаля. Треугольная таблица, задающая этот треугольник, показана на рисунке 56.

Паскаль Блез (1623—1662) — французский математик, физик и философ. Посвятил ряд работ арифметическим рядам и биномиальным коэффициентам. В «Трактате об арифметическом треугольнике» дал так называемый треугольник Паскаля — таблицу, в которой коэффициенты разложения $(a + b)^n$ для разных n расположены в виде треугольника.

Менее известен другой треугольник, связанный с именем немецкого математика Г. В. Лейбница. Речь идет о гармоническом треугольнике Лейбница (рис. 57).

Лейбниц Готфрид Вильгельм (1646—1716) — немецкий математик, физик и философ. Одновременно с И. Ньютоном, но независимо от него, завершил создание дифференциального и интегрального исчисления. С именем Лейбница в науке связано много открытий и гипотез, которые позже получили признание. Его можно считать идейным вдохновителем современной машинной математики.

Эти два треугольника, образно говоря, «противоположны» друг другу. Так, в треугольнике Паскаля каждое число является суммой своих северо-западного и северо-восточного соседей; в треугольнике же Лейбница каждое число есть сумма своих юго-западного и юго-восточного соседей.

В качестве примера выделены числа: $4 + 6 = 10$ и $\frac{1}{20} + \frac{1}{30} = \frac{1}{12}$.

При вводе элементов треугольника Лейбница на экран или на печать ЭВМ должна их записывать так, как записывают обыкновенные дроби в арифметике. Числитель дроби размещать над строкой дроби, а знаменатель — под строкой. Читатель встречается здесь с необходимостью научить ЭВМ изображать на экране обыкновенные дроби.

Текст программы для построения треугольника Паскаля может быть таким, как приводимый ниже.

```

10 'ТРЕУГОЛЬНИК ПАСКАЛЯ
20 CLS:WIDTH 80
30 INPUT "ВВОДИТЕ КОЛ-ВО СТРОК N(11":N
40 IF N>10 THEN PRINT "ПОВТОРИТЕ ВВОД":GOTO 30
50 DIM A(N,N)
60 PRINT
70 LET R=40
80 LET A(1,1)=1:PRINT TAB(R);A(1,1)
90 FOR I=2 TO N
100 LET R=R-2
110 PRINT TAB(R);
120 FOR K=1 TO I
130 LET A(I,K)=A(I-1,K-1)+A(I-1,K)
140 IF A(I,K)<9 THEN PRINT A(I,K);" ";:GOTO 160
150 PRINT A(I,K);
160 NEXT K
170 PRINT
180 NEXT I
190 END

```

```

      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
 1 7 21 35 35 21 7 1
   1 8 28 56 70 56 28 8 1
    1 9 36 84 126 126 84 36 9 1

```

Главное в этой программе — хорошее размещение треугольника на экране. К сожалению, на экран ПЭВМ можно вывести треугольник с числом строк не более 10 ($N \leq 10$).

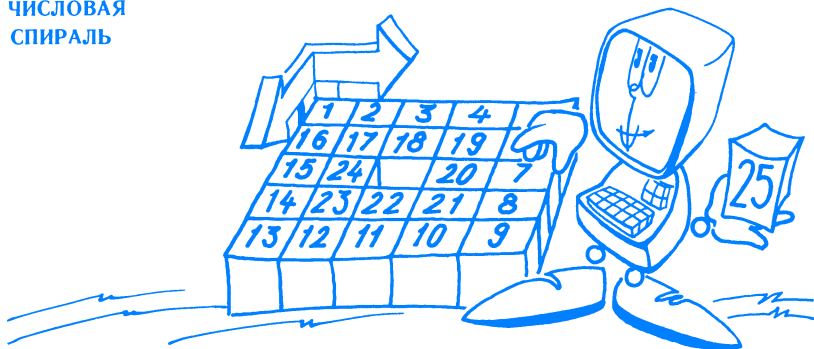
Текст программы для построения треугольника Лейбница сложнее. Школьник, составивший эту программу, продемонстрировал хорошее умение работать с принтером ПЭВМ. Он отлично использовал оператор PRINT в сочетании с оператором TAB. Текст программы приводится ниже.

```

10 REM ТРЕУГОЛЬНИК ЛЕЙБНИЦА
20 CLS:WIDTH 40
30 INPUT "ВВЕДИТЕ КОЛ-ВО СТРОК (<7)";N
40 IF N>6 THEN PRINT "ПОВТОРИТЕ ВВОД."N">6":GOTO 30
50 DIM A(N,N),B(N,N)
60 A(1,1)=1:B(1,1)=1
70 FOR I=2 TO N
80 A(I,1)=1:B(I,1)=I
90 FOR K=2 TO I
100 A(I,K)=1:X=B(I,K-1):Y=B(I-1,K-1)
110 IF X>Y THEN X=X-Y ELSE Y=Y-X
120 IF X<>Y THEN 110
130 M=B(I,K-1):T=B(I-1,K-1):E=(M*Y)/X:X=E/B(I,K-1)
140 Y=E/B(I-1,K-1):M=Y-X:B(I,K)=E/M
150 NEXT K
160 NEXT I
170 R=17:PRINT " "
180 FOR I= 1 TO N
190 FOR K=1 TO I
200 PRINT TAB(R) A(I,K);" ";
210 NEXT K
220 PRINT
230 FOR K=1 TO I
240 IF B(I,K)<10 THEN PRINT TAB(R)" " "-" " ";:GOTO 260
250 PRINT TAB(R) "-" " " ";
260 NEXT K
270 PRINT
280 FOR K=1 TO I
290 IF I<=3 THEN PRINT TAB(R) B(I,K) " ";:GOTO 320
300 IF B(I,K)<10 THEN PRINT TAB(R-1)" "B(I,K);:GOTO 320
310 PRINT TAB(R) B(I,K);
320 NEXT K
330 PRINT
340 R=R-2
350 NEXT I
360 END

```

Итак, в поле зрения школьника попали обыкновенные арифметические дроби. Будет естественно, если мысль читателя устремится к проблеме арифметических дробей и действий с ними на ЭВМ. Нельзя ли ЭВМ научить складывать, вычитать, умножать и делить обыкновенные арифметические дроби? Нельзя ли научить ЭВМ решать арифметические задачи для шестого класса? Такая задача очень заманчива, школьники из Малой академии наук «Искатель» взялись за ее решение и справились «на отлично».



От программиста часто требуется умелое размещение результатов работы ЭВМ на экране. Иногда необходимо вывести числовую таблицу из нескольких строк и столбцов, часто требуется печатание различных списков, например, списков деталей или книг. Предлагаемая задача-шутка на эту тему помогает понять трудности и познакомиться с особенностями решения задач вывода информации на экран ЭВМ.

Пусть дано некоторое число N , натуральное и не простое. Необходимо сформировать квадратную или прямоугольную таблицу, в каждую клетку которой разместить по одному числу: 1, 2, 3, ..., N . Числа следует размещать «по спирали» так, как показано на рисунке 58. На левом рисунке спираль закручивается вправо, а на правом — влево.

Предлагаемая ниже программа составлена семиклассником. Обращаем внимание на то, что в программе ведется контроль за вводимым числом N . Если оно меньше единицы или больше 100 или не является простым, то ЭВМ просит ввести новое число. Программа позволяет получать числовые спирали, закрученные вправо и влево. При формировании таблицы очень разнообразно используется оператор PRINT. Рекомендуем попробовать по тексту программы восстановить и сформулировать алгоритм.

```

10 'ПОСТРОЕНИЕ ЧИСЛОВОЙ СПИРАЛИ
20 PRINT "УКАЖИТЕ ВИД СПИРАЛИ"
30 INPUT "1-ПРАВАЯ, 2-ЛЕВАЯ";H
40 INPUT "ВВЕДИТЕ МАХ-ЧИСЛО СПИРАЛИ P,1(P<100)";P
45 PRINT "ЧИСЛО P НЕ ДОЛЖНО БЫТЬ ПРОСТЫМ"
60 IF P<1 THEN PRINT "ПОВТОРИТЕ ВВОД (P<1)":GOTO 40
70 IF P>100 THEN PRINT "ПОВТОРИТЕ ВВОД (P>100)":GOTO 40
80 LET R=INT(SQR(P)):Q=0:C=1:IF P=1 THEN 130
90 IF R<2 THEN 120
100 FOR L=2 TO R:IF P/L=INT(P/L) THEN 130
110 NEXT L
120 PRINT "ПОВТОРИТЕ ВВОД (;P;)-ПРОСТОЕ":GOTO 40
130 IF P/R=INT(P/R) THEN N=P/R ELSE R=R+1:GOTO 130
140 IF R)=10 THEN 160

```

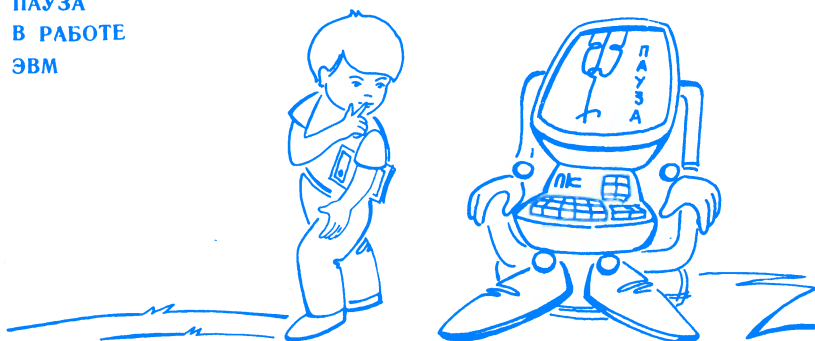
```

150 IF R<N THEN SWAP R,N
160 LET F=N:M=R
170 DIM A(R,N)
180 IF H=2 THEN 290
190 FOR I=1 TO N:Q=Q+1:K=I:A(1,K)=Q:NEXT I
200 LET R=R-1:IF R=0 THEN 390
210 FOR I=1 TO R:Q=Q+1:S=C+I:A(S,K)=Q:NEXT I
220 LET N=N-1:IF N=0 THEN 390
230 FOR I=1 TO N:Q=Q+1:K=F+1-C-I:A(S,K)=Q:NEXT I
240 LET R=R-1:IF R=0 THEN 390
250 FOR I=1 TO R:Q=Q+1:S=M+1-C-I:A(S,K)=Q:NEXT I
260 LET N=N-1:IF N=0 THEN 390
270 FOR I=1 TO N:Q=Q+1:K=C+I:A(S,K)=Q:NEXT I
280 LET C=C+1:GOTO 200
290 FOR I=1 TO R:Q=Q+1:S=I:A(S,1)=Q:NEXT I
300 LET N=N-1:IF N=0 THEN 390
310 FOR I=1 TO N:Q=Q+1:K=C+I:A(S,K)=Q:NEXT I
320 LET R=R-1:IF R=0 THEN 390
330 FOR I=1 TO R:Q=Q+1:S=M+1-C-I:A(S,K)=Q:NEXT I
340 LET N=N-1:IF N=0 THEN 390
350 FOR I=1 TO N:Q=Q+1:K=F+1-C-I:A(S,K)=Q:NEXT I
360 LET R=R-1:IF R=0 THEN 390
370 FOR I=1 TO R:Q=Q+1:S=C+I:A(S,K)=Q:NEXT I
380 LET C=C+1:GOTO 300
390 FOR S=1 TO M
400 FOR K=1 TO F
410 IF A(S,K) < 0 THEN PRINT A(S,K) " ";:GOTO 430
420 PRINT A(S,K);
430 NEXT K
440 PRINT
450 NEXT S
460 END

```

Может быть кто-то из читателей сумеет найти иной подход.

ПАУЗА
В РАБОТЕ
ЭВМ



Иногда в работе ЭВМ необходимо сделать паузу на несколько секунд. Как это сделать? Вот что предлагают школьники: они используют оператор цикла, в котором «тело цикла» не содержит ни одного оператора. Оператор цикла ведет только наращивание значений параметра цикла.

Программа может быть такой, как приведенная ниже:


```

5 REM ПАУЗА В РАБОТЕ
15 INPUT B
25 LET TIME=0
35 FOR I=1 TO B
45 NEXT I
55 PRINT "ПАУЗА ДЛИЛАСЬ";TIME/50;"СЕК"
65 END

```

Располагая такой программой, в которую встроен «секундомер» (таймер), можно провести исследование различных циклов, которые используются для образования пауз. Наиболее подходящий цикл вставляется в те места программы, где Вы хотели бы приостановить ее исполнение.

Еще одно предложение. В нем используется переменная TIME, эта переменная принимает значения автоматически. Каждую секунду ее значение возрастает на 50 единиц. Предположим, что у Вас в программе отведено время 20 секунд на принятие решения человеком, играющим с ЭВМ в какую-то игру. Если человек свой ход за 20 секунд сделать не успел, то ему засчитывается поражение. Как можно организовать контроль за временем размышления человека?

Вариант проекта программы может быть таким:

```

10 '...
20 '...
30 PRINT "ВАМ ХОДИТЬ, СЕКУНДОМЕР РАБОТАЕТ"
40 LET TIME=0
50 INPUT "S=";S
60 IF TIME/50<20 THEN ....
70 PRINT "ВРЕМЯ ИСТЕКЛО"
80 '...

```

Еще раз подчеркнем, что в первом варианте организации паузы секундомер не нужен, достаточно только применить оператор цикла. Второй метод пригоден там, где переменная TIME может использоваться в программе. Это возможно, к сожалению, не на всех типах ЭВМ.

Организуя паузы с помощью цикла FOR — NEXT, можно управлять их длительностью. С этой целью заключительное значение параметра цикла (N) делают переменным.

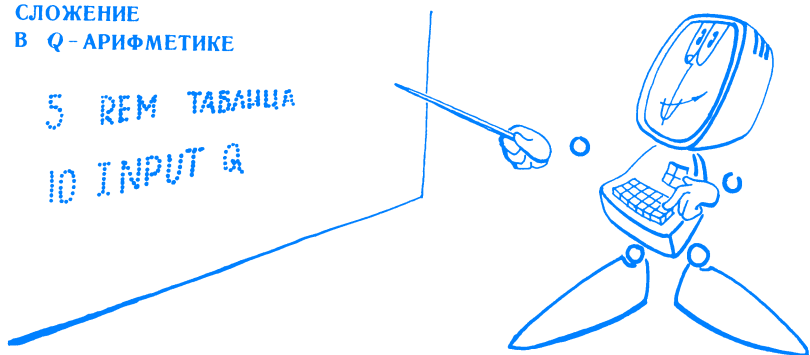
Пример такого цикла-паузы можно записать так:

```
FOR I=1 TO N : NEXT
```

Паузы различной длительности часто используют в обучающих и тренирующих программах. Сначала обучаемому дают на обдумывание ответа больше времени, а по мере его продвижения вперед это время уменьшают.

Используют паузы и в тех мультфильмах, которые в настоящее время художники создают на экранах ЭВМ.

СЛОЖЕНИЕ
В Q -АРИФМЕТИКЕ



Арифметика в любой позиционной системе счисления с натуральным основанием q основана на таблицах сложения цифр. На рисунке 59 для напоминания изображены таблицы сложения цифр в троичной системе счисления и в системе счисления с основанием $q=5$.

Необходимо разработать программу, действуя по которой ЭВМ сумеет составить и напечатать таблицу сложения цифр для системы счисления с любым основанием $q \in N$ и $q < 10$.

Вот какую программу разработали школьники.

```
5 REM ТАБЛИЦА СЛОЖЕНИЯ В Q
10 INPUT Q
15 FOR I=0 TO Q-1
20 FOR J=0 TO Q-1
25 LET S=I+J
30 IF S<Q THEN 40
35 LET S=S+10-Q
40 PRINT S;
45 NEXT J
50 PRINT
55 NEXT I
60 END
```

Ниже приводятся две распечатки таблиц сложения, составленных машиной для систем с основанием $q=6$ и $q=8$ (табл. 22—23).

Таблица 22

$q=6$

0	1	2	3	4	5
1	2	3	4	5	10
2	3	4	5	10	11
3	4	5	10	11	12
4	5	10	11	12	13
5	10	11	12	13	14

Таблица 23

$$q=8$$

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	10
2	3	4	5	6	7	10	11
3	4	5	6	7	10	11	12
4	5	6	7	10	11	12	13
5	6	7	10	11	12	13	14
6	7	10	11	12	13	14	15
7	10	11	12	13	14	15	16

Более сложный случай, если основание системы $q > 10$. В этом случае таблица должна включать «необычные цифры». Общепринято, например, в системе с основанием $q=16$ использовать цифры $A_{16}=10_{10}$; $B_{16}=11_{10}$; $C_{16}=12_{10}$; $D_{16}=13_{10}$; $E_{16}=14_{10}$ и $F_{16}=15_{10}$.

Но и в этом случае составление программы вполне по силам школьникам. Нижеприведенная программа составлена ребятами.

```

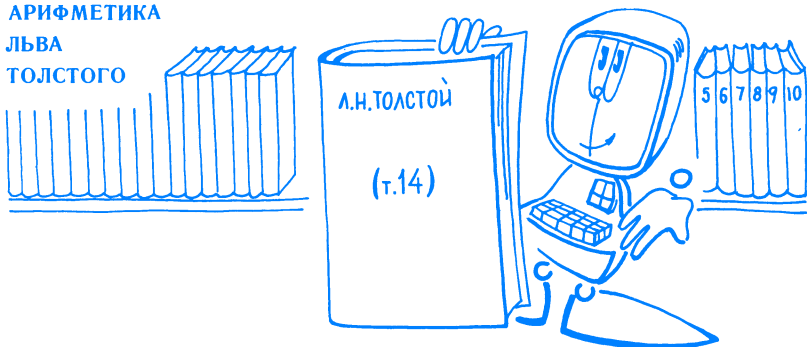
10 REM СУММАТОР В СИСТЕМЕ С ОСНОВАНИЕМ Q=12
20 INPUT "КОЛ-ВО РАЗРЯДОВ В НАИБОЛЬШЕМ СЛАГАЕМОМ";N
30 DIM S1(N+1),S2(N+1),S3(N+2)
40 FOR I=1 TO N
50 PRINT "ВВОД ЦИФРЫ ";I;"-ОГО РАЗРЯДА S1";
60 INPUT K$
70 LET S1(I)=ASC(K$)-48
80 IF S1(I)=17 THEN S1(I)=10
90 IF S1(I)=18 THEN S1(I)=11
100 NEXT I
110 FOR I=1 TO N
120 PRINT "ВВОД ЦИФРЫ ";I;"-ОГО РАЗРЯДА S2";
130 INPUT K$
140 LET S2(I)=ASC(K$)-48
150 IF S2(I)=17 THEN S2(I)=10
160 IF S2(I)=18 THEN S2(I)=11
170 NEXT I
180 CLS:LOCATE 3,10:PRINT "СУММА = ";
190 FOR I=1 TO N+1
200 LET S3(I)=S3(I)+S1(I)+S2(I)
210 IF S3(I) > 12 THEN GOTO 250
220 LET S3(I)=S3(I)-12:S3(I+1)=S3(I+1)+1:GOTO 250
230 NEXT I
240 END
250 LET Z = (S3(I)+48)
260 IF Z=58 THEN Z=65
270 IF Z=59 THEN Z=66
280 REM ВЫВОД РЕЗУЛЬТАТА
290 LOCATE 3,(20+N-1):PRINT CHR$(Z);
300 GOTO 230

```

АРИФМЕТИКА

ЛЬВА

ТОЛСТОГО



В 1870 г. Лев Толстой преподавал арифметику крестьянским детям. Свой педагогический опыт он описал в статье «Арифметика». Эту статью можно прочесть в 14-м томе полного собрания сочинений Л. Толстого. Обращает внимание, что Л. Толстой много времени уделял изучению с учениками системы счисления с основанием 12 и римской нумерации. Эти системы счета в то время широко применялись на практике. Сегодня эти системы утратили свое значение, но каждый грамотный человек должен уметь читать числа, записанные в римской нумерации, и уметь десятичные числа преобразовывать в числа, записанные в римской нумерации.

Напомним, что для записи чисел в римской нумерации используются цифры: $M=1000$, $D=500$, $C=100$, $L=50$, $X=10$, $V=5$, $I=1$. Важно помнить, что $CM=900$, $CD=400$, $XC=90$, $XL=40$, $IX=9$ и $IV=4$.

Примеры чисел, записанных по правилам римской нумерации: $MCMXXXII=1932$, $CCXXI=221$.

Ниже приведена программа, которая поручает ЭВМ выступить в роли дешифратора, заменяющего данное десятичное число равным ему числом римской нумерации.

Работа программы проста, схематически она задается алгоритмом, представленным на рисунке 60. Здесь обращаем внимание на оператор `100 RESTORE 30`. С его помощью ЭВМ восстанавливает содержимое хранилища `DATA` и тем самым дает возможность осуществить замену следующего десятичного числа равным ему числом римской нумерации.

```
10 REM АЛГОРИТМ ЗАМЕНЫ ЧИСЕЛ ДЕСЯТИЧНОЙ СИСТЕМЫ
20 REM РАВНЫМИ ИМ ЧИСЛАМИ РИМСКОЙ НУМЕРАЦИИ
30 DATA 1000, M, 900, CM, 500, D, 400, CD, 100, C, 90, XC
40 DATA 50, L, 40, XL, 10, X, 9, IX, 5, V, 4, IV, 1, I
50 INPUT N
60 READ A, A$
70 IF A>N THEN 90
80 PRINT A$;:LET N=N-A:GOTO 70
90 IF N>0 GOTO 60
100 RESTORE 30:RESTORE 40
110 END
```

Оператор PRINT A\$ означает, что один за другим выводятся символы римской нумерации — римские цифры.

Программа для решения обратной задачи более интересна, и прежде всего тем, что в ней используются операторы для обработки символьных величин. Речь идет об операторах LEFT \$ и RIGHT \$. Суть каждого из них в том, что они работают не с числами, а со словами. Ранее уже подчеркивалось, что имена символьных переменных должны заканчиваться знаком \$, например, A\$, B\$ или C\$ — это имена символьных переменных. Выше было показано, что оператор LET может использоваться и для присвоения значений символьным переменным.

Примеры

```
5 LET A$="ПАРОВОЗ"  
15 LET N$="MLVI"  
25 LET R$="НЕЗАБУДКА"
```

Операторы LEFT \$ и RIGHT \$ задают преобразование слова, являющегося значением символьной переменной. Указание N RIGHT \$(A\$,N) означает, что ЭВМ должна взять значение символьной переменной A\$ и стереть все символы слева направо, оставив N крайних справа символов обрабатываемого слова.

Пример

Пусть A\$=«пароход», то после выполнения оператора 5 RIGHT \$(A\$, 3) новым значением A\$ будет слово «ход».

Указание N LEFT \$(A\$,N) означает, что ЭВМ в слове, являющемся значением переменной A\$, стирает все символы кроме первых N символов слева.

Пример

Пусть A\$=„MLVII“, то после выполнения оператора 10 LEFT \$(A\$,3) значением символьной переменной A\$ будет слово „MLV“.

Оператор LEN(A\$), примененный к значению переменной A\$, вычисляет число символов в том слове, к которому применяется.

Пример

Пусть A\$=«незабудка», то после выполнения оператора 10 LET R=LEN(A\$) значение R будет обязательно числом, в данном случае R=9, так как в слове «незабудка» девять букв.

В программе замены чисел римской нумерации равными им десятичными числами необходимо обратить внимание на два условных оператора: 40 и 60. В операторе 40 IF A\$<>LEFT \$(N\$,LEN(A\$)) выясняется, не совпадают ли два слова: A\$ и то слово, что остается после применения к слову N\$ оператора: LEFT \$(N\$,LEN(A\$)).

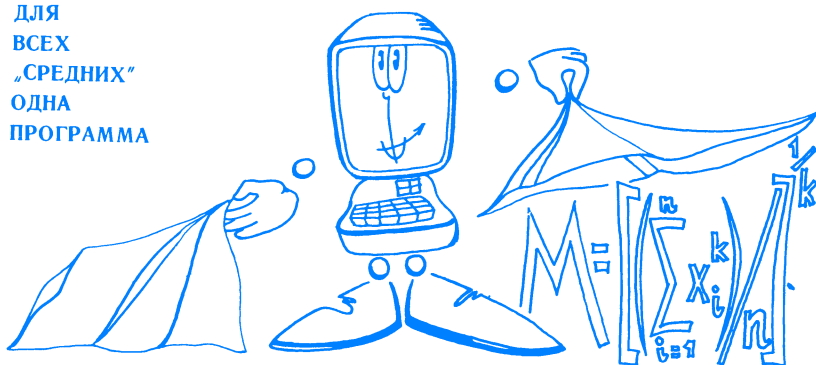
В операторе 60 ведется сравнение значения переменной N\$ с пустым словом: пока N\$ не пусто, управление передается оператору 30. Рекомендуется вычертить схему алгоритма, задаваемого этой программой.

```

5 REM АЛГОРИТМ ЗАМЕНЫ ЧИСЕЛ РИМСКОЙ НУМЕРАЦИИ
  РАВНЫМИ ИМ ДЕСЯТИЧНЫМИ АРАБСКИМИ ЧИСЛАМИ
10 DATA 1000,M,900,CM,500,D,400,CD,100,C,90,XC,
  50,L,40,XL,10,X,9,IX,5,V,4,IV,1,I
20 INPUT N$
30 READ A,A$
40 IF A$<>LEFT$(N$,LEN(A$)) THEN GO TO 60
50 LET NS=RIGHT$(N$,LEN(N$)-LEN(A$)):LET N=N+A$
55 GO TO 40
60 IF N$ > ' ' GO TO 30
70 PRINT 'ВЫВОД ИСКОМОГО ЧИСЛА' N
80 RESTORE 5
90 END

```

ДЛЯ
ВСЕХ
„СРЕДНИХ“
ОДНА
ПРОГРАММА



Пусть величина x имеет n значений: x_1, x_2, \dots, x_n , тогда: *среднее арифметическое* вычисляется по формуле:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{x_1 + x_2 + \dots + x_n}{n}, \quad (1)$$

среднее квадратичное — по формуле:

$$\bar{x}_q = \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \dots + x_n^2}{n}}, \quad (2)$$

среднее кубическое — по формуле:

$$\bar{x}_k = \sqrt[3]{\frac{1}{n} \sum_{i=1}^n x_i^3} = \sqrt[3]{\frac{x_1^3 + x_2^3 + \dots + x_n^3}{n}}, \quad (3)$$

среднее гармоническое — по формуле:

$$\overline{x_n} = n : \sum_{i=1}^n \left(\frac{1}{x_i} \right) = n : \left(\frac{1}{x_1} + \frac{1}{x_2} + \dots + \frac{1}{x_n} \right). \quad (4)$$

Для каждого из четырех типов средних своя расчетная формула. Нельзя ли найти общую формулу? Можно, вот она:

$$M = \left(\sum_{i=1}^n x_i^k / n \right)^{\frac{1}{k}}. \quad (5)$$

При $k=1$, $k=2$ и $k=3$ получаются формулы (1), (2), (3). Если $k=-1$, то из формулы (5) получается формула для вычисления среднего гармонического.

Формулу (5) можно положить в основу программы для вычисления практически всех часто используемых типов средних. Программа может быть такой:

```
10 'НАХОЖДЕНИЕ СРЕДНИХ
20 PRINT "1. СРЕДНЕЕ АРИФМЕТ."
30 PRINT "2. СРЕДНЕЕ КВАДРАТ."
40 PRINT "3. СРЕДНЕЕ КУБ."
50 PRINT "4. СРЕДНЕЕ ГАРМОН."
60 INPUT "ВВЕДИТЕ К ИСКОМОГО СРЕДН. "; K
70 IF K=4 THEN LET K=-1
80 INPUT "ВВЕДИТЕ N - КОЛ-ВО ДАННЫХ"; N
90 PRINT "ВВОДИТЕ ДАННЫЕ"
100 FOR I=1 TO N
110 INPUT X(I):NEXT I
120 LET S=0
130 FOR I=1 TO N
140 LET S=S+X(I)^K:NEXT I
150 LET S=S/N
160 LET M=S^(1/K)
170 PRINT "ИСКОМОЕ СРЕДНЕЕ РАВНО "; M
180 END
```

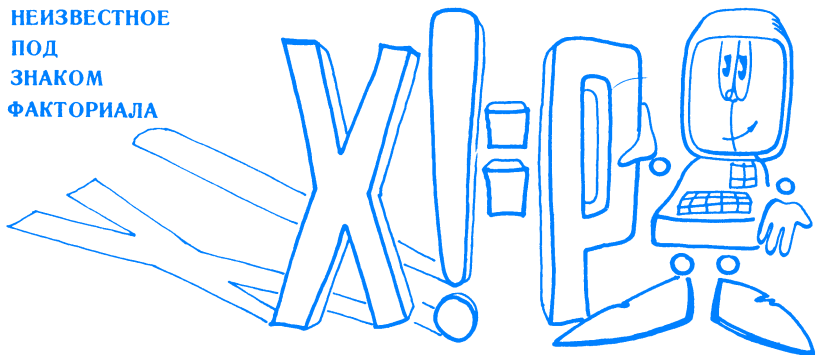
Это хороший пример того, как достаточно общая формула, известная математикам, оказалась положенной в основу очень простой программы.

Умение программировать процессы вычислений с применением операций извлечения квадратного и кубического корня читатель может закрепить, составив программу для проверки неравенства:

$$\sqrt{\frac{a^2 + b^2 + c^2 + d^2}{4}} \geq \sqrt[3]{\frac{abc + abd + acd + bcd}{4}}$$

где числа a , b , c и d есть числа натуральные. Проверку провести, используя первые 10 значений каждой из переменных a , b , c и d .

НЕИЗВЕСТНОЕ
ПОД
ЗНАКОМ
ФАКТОРИАЛА



Как решить уравнение $x! = p$? Если обратиться к элементарной математике, то ответ найти непросто. Дело в том, что, решая уравнения показательные, линейные, квадратные или логарифмические, школьники обращаются к обратным функциям и решение уравнения, например показательного $2^{x-1} = 16$, сводят к вычислению значения обратной (логарифмической) функции: $x = \log_2 16 + 1$. Решение квадратного уравнения $(x-1)^2 = 19$ сводят к вычислению значения обратной функции: $x = \sqrt{19} + 1$.

Множество типов уравнений можно решить, используя связь между функцией, под знаком которой находится неизвестная, и функцией, ей обратной. К сожалению, если вычисление факториала мы будем рассматривать как функцию $y = x!$, то обратной функции для $y = x!$ школьники не знают.

Проторенным в математике путем решить уравнение не удастся. Решение, однако, найти несложно, если алгоритм решения представить в виде текста программы. Здесь мы еще раз встречаемся с тем, что привычная для школьника форма задания алгоритма в виде формулы в ряде случаев совершенно неприменима. В действие вступает алгоритм, записанный в виде текста программы.

В данном примере речь идет о том, что мы имеем функцию $y = x!$ и имеем алгоритм вычисления ее значений при различных $x \in N$. Если требуется решить уравнение $x! = p$, то в основе алгоритма-решения будет лежать основной алгоритм, дополненный несколькими операторами.

Приведем программу вычисления $N!$.

```
10 'НАХОЖДЕНИЕ N!  
20 INPUT N  
30 LET P=1  
40 FOR I=1 TO N  
50 P=P*I:NEXT I  
60 PRINT P  
70 END
```

Если нам требуется решить уравнение $x! = p$ (рис. 61), то программа может быть такой:


```

10 * РЕШЕНИЕ УРАВ-НЕНИЯ  $x!=p$ 
20 INPUT "ВВОД P";P
30 LET X=1
40 LET N=1
50 LET N=N*X
60 IF N=P THEN GOTO 100
70 IF N>P THEN GOTO 110
80 LET X=X+1
90 GOTO 50
100 PRINT "X=";X :GOTO 120
110 PRINT "X С НЕДОСТАТКОМ=";X-1
120 END

```

Наряду с уравнением $x!=p$, где $x \in N$, можно рассматривать уравнения: $(2x-1)!!=p$ и $(2x)!!=p$. В первом из них речь идет о произведении вида: $1 \cdot 3 \cdot 5 \cdot 7 \cdot \dots \cdot (2x-1)$, а во втором о произведении вида: $2 \cdot 4 \cdot 6 \cdot 8 \cdot \dots \cdot (2x)$.

Иначе говоря, речь идет о решении уравнений

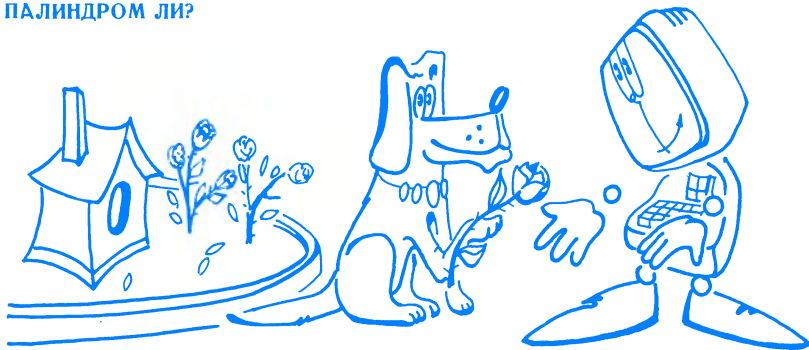
$$1 \cdot 3 \cdot 5 \cdot \dots \cdot (2x-1) = p, \text{ где } x \in N$$

и

$$2 \cdot 4 \cdot 6 \cdot \dots \cdot (2x) = p, \text{ где } x \in N.$$

О том, какие поправки следует внести в программу решения уравнения $x!=p$, читатель догадается сам.

ПАЛИНДРОМ ЛИ?



Среди шуточных задач популярна задача поиска слов — *палиндромов*, то есть слов, которые читаются одинаково слева направо и справа налево (рис. 62). Это по силам и ЭВМ.

Задача. Среди N слов найти все слова, являющиеся палиндромами.

Решение этой задачи значительно упростится, если читатель возьмет на вооружение еще одну функцию, используемую при работе со словами, с символьными величинами. Речь идет о функции **MID\$**. С помощью функции **MID\$** можно в данном слове выбрать k символов, начиная с символа, занимающего 1-е место при рассмотрении слова слева направо.

Форма функции **MID\$**; **MID\$(Y\$,I,K)** или **MID\$(Y\$,I)**

(здесь опущено k) это означает, что значением функции MID\$ будет все слово с 1-й буквы.

Примеры

```
10 LET Y$="НОСОК"  
20 LET X$=MID$(Y$,3,3)  
30 PRINT X$  
40 END
```

ЭВМ выдаст на печать значение X\$, то есть слово: СОК. Этот же результат будет получен, если метод 10 будет записан так: 10 MID\$(Y\$,3).

Решая задачу, ЭВМ сначала запросит у человека все N исследуемых слов и образует из них массив R\$, массив не чисел, а слов. Затем она будет вести анализ выбираемых одно за другим слов. Каждое слово, оказавшееся палиндромом, будет напечатано.

Выяснение того, является ли выбранное слово R\$(I) палиндромом, ведется буквенно, сравниваются буквы, равноотстоящие от концов данного слова. Если все сравниваемые буквы окажутся совпадающими, то слово является палиндромом.

Текст программы приводится ниже.

```
10 REM "ПОИСК ПАЛИНДРОМОВ"  
20 REM "ФОРМ. МАССИВА ИСХОДНЫХ СЛОВ"  
30 INPUT "КОЛ-ВО СЛОВ";N  
40 DIM R$(N)  
50 FOR I=1 TO N  
60 INPUT R$(I)  
70 NEXT I  
80 FOR I=1 TO N  
90 LET L=LEN(R$(I))  
100 IF L=0 THEN GOTO 180  
110 LET K=INT(L/2)  
120 FOR G=1 TO K  
130 LET X$= MID$(R$(I),G,1)  
140 LET Y$= MID$(R$(I),L-G+1,1)  
150 IF X$<>Y$ THEN GOTO 180  
170 NEXT G  
175 PRINT R$(I)  
180 NEXT I  
190 END
```

Как и другие задачи, задачу поиска палиндромов можно решить и другим способом.

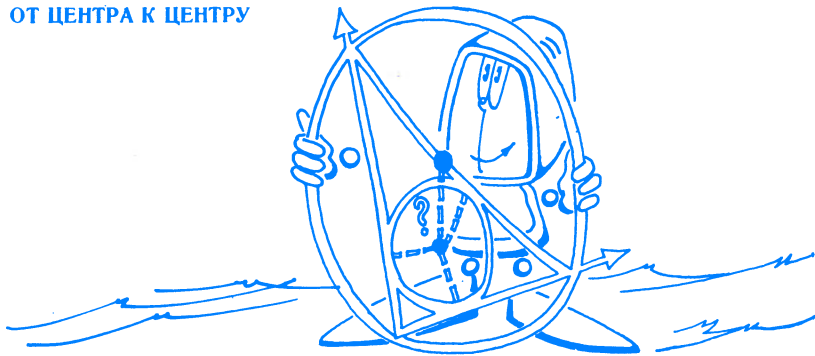
Предлагается такой подход: каждое исследуемое слово сначала записывается в обратном порядке, а затем проверяется, не совпадают ли исходное и полученное слова. Программа образования слов, записанных в обратном порядке, может быть такой:

```
10 'ИЗМЕНЕНИЕ ПОРЯДКА БУКВ  
20 'В СЛОВЕ НА ОБРАТНЫЙ  
30 INPUT R$  
40 LET L=LEN(R$)  
50 FOR I=L TO 1 STEP-1  
60 LET A$=A$+MID$(R$,I,1)  
70 NEXT I  
80 PRINT A$  
90 END
```

Может программа, в которой используется новый подход, окажется более быстрой? Создание программы, включающей переменную порядка букв в исследуемом слове, предлагается создать читателю самостоятельно.

В программе поиска палиндромов эффективно использовались функции MID\$ и LEN, а также операция сцепления строк ($A\$ + B\$$), называемая *конкатенацией*.

ОТ ЦЕНТРА К ЦЕНТРУ



Выше уже подчеркивалась роль метода координат, который связывает в единое целое алгебру и геометрию. Современные математики очень часто, обдумывая задачу, создают ее геометрическую модель. В случае затруднений ищут идею решения, привлекая алгебраические соображения. Владеть методом координат учится сегодня и школьник. Рассмотрим пример задачи, алгебраическая модель которой хорошо обрабатывается ЭВМ.

Задача. Дан прямоугольный треугольник с катетами длиной a и b . Найти расстояние от центра вписанной в треугольник окружности до центра описанной около него окружности.

Решение. Используем метод координат. Введем прямоугольную систему координат с началом в вершине прямого угла данного треугольника (см. рис. 63). Тогда $A(0; b)$; $B(a; 0)$ и центр описанной окружности $M(a/2; b/2)$. Обозначим координаты центра вписанной окружности $N(x; y)$. Обозначим точки касания вписанной окружности сторон треугольника N_1, N_2 и N_3 .

Используя равенство $AN_2 + BN_1 = AB$, получим уравнение:

$$(a - x) + (b - x) = \sqrt{a^2 + b^2},$$

или

$$x = 0,5(a + b) - \sqrt{a^2 + b^2}.$$

Пользуясь формулой для вычисления расстояния между двумя точками $M(a/2; b/2)$ и $N(x; y)$, получим:

$$MN = \sqrt{\left(\frac{a}{2} - \frac{a+b-\sqrt{a^2+b^2}}{2}\right)^2 + \left(\frac{b}{2} - \frac{a+b-\sqrt{a^2+b^2}}{2}\right)^2} =$$

$$=0,5 \sqrt{(\sqrt{a^2+b^2}-b)^2 + (\sqrt{a^2+b^2}-a)^2}.$$

Задача всегда имеет единственное решение при положительных значениях a и b .

Текст программы может быть таким:

```
10 INPUT "ВВОД ДЛИН КАТЕТОВ А,В";А,В
20 PRINT "НАХОЖДЕНИЕ ИСКОМОГО РАССТОЯНИЯ МЕЖДУ ЦЕНТРАМИ"
30 D=(SQR(SQR(A^2+B^2)-B)^2+(SQR(A^2+B^2)-A)^2)/2
40 PRINT D
50 COLOR 1,15,8:SCREEN 2
60 LINE (0,0)-(0,B),6
70 LINE (0,0)-(A,0),8
80 LINE (A,0)-(0,B),5.
90 GOTO 90
```

ЛИНЕЙКА
НА ЭКРАНЕ



Работа с ЭВМ осуществляется в разнообразных формах. Человек может поручить машине обработку данных и не вмешиваться в ее работу до конца решения задачи. Может быть так, что человек намерен контролировать процесс работы ЭВМ на определенных этапах решения задачи. В этом случае ЭВМ, дойдя до «промежуточного финиша», выдает сигнал оператору и он анализирует ситуацию. Если все идет, как намечено, ЭВМ получает команду продолжать. Может быть так, что человек все время находится в «диалоге» с ЭВМ. Слово диалог мы взяли в кавычки, чтобы подчеркнуть, что под диалогом мы будем понимать нечто большее, чем двустороннюю беседу, для нас диалог — двустороннее сотрудничество. Такое взаимодействие может включать обмен информацией между человеком и машиной, приостановку решения основной задачи для краткого анализа варианта решения. ЭВМ по поручению человека, выдаваемому «с пульта» вручную, может проводить линии или изображать окружности. Иначе говоря, ЭВМ может предоставить человеку быстрый карандаш. Карандаш при этом будет в твердой уверенной руке. Изображаемые линии будут аккуратно и точно вычерчены. Режим диалогового взаимодействия иногда называют *интерактивным*.

Ниже приводится текст программы «Линейка».

```
10 ' ЛИНЕЙКА
20 DIM LI(2,2,20)
30 CLS
40 SCREEN 3
50 Y1=10:Y2=10:X1=1:X2=400:CK=1:N=0
60 '
70 LINE (X1,Y1)-(X2,Y2),2
80 IF N=0 THEN GOTO 120
90 FOR I=1 TO N
100 LINE (LI(1,1,I),LI(1,2,I))-(LI(2,1,I),LI(2,2,I)),5
110 NEXT I
120 '
130 S$=INKEY$
140 IF S$="Z" THEN LINE (X1,Y1)-(X2,Y2),0:Y1=Y1+CK
150 IF S$="X" THEN LINE (X1,Y1)-(X2,Y2),0:Y1=Y1-CK
160 IF S$="C" THEN LINE (X1,Y1)-(X2,Y2),0:Y2=Y2+CK
170 IF S$="V" THEN LINE (X1,Y1)-(X2,Y2),0:Y2=Y2-CK
180 IF S$="S" THEN LINE (X1,Y1)-(X2,Y2),0:X1=X1-CK
190 IF S$="D" THEN LINE (X1,Y1)-(X2,Y2),0:X2=X2+CK
200 IF S$="F" THEN LINE (X1,Y1)-(X2,Y2),0:X2=X2-CK
210 IF S$="A" THEN LINE (X1,Y1)-(X2,Y2),0:X1=X1+CK
220 IF S$="G" THEN INPUT "CK=";CK
230 IF S$="N" THEN GOTO 260
240 IF S$="B" THEN GOTO 290
250 GOTO 70
260 N=N+1:LI(1,1,N)=X1:LI(1,2,N)=Y1:LI(2,1,N)=X2:LI(2,2,N)=Y2
270 ' ЗАПОМИНАНИЕ НОВОЙ ЛИНИИ
280 GOTO 70
290 LOCATE 21,10 :PRINT "X1=";X1;"Y1=";Y1;
300 LOCATE 20,10 :PRINT "X2=";X2;"Y2=";Y2;
310 END
```

Программа достаточно сложная, потому ее освоение рекомендуется осуществить за пультом ЭВМ.

В начале работы линейка располагается на экране у верхней кромки. Линейка в этот момент есть отрезок с концами в точках $A(x_1; y_1)$ и $B(x_2; y_2)$. Начальные значения: $x_1=1$, $y_1=10$, $x_2=400$, $y_2=10$. Для изображения отрезка AB используется оператор $\text{LINE } (X1,Y1) - (X2,Y2), C$, где C — цвет отрезка.

Вызвать движение одного из концов линейки — это значит изменить его координаты, дать приращение (СК). Изменять каждую из четырех координат можно отдельно вручную, нажимая на клавиши с литерами: Z, X, C, V, S, D, F, A. Нажатие каждой вызовет изменение соответствующей координаты на величину СК (см. строки 140—210). Величину СК можно изменять, достаточно нажать клавишу G и ввести новое значение СК вручную (см. строку 220). Величину СК можно образно назвать *скоростью движения* линейки по экрану: если СК большое, то и координата получает большое приращение, конец линейки «скачет» на большую величину. Наименьшее значение $СК=1$, в этом случае конец линейки будет смещаться на один пиксел (точку) на экране.

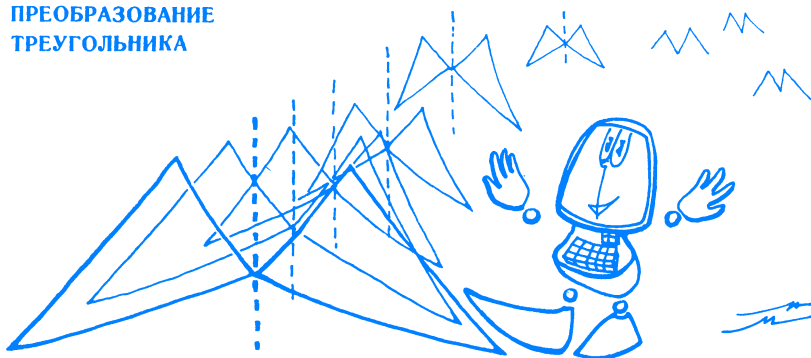
Если какую-то линию следует оставить на экране, а линейку после этого переместить в другое место, то и это можно сде-

лать (см. строки 290—300). Приведенной программой предусматривается «вычерчивание» всего 20 отрезков, «координаты» их запоминаются в массиве LI(2, 2, 20).

В этой программе важную роль играет операция INKEY\$, которая код конкретной клавиши присваивает переменной S\$. Научиться применять оператор INKEY\$ очень полезно. Многие диалоги строятся с его использованием.

Конечно, число удобных графических средств, которые ЭВМ может предоставить человеку для работы в интерактивном режиме, может быть увеличено. Человек может, например, получить возможность быстро размещать на экране окружности разного радиуса и цвета или проводить пунктирные отрезки. Совокупность таких быстро действующих средств образует *графический редактор*. На практике используется множество удивительных графических редакторов.

ПРЕОБРАЗОВАНИЕ ТРЕУГОЛЬНИКА



Ниже рассматривается несколько задач-миниатюр, задач-шуток.

Дан остроугольный треугольник ABC , основание треугольника CA параллельно оси OX . Программа построения треугольника приведена ниже.

```
5 REM ПОСТРОЕНИЕ ТРЕУГОЛЬНИКА
10 INPUT XA, YA, XB, YB, XC, YC, Q
15 LINE (XA, YA) - (XB, YB), Q
20 LINE (XA, YA) - (XC, YC), Q
25 LINE (XB, YB) - (XC, YC), Q
30 GOTO 30
```

Предлагаем внести в данную программу такие дополнения, чтобы на экране вместе с данным треугольником появился:

- 1) равновеликий данному прямоугольный треугольник с тем же основанием;
- 2) тупоугольный треугольник, равновеликий данному, с тем же основанием;

- 3) треугольник, симметричный данному относительно оси, проходящей через вершину A перпендикулярно основанию;
- 4) треугольник, симметричный данному относительно оси OX ;
- 5) треугольник, симметричный данному относительно начала системы координат;
- 6) треугольник, симметричный данному относительно основания AC ;
- 7) треугольник, симметричный данному относительно вершины A ;
- 8) треугольник, гомотетичный данному (центр гомотетии в точке A), имеющий в K раз большую площадь;
- 9) треугольник, равный данному, с вершиной A , лежащей в точке $A(XA1;YA1)$, стороны нового и старого треугольников параллельны;
- 10) параллелограмм, полученный достройкой данного треугольника так, что сторона BC оказалась диагональю параллелограмма.

На рис. 64—73 показаны все требуемые построения. Дополнения к тексту приводим ниже.

1. Чтобы данный треугольник ABC преобразовать в равновеликий ему прямоугольный треугольник с тем же основанием (рис. 64), достаточно ввести в рассмотрение точку $B1(XA;YB)$. Приводим дополняющие программу строки:

```
30 LET XB1=XB:LET YB1=YB
35 LINE (XA,YA)-(XB1,YB1),R
40 LINE (XB,YB)-(XB1,YB1),R
45 GOTO 45
```

2. Для решения этой задачи достаточно ввести точку $B1$ с абсциссой $XB1 < XA$ (рис. 65).

3. Главное — в вычислении координат точек $B1$ и $C1$ (см. рис. 66). Для их вычисления достаточно использовать операторы:

```
30 LET XB1=XA-XB:LET XC1=XA-XC
```

4. В этом случае заново вычисляются только ординаты $A1, B1, C1$ (рис. 67)..

```
30 LET YA1=-XA:LET YB1=-YB:LET YC1=-YC
```

5. В этом случае для построения треугольника, симметричного данному относительно начала системы координат, достаточно вычислить координаты вершин создаваемого треугольника (рис. 68).

```
30 LET XA1=-XA:LET YA1=-YA:LET XB1=-XB
35 LET YB1=-YB:LET XC1=-XC:LET YC1=-YC
```

6. В этом случае (рис. 69) следует вычислить:

```
30 LET YB1=YA-YB
```

7. В этом случае (рис. 70) координаты искомых вершин B1 и C1 можно вычислить, введя операторы:

```
30 LET XB1=XA-XB:LET YB1=YA-YB
35 LET XC1=XA-XC:LET YC1=YA-YC
```

8. В программу (рис. 71) вводятся операторы:

```
30 LET XB1=SQR(K)*(XB-XA)+XA
35 LET YB1=SQR(K)*(YB-YA)+YA
40 LET XC1=SQR(K)*(XC-XA)+XA
```

9. Для решения задачи (рис. 72) достаточно координаты точек A1, B1 и C1 вычислить, введя операторы:

```
30 LET A=XA1-XA:LET B=YA1-YA
35 LET XB1=XB+A:LET YB1=YB+B
40 LET XC1=XC+A:LET YC1=YC+B
```

Для построения искомых фигур достаточно применить операторы: LINE(XA1,YA1)—(XB1,YB1), C: LINE(XB1,YB1)—(XC1,YC1), C: LINE(XA1,YA1)—(XC1,YC1), C.

10. Более трудной является последняя задача — задача построения параллелограмма (рис. 73). В этом случае необходимо вычислить координаты точки A1(XA1;YA1):

```
30 LET XA1=XC+(XB-XA):LET YA1=YB
```

Графическая часть основной программы дополняется четырьмя операторами LINE для построения сторон параллелограмма.

СОРТИРОВКА



Выполнить сортировку элементов какого-нибудь множества — это значит упорядочить их по выбранному признаку. Числа можно сортировать по величине или абсолютной величине, детали — по их весу, список авторов составляется по так называемому лексикографическому признаку, в порядке алфавита.

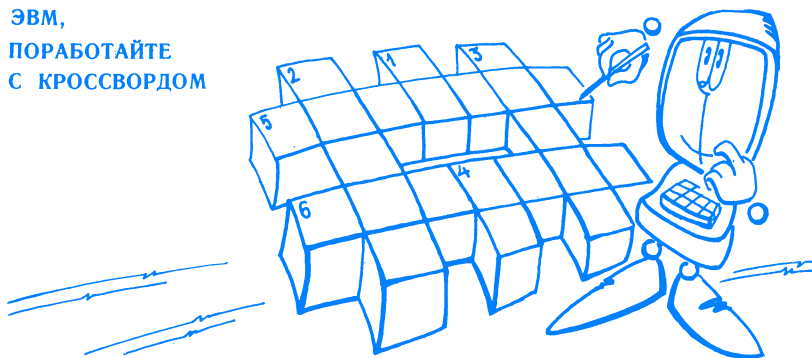
Задача сортировки — одна из практически важных задач, и поэтому математики и программисты потратили немало усилий для отыскания алгоритмов и эффективных программ ее решения. Приводим пример простейшей программы сортировки чисел по их величине.

```

10 REM СОРТИРОВКА ЭЛЕМЕНТОВ МАССИВА
20 PRINT "ИЗ N СОРТИРУЕМЫХ ЧИСЕЛ СФОРМИРУЙТЕ МАССИВ A(N)"
30 INPUT N
35 DIM A(N)
40 FOR I=1 TO N
50 INPUT A(I)
60 NEXT I
70 PRINT "МАССИВ СФОРМИРОВАН"
80 PRINT "УКАЖИТЕ ПРИНЦИП СОРТИРОВКИ"
90 PRINT "ПО ВОЗРАСТАНИЮ ЭЛЕМЕНТОВ ? ДА ИЛИ НЕТ ?"
100 INPUT K$
110 IF K$="ДА" THEN 190
120 FOR I=1 TO N-1
130 FOR J=I+1 TO N
140 IF A(I) > A(J) THEN 160
150 GOTO 170
160 SWAP A(I),A(J)
170 NEXT J,I
180 PRINT "МАССИВ ОТСОРТИРОВАН":GOTO 250
190 FOR I=1 TO N-1
200 FOR J=I+1 TO N
210 IF A(I) <= A(J) THEN 230
220 GOTO 240
230 SWAP A(I),A(J)
240 NEXT J,I:GOTO 180
250 PRINT "ВЫВОД ОТСОРТИРОВАННОГО МАССИВА"
260 FOR K=1 TO N
270 PRINT A(K);:NEXT K
280 END

```

ЭВМ,
ПОРАБОТАЙТЕ
С КРОССВОРДОМ



Рассказ о необычных геометрических экранных инструментах, по существу, — сборник заданий повышенной программистской трудности. Эти задачи под силу тем школьникам, которые хорошо овладели всеми основными средствами языка программирования Бейсик, особенно графическими.

Предлагаем рассмотреть еще один цикл заданий повышенной сложности. На этот раз в центре внимания средства символьной обработки. Речь пойдет о применении ЭВМ для решения и составления кроссвордов.

Пусть даны слова для заполнения небольшого кроссворда: АЛГОЛ, АВОСТ, СЛОВО, КРОССВОРД. Необходимым условием составления кроссворда является требование, чтобы каждое слово имело не менее двух пересечений с другими словами. Пусть в данной конкретной задаче известно, что слова КРОССВОРД и СЛОВО располагаются «по горизонтали», а слова АВОСТ и АЛГОЛ — «по вертикали». Читателю предлагается попробовать свои силы в организации того кроссворда (или кроссвордов — может быть, из этих же слов можно составить не один, а несколько кроссвордов), для которого эти слова образуют решение.

Сформулируем задание несколько иначе. На рисунке 74 изображена клетчатая доска 8×9 клеток. В некоторые строки и столбцы этой таблицы следует вписать заданные слова, образуя кроссворд.

Главное в задании — ответ на вопрос: как следует организовать работу с данными словами на выделенном пространстве? Может быть, можно создать подходящую математическую модель кроссворда и вести поиск на модели? Словом, дело за изобретательностью читателя. Вызов ему брошен серьезный.

На рисунке 75 показан наш вариант кроссворда.

Если у читателя сложилась какая-то методика решения таких задач, то предлагаем составить кроссворд из слов: КРОНА, КВАНТ, КОРАБЛЬ, КАРАВАН, АНАПА, ТОЛПА.

На рисунке 76 изображен возможный вариант кроссворда. В этом случае каждое слово имеет по три пересечения с другими словами.

Может быть, более простой для математического моделирования является задача формирования не кроссворда, а чайнворда — замкнутой последовательности слов? При этом последняя буква текущего слова является первой буквой следующего слова.

Задача. Из данных десяти слов составить чайнворд. Это значит указать номер каждого слова, входящего в чайнворд.

Слова для чайнворда: NEXT, END, PRINT, RETURN, TAB, TIME, EXT, COLOR, DELETE, BASIC.

С решением знакомит рисунок 77.

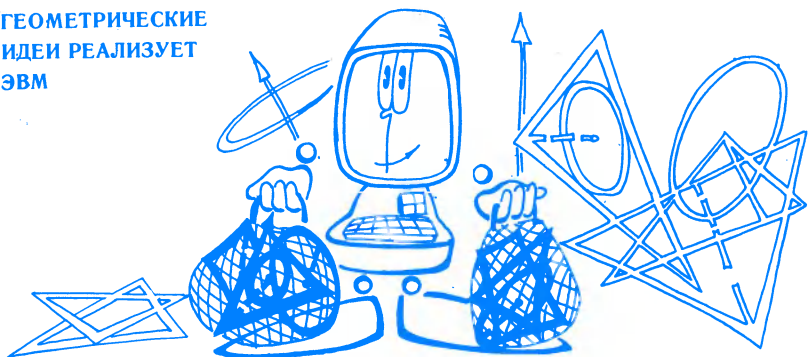
Наконец, еще одним вариантом задания может быть задача заполнения кроссворда словами. На рисунке 78 изображен незаполненный кроссворд. Известно, что слова, выписываемые в строки и столбцы, даны. Требуется только поручить машине расстановку этих слов и готовую схему.

Вновь — главное в идее, в подходе к общему решению. Какие ведущие соображения будут положены в основу машинного алгоритма? Как следует начать поиск, если нужно разместить слова:

По горизонтали: ПАК, КОМАНДА, ОТГУЛ.

По вертикали: КОНСУЛ, ПАМЯТЬ.

Вариант решения приводится на рисунке 79.



Ранее было рассказано, как можно изготовить «линейку», свободно перемещаемую на экране ЭВМ. Рассказ о линейке невольно подсказывает мысль об изготовлении других инструментов, с помощью которых можно на экране ЭВМ изображать множество различных линий. Интересным может стать задание на изготовление «экранного циркуля». Речь идет о циркуле, ножка которого может быть установлена в той точке экрана, на которую указывает курсор (управление курсором ведется с клавиатуры дисплея). Радиус окружности устанавливается по желанию пользователя, и его величина изменяется с помощью выделенных клавиш.

На рисунке 80 показаны две окружности, одна из которых изображена ЭВМ, выполнившей оператор: 15 CIRCLE(50, 60), 30, 7; другая вычерчена «экраным циркулем». В этом случае пользователь освобождает себя от всяких дел с координатами.

Следующим, более интересным, шагом в конструировании экранных инструментов может быть разработка для школы инструментов необычных. Так читатель может попробовать свои силы в изготовлении *эллипсопостроителя*.

Напомним, что эллипсом называется множество точек плоскости, сумма расстояний которых до двух фокусов есть величина постоянная. На рисунке 81 показано, как можно различными способами получить эллипс.

Уравнение эллипса в прямоугольной системе координат:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1.$$

Экранный эллипсопостроитель — это программа для ЭВМ, которая поручает машине построение эллипса на экране по заданным его полуосям (величины «*a*» и «*b*»), располагая его центр в точке, на которую указывает курсор. При этом оси эллипса могут быть параллельны осям координат или образовывать с ними угол. Построение эллипса должно осуществляться в режиме диалога. Пользователь должен иметь возможность вести наблюдения за всеми перемещениями эллипса и его поворотами.

На рисунке 82 показаны два эллипса. Один расположен так, что его горизонтальная ось совпадает с осью абсцисс, а другой эллипс повернут.

Ясно, что если в руках школьника помимо экранных циркуля и линейки появляется еще и экранный эллипсостроитель, то его возможности в решении геометрических задач на построение возрастают. Расширяется круг решаемых задач.

Остановимся для примера на одной: пусть нужно построить треугольник MNK , сумма боковых сторон которого равна S ($k + m = S$), длина медианы, проведенной к основанию, — l_n , а длина основания — n .

На рисунке 83 требуемый треугольник изображен, он вписан в эллипс. Это сделано потому, что фраза «сумма боковых сторон равна S » подсказывает мысль использовать для решения задачи эллипсостроитель.

Решение можно вести по плану.

1. Вычерчивается эллипс, величина осей которого: $a = \frac{S}{2}$;
 $b = \sqrt{\left(\frac{S}{2}\right)^2 + \left(\frac{n}{2}\right)^2}$, уравнение эллипса имеет вид:

$$\frac{x^2}{\left(\frac{S}{2}\right)^2} + \frac{y^2}{\left(\frac{n}{2}\right)^2} = 1.$$

2. Строится окружность с центром в центре эллипса, радиусом $R = l_n$. Для построения использован ранее изготовленный экранный циркуль.

3. С помощью экранной линейки строится искомый треугольник. На рисунке 84 показан весь ход построения.

Нетрудно сделать следующий шаг — попробовать сконструировать экранный *параболопостроитель*.

Заказ на программирование в этом случае может быть таким.

1. Параболопостроитель должен изображать на экране параболы вида $y = ax^2$.

2. Вершина параболы должна размещаться в точке, на которую указывает курсор.

3. Оптическая ось параболы, которая для таких парабол обычно расположена вертикально, по желанию пользователя должна менять свое положение. Управление поворотом оси должно осуществляться с клавиатуры.

4. Параметр « a » должен просто и быстро изменяться с клавиатуры.

На рисунке 85 показаны две параболы — одна расположена традиционно, ее ветви идут вверх, другая имеет наклоненную ось.

Располагая параболопостроителем, можно продемонстрировать простое и красивое решение одной из знаменитых задач геометрии. Речь идет о *задаче удвоения куба*.

Задача. Дан куб со стороной 1. Построить куб, объем которого в два раза больше объема данного куба.

Решить задачу — это значит найти сторону искомого куба. Пусть сторона искомого куба x , тогда $x^3=2$ и $x=\sqrt[3]{2}$. Известно, что, располагая только циркулем и линейкой, отрезок, равный $\sqrt[3]{2}$, построить невозможно. Обратимся к использованию экранного параболопостроителя.

Рассмотрим две параболы: $y=x^2$ и $y^2=2x$. Найдем абсциссы точек пересечения парабол.

$$x^4=2x \text{ или } x(x^3-2)=0 \text{ и } x_1=0, x_2=\sqrt[3]{2}.$$

Итак, искомая сторона куба есть абсцисса точки пересечения двух парабол $y=x^2$ и $y^2=2x$. Обе эти параболы изображены на экране (см. рис. 86).

Подчеркнем, что парабола $y^2=2x$ — это парабола, повернутая так, что ее оптическая ось оказалась расположенной горизонтально.

На рисунке 87 показано окончательное построение искомого отрезка. Все свелось к построению на экране параболы $y=x^2$ и параболы $y=\frac{1}{2}x^2$ с последующим ее поворотом на 90° . Не правда ли интересное решение?

В заключение остановимся еще на одном экранном геометрическом инструменте. Создание его потребовала одна древняя задача на построение. Речь идет о *трисекции угла*. Известно, что произвольный угол нельзя разделить на три равные части, пользуясь только циркулем и линейкой.

Задача эта становится легко и красиво разрешимой, если воспользоваться вспомогательным инструментом — *экраным построителем спиралей*.

Пусть в нашем распоряжении имеется экранный построитель спиралей Архимеда, то есть линий, имеющих вид, показанный на рисунке 88.

Спираль Архимеда можно построить так: начальная точка луча OA , расположенного горизонтально, закрепляется, и вокруг нее против часовой стрелки (или по часовой стрелке) начинает вращаться луч OA . В момент начала вращения луча по нему начинает перемещаться «жук», держащий в руках карандаш, с помощью которого ведется вычерчивание спирали. Скорость вращения луча и скорость движения «жука» постоянны. Уравнение спирали в полярной системе координат имеет вид $\rho=a\varphi$, где ρ — пройденный «жуком» путь, φ — величина угла поворота, a — коэффициент пропорциональности. Ясно, что увеличение одного из параметров (ρ или φ) вызывает увеличение другого.

На рисунке 89 показано, как следует разместить угол, трисекция которого осуществляется с помощью спирали Архимеда.

Пусть берется один из витков спирали Архимеда, и с его помощью выполняется трисекция. Процесс трисекции ясен из рисунка 90.

Основная идея решения: трисекцию угла свести к трисекции отрезка.

Нетрудно сообразить, что аналогичным образом можно данный угол разделить на три не обязательно равные части, а на части, пропорциональные числам $m:n:p$. Рисунок 90 разъясняет суть дела.

Пропорциональное деление отрезка на части вызывает пропорциональное деление дуги (витка) спирали на такие же пропорциональные части.

Рассматривая геометрические задачи перед их решением с помощью ЭВМ, следует ясно представлять, почему привлекается ЭВМ. Причин может быть несколько. Учитель может продемонстрировать новую фигуру школьникам, например, показать на экране все возможные виды параллелограммов или трапеций. На экране можно показать, как для какой-то фигуры строится симметричная ей относительно оси или точки, как осуществляется преобразование гомотетии или инверсия.

Все знают пословицу: «Лучше один раз увидеть, чем сто раз услышать», и человек сегодня с помощью ЭВМ обеспечивает демонстрацию удивительных геометрических и графических преобразований. Машина сегодня может изобразить такие фигуры и тела, которые человек при всем своем желании осуществить вручную не сможет. При этом ЭВМ самым причудливым образом будет менять цвет изображений, перемещать изображения, представлять их в различном масштабе и все это очень быстро и точно.

Следует заметить, что ЭВМ может не только формировать изображения, но и исследовать изображения и даже тела.

Речь идет о распознавании графических образов. Существуют ЭВМ, в числе внешних устройств которых имеется «электронный глаз». Такие машины могут отличить трапецию от параллелограмма, как бы они не были спроектированы на плоскости.

Важное применение уже сегодня находят машины, которые по команде: «возьми цилиндр» находят среди нескольких тел, размещенных на столе, цилиндр и приказывают «механической руке» взять именно цилиндр.

Можно предположить, что будут созданы ЭВМ, которые смогут рисовать «с натуры». Конечно это они будут делать по необычайно сложной программе, разработанной изобретательным программистом.

Фантазируя, можно представить себе конкурс машин-художников, подобно тому, как сегодня проводятся конкурсы ЭВМ, умеющих играть в шахматы.

ЗАКЛЮЧЕНИЕ



Приглашаем читателя, прочитавшего всю книгу или какую-то ее часть, оглянуться назад, просмотреть и оценить всю свою работу с книгой.

Через задачи — к программированию — какую цель ставил автор перед собой и перед читателем? Достигнута ли она? Цель ясна — познакомить читателя-школьника с тем, как готовят задачи к решению их на ЭВМ, и как машина задачи решает. В книге суть дела раскрывалась на примерах. Читатель узнал о многих сторонах взаимодействия человека и машины — преобразователя информации. Читатель увидел, как уже известные методы математики приобрели, в связи с использованием их на машинах, новое дыхание. Читатель мог разобраться в том, как вычислительная машина берется за решение геометрических задач, какую роль при этом играет метод координат; какие методы используются сегодня, если нужно решить задачу, которую раньше, всего несколько лет назад, мы назвали бы задачей из грамматики, а не из математики. Достаточно подробно было рассмотрено решение логических задач на ЭВМ. Связь между арифметикой и математической логикой показана в действии.

Все задачи строились как задачи-рассказы, и важной частью каждого был поиск алгоритма. Успех в применении ЭВМ определяется умением разработать эффективный алгоритм. Алгоритмическая культура — это понятие не абстрактное, а конкретное. Примеры неожиданных алгоритмических решений впечатляют.

Конечно, переход от алгоритма к тексту программы для ЭВМ требует высокой программистской культуры, основан на компьютерной грамотности.

Все сказанное в книге следует рассматривать лишь как дополнение к систематическому изучению основ информатики и вычислительной техники. Автор надеется, что книга поддержала интерес читателя к предмету.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Аршинов М. Н., Садовский Л. Е. **Коды и математика.**— М.: Наука, 1983.— 140 с.
2. Воробьев Н. Н. **Числа Фибоначчи.**— М.: Наука, 1972.— 180 с.
3. Гарднер Мартин. **Математические головоломки и развлечения.**— М.: Мир, 1971.— 510 с.
4. Гарднер Мартин. **Математические досуги.**— М.: Мир, 1972.— 496 с.
5. Гарднер Мартин. **Математические новеллы.**— М.: Мир, 1974.— 454 с.
6. Живые числа /Боро В., Цагир Д., Рольфо Ю. и др.— М.: Мир, 1985.— 126 с.
7. Журавлев А. П. **Языковые игры на компьютере:** Кн. для учащихся ст. классов сред. шк.— М.: Просвещение, 1988.—144 с.
8. Касаткин В. Н. **Логическое программирование в занимательных задачах.**— К.: Техника, 1980.— 80 с.
9. Касаткин В. Н., Владыкина Л. И. **Алгоритмы и игры.**— К.: Рад. шк., 1984.— 90 с.
10. Касаткин В. Н. **Новое о системах счисления.**— К.: Вища шк., 1982.— 90 с.
11. Кордемский Б. А., Ахадов А. А. **Удивительный мир чисел:** (Мат. головоломки и задачи для любознательных): Кн. для учащихся.— М.: Просвещение, 1986.— 144 с.
12. Котов Ю. В. **Как рисует машина.**— М.: Наука, 1988.— 224 с.
13. Кнут Д. **Искусство программирования.**— М.: Мир, 1976.— 831 с.
14. Миги Д., Джонсон Р. **Компьютер — творец:** Пер. с англ. /Предисл. А. А. Поспелова.— М.: Мир, 1987.— 255 с.
15. Нивергельт Ю., Фарар Дж., Рейнгольд Э. **Машинный подход к решению математических задач.**— М.: Мир, 1977.— 320 с.

СОДЕРЖАНИЕ

Введение	3
Глава I. Вычисляем вместе с ЭВМ	5
Как найти сумму?	7
Сколько цифр в числе 100!?	13
Основание системы счисления — миллион	17
Шахматное число на экране ЭВМ	20
Задача о дешифраторе	21
Вычисляем в СОК	26
Числа Фибоначчи	33
Числа Бернулли	34
Числа Каталана	36
Числа Мерсенна	38
Генератор простых чисел	40
Глава II. В геометрию с ЭВМ	45
Первые шаги ЭВМ в геометрии	47
Еще один шаг в геометрию	51
Встреча на орбите	60
Формула Кардано на ЭВМ	65
Глава III. ЭВМ учится логике	69
ЭВМ выполняет логические операции	71
По единому алгоритму	73
Арифметика служит логике	76
Глава IV. В диалоге с ЭВМ	79
Автоматический телефонный справочник	81
В соперничестве с человеком	84
Задача о фальшивой монете	89
Глава V. И в шутку, и всерьез	91
Упорядочение таблицы наблюдений	93
Получите сдачу	94
Магический ли квадрат?	95
Задание Кванта	97

Треугольник Паскаля и треугольник Лейбница	98
Числовая спираль	101
Пауза в работе ЭВМ	102
Сложение в Q -арифметике	104
Арифметика Льва Толстого	106
Для всех «средних» одна программа	108
Неизвестное под знаком факториала	110
Палиндром ли?	111
От центра к центру	113
Линейка на экране	114
Преобразование треугольника	116
Сортировка	118
ЭВМ, поработайте с кроссвордом	119
Геометрические идеи реализует ЭВМ	121
Заключение	125
Список рекомендуемой литературы	126

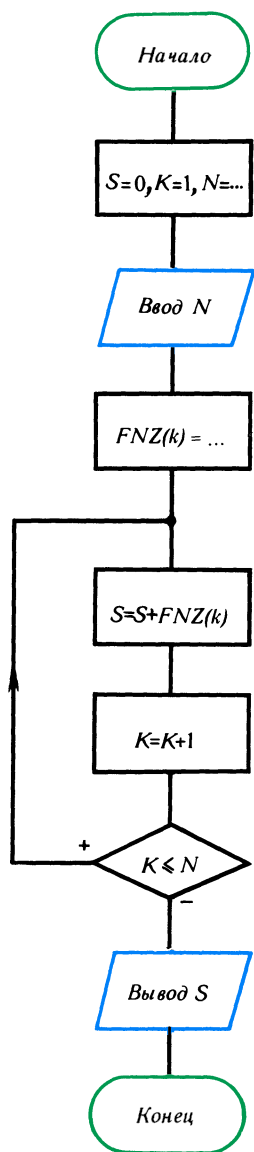


Рис. 1

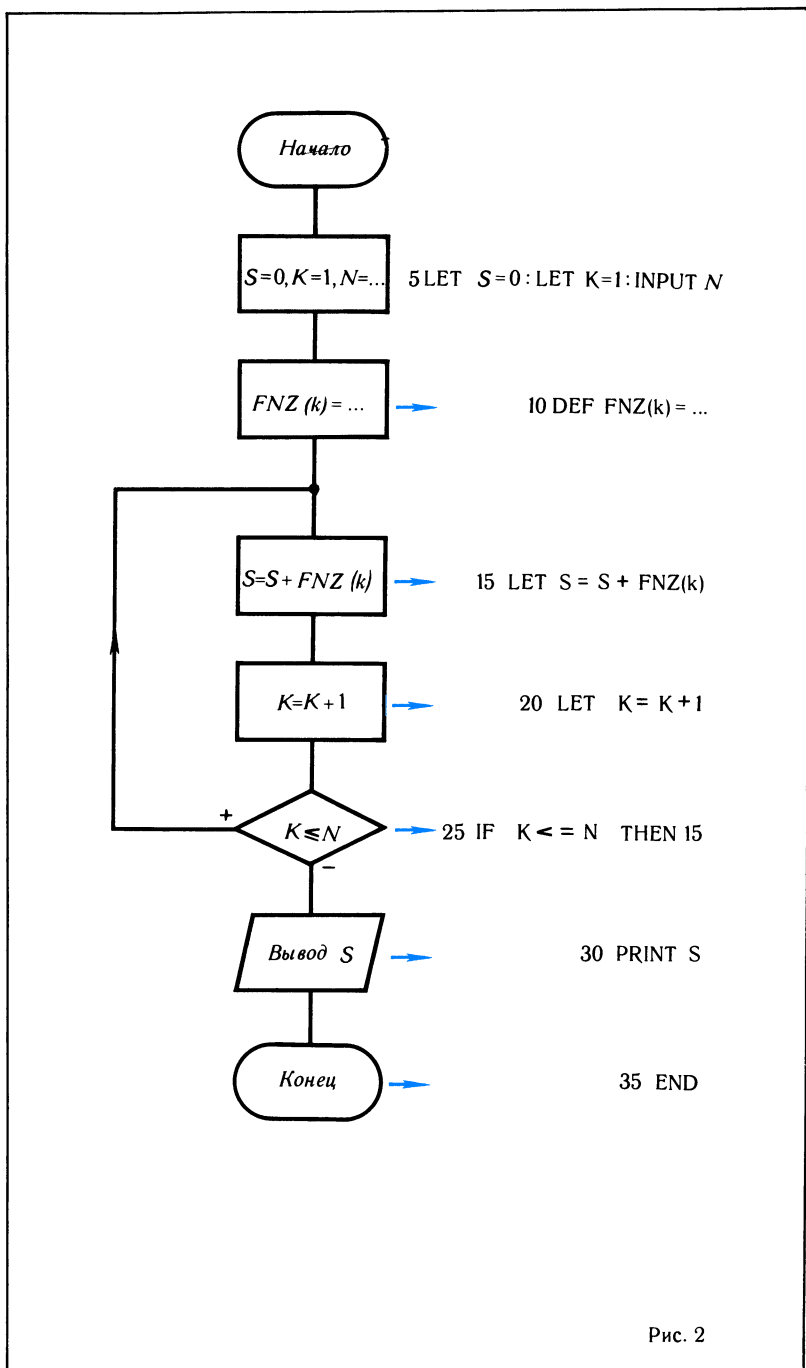


Рис. 2

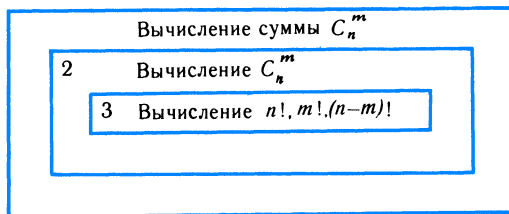


Рис. 3

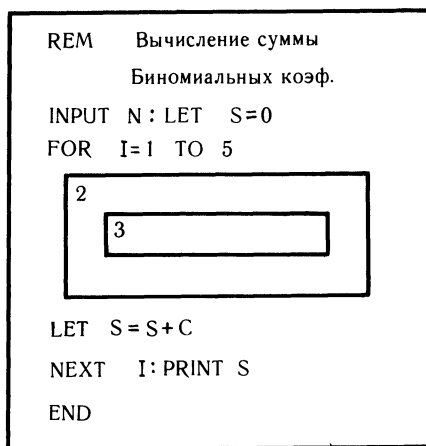


Рис. 4

a_n	...	a_4	a_3	a_2	a_1	
0		0	0	0	0	
0		0	0	0	1	Вычислен $1! = 1$
0		0	0	0	2	Вычислен $2! = 2$
0		0	0	0	6	Вычислен $3! = 6$
0		0	0	2	4	Вычислен $4! = 24$
0		0	1	2	0	Вычислен $5! = 120$

Рис. 5

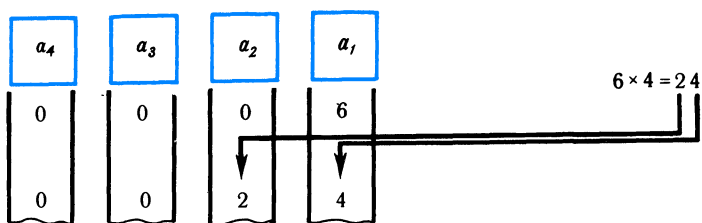


Рис. 6

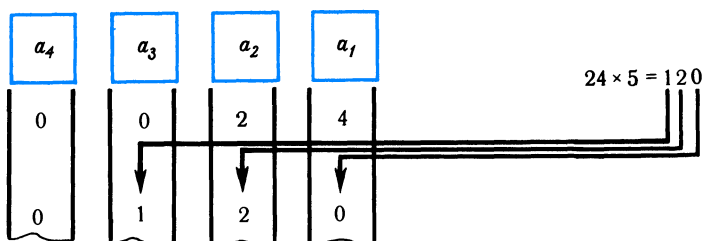


Рис. 7

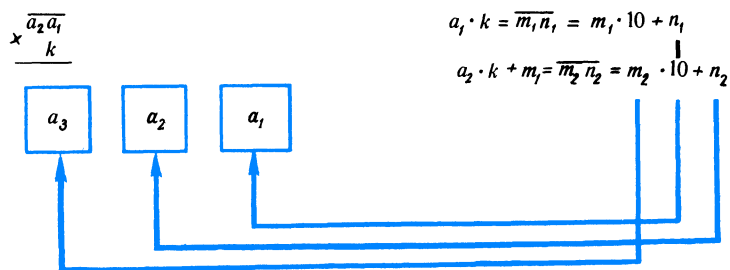


Рис. 8

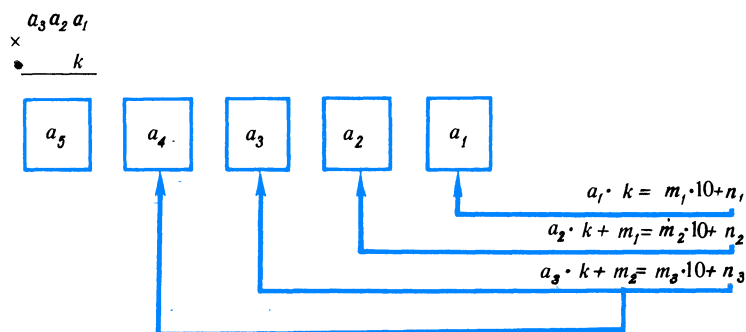


Рис. 9

Таблица 1	
210 REM ВЫЧИСЛЕНИЕ ФАКТОРИАЛА ЧИСЛА N	- название программы
20 DIM A(100)	- резервирование места в памяти ЭВМ для массива из 100 переменных
30 INPUT N	- ввод числа N , факториал которого вычисляет ЭВМ
40 FOR I=2 TO 100 50 LET A(I)=0:NEXT I	- переменным a_i присваивается начальное значение 0, $a_1=1$
60 LET A(1)=1:LET L=1	- количество пока использованных L разрядов искомого числа $N!$
70 FOR K=1 TO N 80 LET R2=0:LET R1=0:LET I=1 90 IF R2=0 THEN IF I>L THEN 160	- формирование множителей в произведение $K(K-1)!$
100 LET R=A(I)*K+R2	- вычисление $R = a_i \cdot K + R_{i-1}$
110 LET R2=INT(R/(10^6))	- выделение единиц следующего разряда
120 LET R1=R-R2*(10^6)	- выделение из числа R единиц i -го разряда
130 LET A(I)=R1	- запоминание текущей цифры i -го разряда в ячейке a_i
140 LET I=I+1 150 GOTO 90	- переход к работе с цифрами следующего разряда
160 LET L=I-1	- уточнение числа использованных разрядов
170 NEXT K 180 REM ВЫВОД ФАКТОРИАЛА ЧИСЛА N 190 FOR J=1 TO I-1 200 PRINT A(I-J); 210 NEXT J 220 END	<p>- переход к вычислению $K(K-1)!$ - вывод вычисленного числа $N!$</p> <p>Выводятся значения переменных $\overline{a_i a_{i-1} a_{i-2} \dots a_2 a_1}$</p>

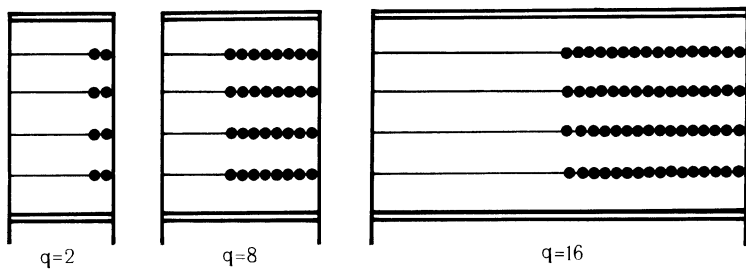


Рис. 10

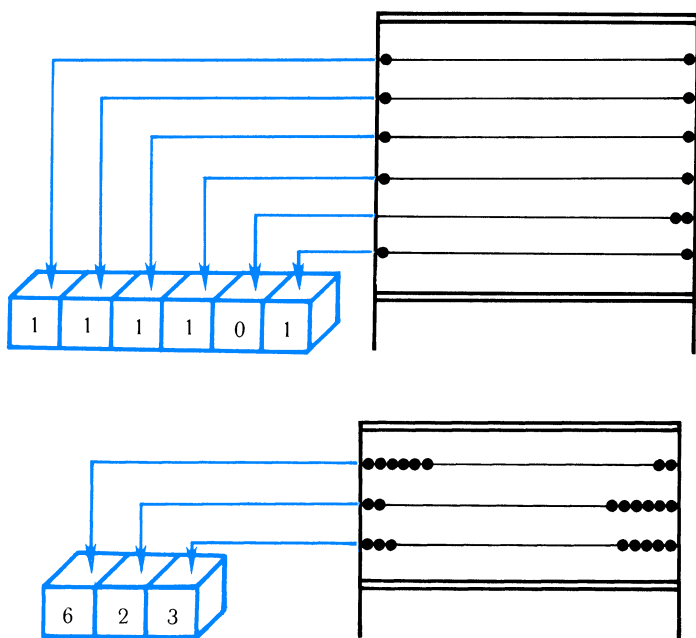


Рис. 11

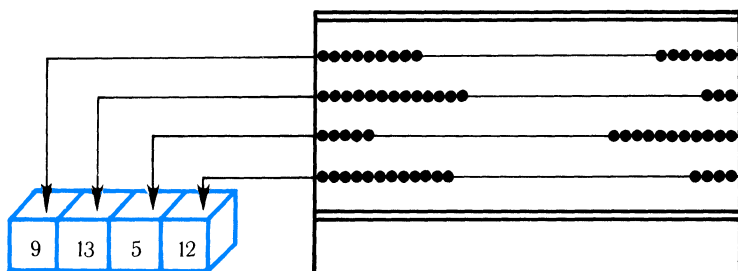


Рис. 12

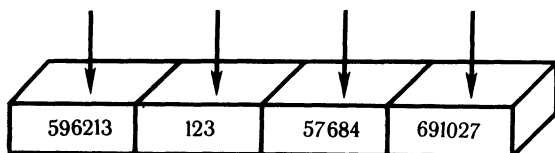


Рис. 13

N	N!
...	...
8	40320
9	362880
10	3628800
<div> <div></div> <div></div> <div></div> </div>	

	40320
	362880
3	628800

Рис. 14

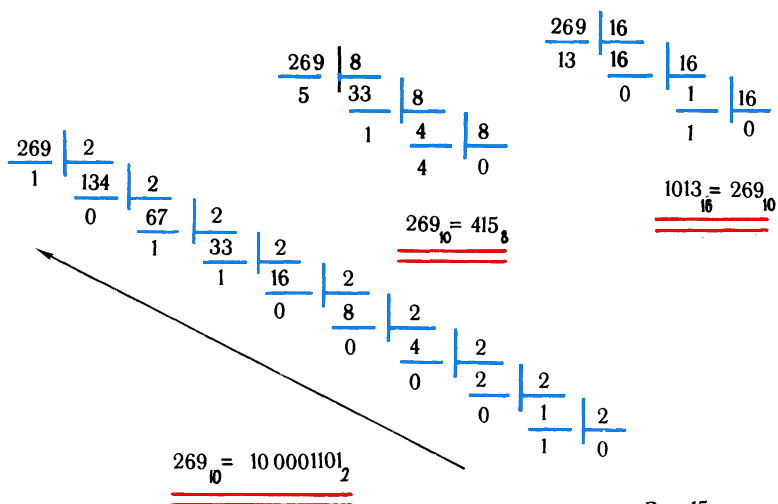


Рис. 15

```

5 REM ГЕНЕРАТОР ЧИСЕЛ БЕРНУЛЛИ
10 INPUT "ВВЕДИ НОМЕР ИСКОМОГО ЧИСЛА А";А
20 LET B(0)=1
30 IF A=1 THEN 170
40 REM ВЫЧИСЛЕНИЕ ПРЕДЫДУЩИХ ЧИСЕЛ БЕРНУЛЛИ
50 FOR I=1 TO A-1
60 LET N=I+1
70 FOR M=0 TO I-1
80 IF M=0 THEN LET CNM=1:GOTO 100
90 GOSUB 280
100 LET B(I)=B(I)+CNM*B(M)
110 NEXT M
120 LET M=I
130 GOSUB 280
140 LET B(I)=-B(I)/CNM
150 NEXT I
160 REM ВЫЧИСЛЕНИЕ ИСКОМОГО ЧИСЛА БЕРНУЛЛИ
170 FOR M=0 TO A-1
180 LET N=A+1
190 GOSUB 280
200 LET B(A)=B(A)+CNM*B(M)
210 NEXT M
220 LET M=A
230 GOSUB 280
240 LET B(A)=-B(A)/CNM
250 PRINT B(A)
260 END
270 REM ВЫЧИСЛЕНИЕ БИНОМИАЛЬНОГО К-ТА
280 LET K=N
290 GOSUB 390
300 LET D=F
310 LET K=M
320 GOSUB 390
330 LET E=F
340 LET K=N-M
350 GOSUB 390
360 LET CNM=D/(E*F)
370 RETURN
380 REM ВЫЧИСЛЕНИЕ ФАКТОРИАЛА
390 LET F=1
400 FOR Y=1 TO K
410 LET F=F*Y
420 NEXT Y
430 RETURN

```

The flowchart illustrates the program's logic. Red arrows show the flow for calculating Bernoulli numbers, starting from line 10, through the nested loops for I and M (lines 50-150), and then through the calculation of the final value B(A) (lines 170-250). Green arrows show the flow for the auxiliary subroutines: calculating binomial coefficients (lines 280-370) and factorials (lines 390-430), which are called from lines 90, 130, 190, 230, 290, 320, 350, and 360.

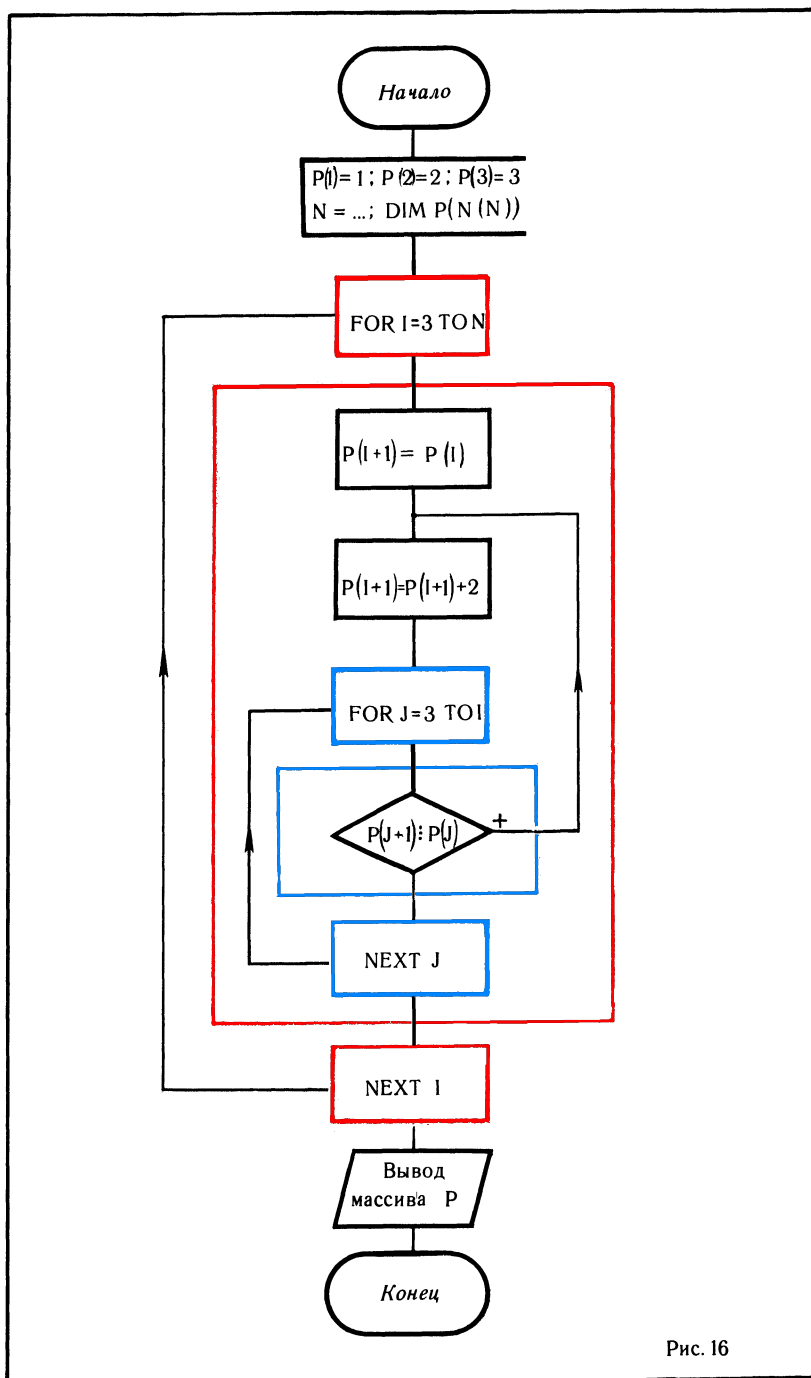


Рис. 16

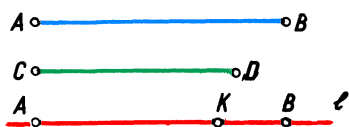


Рис. 17

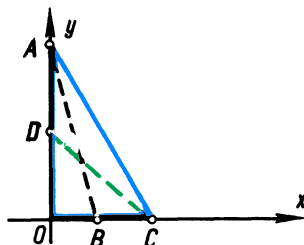


Рис. 20

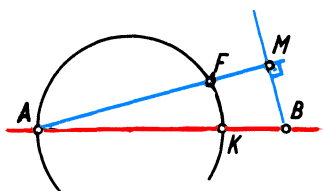


Рис. 18

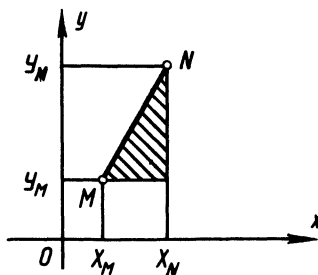


Рис. 21

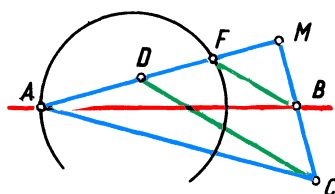


Рис. 19

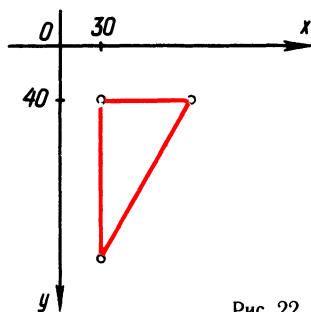


Рис. 22

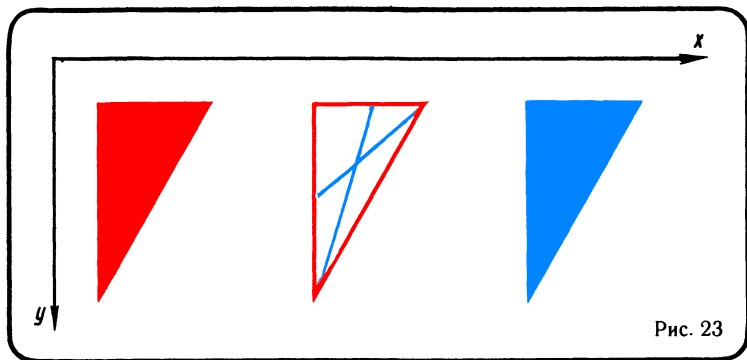


Рис. 23

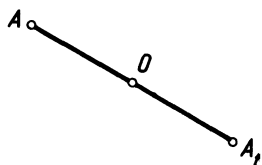


Рис. 24

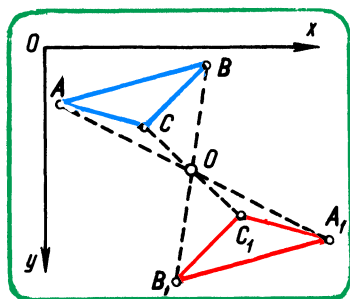


Рис. 27

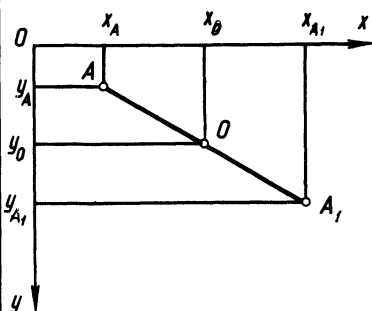


Рис. 25

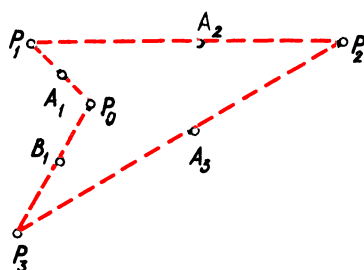


Рис. 28

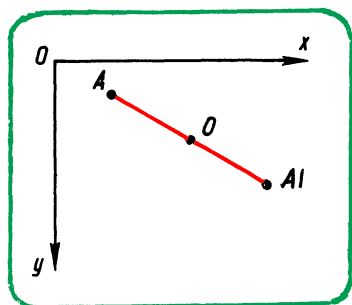


Рис. 26

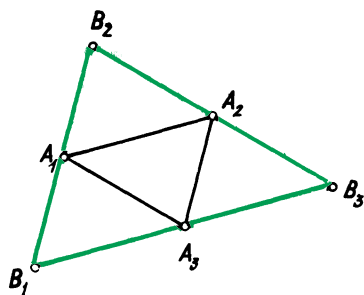


Рис. 29

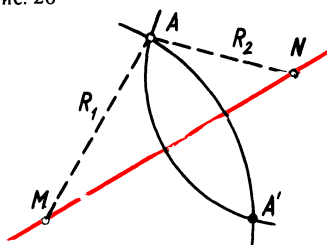


Рис. 30

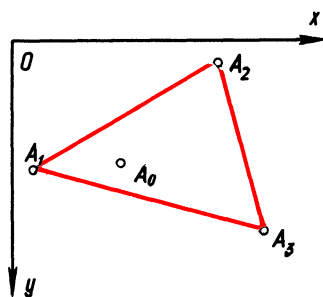


Рис. 31

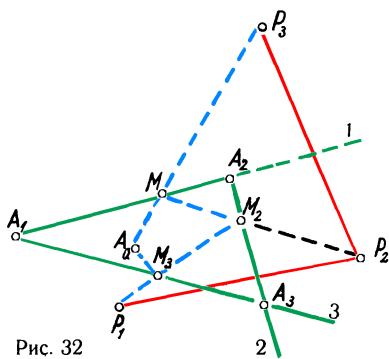


Рис. 32

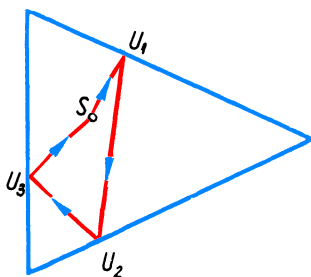


Рис. 33

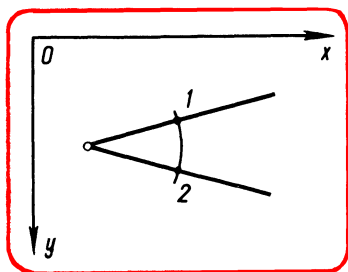


Рис. 34

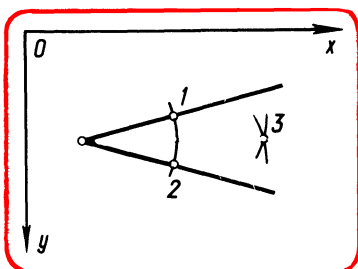


Рис. 35

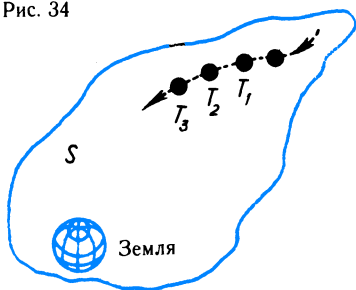


Рис. 36

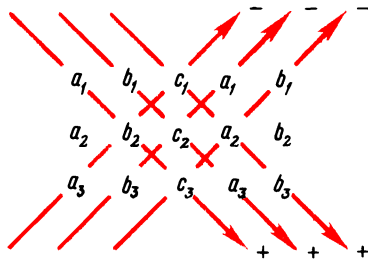


Рис. 37

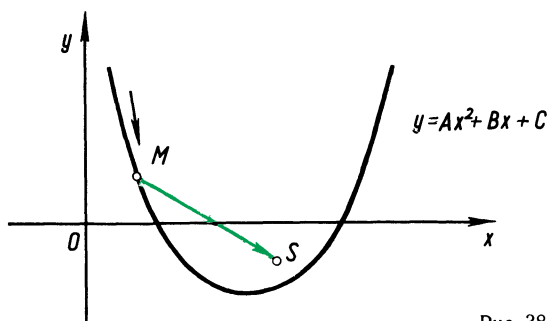


Рис. 38

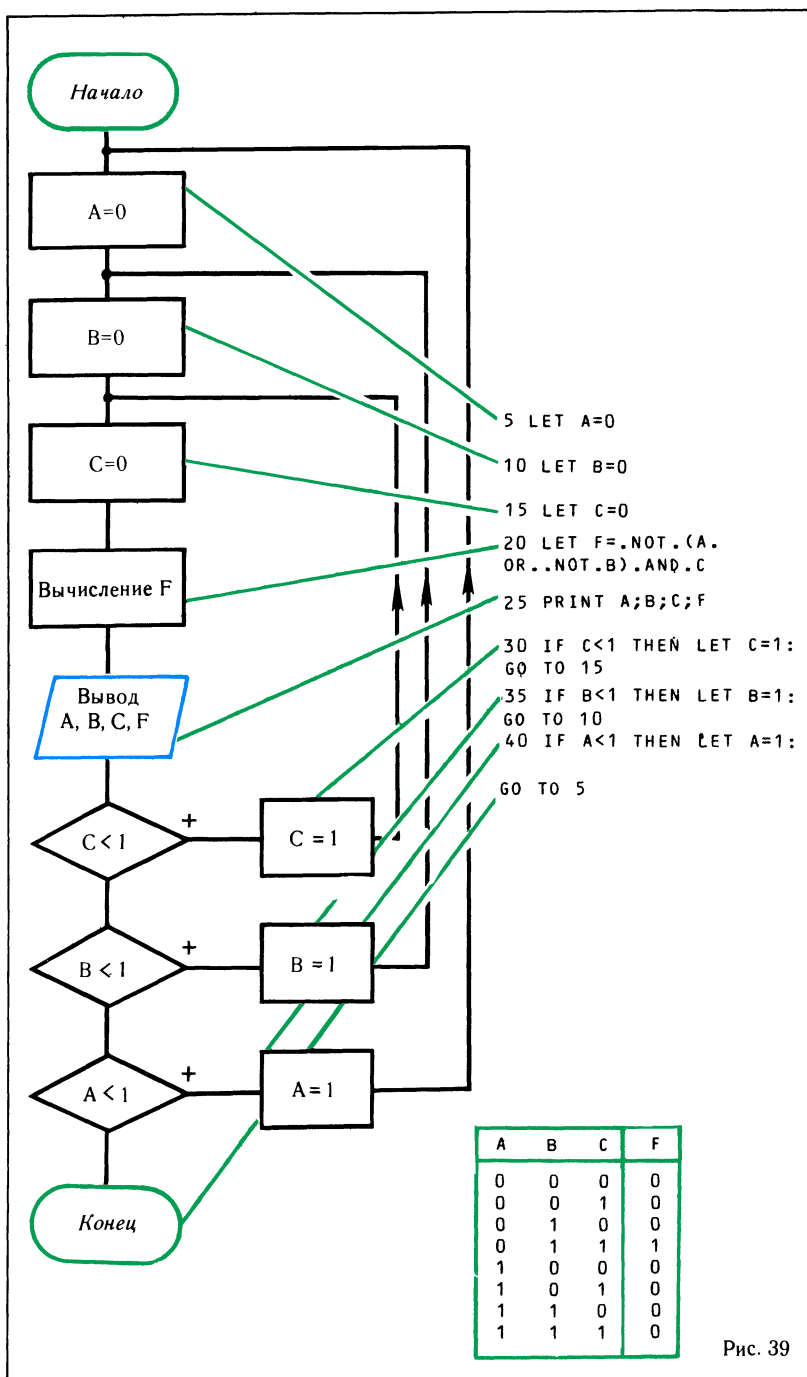


Рис. 39

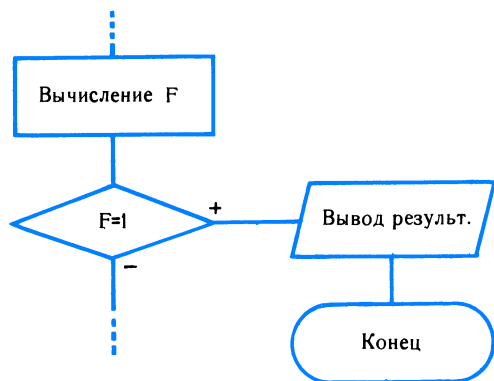


Рис. 40

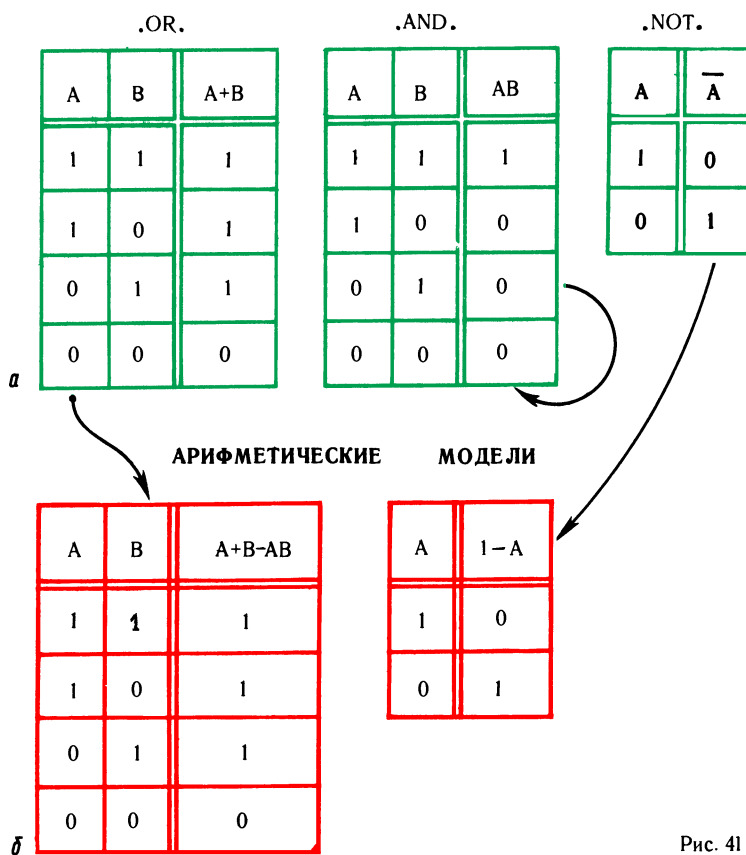


Рис. 41

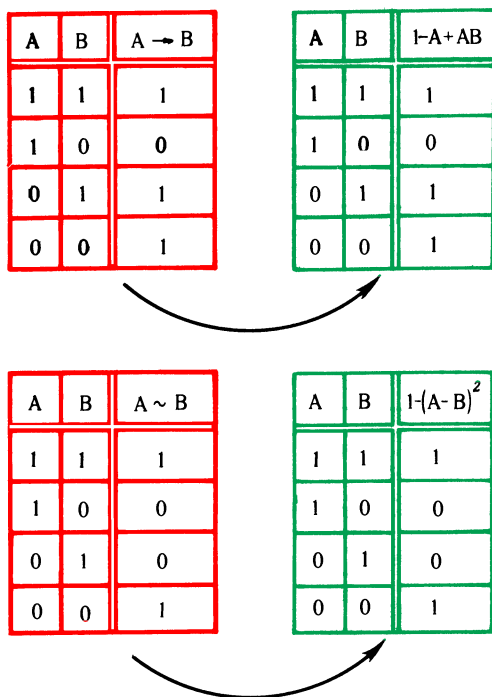


Рис. 42

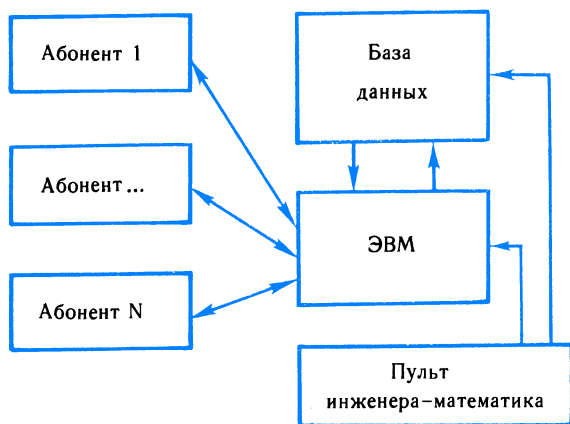


Рис. 43

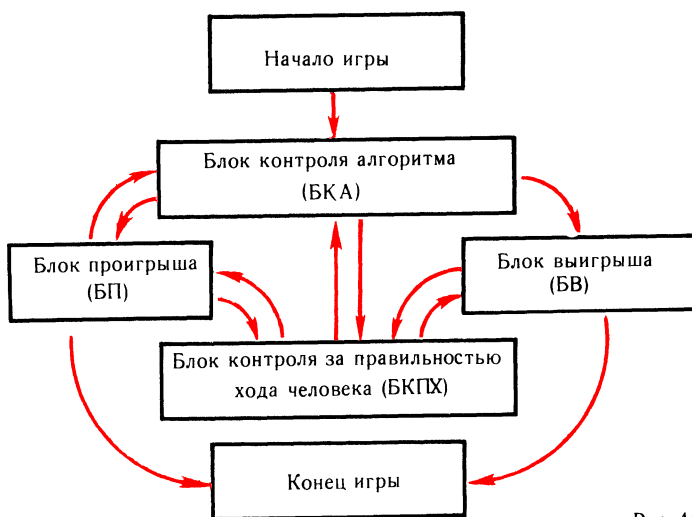


Рис. 44

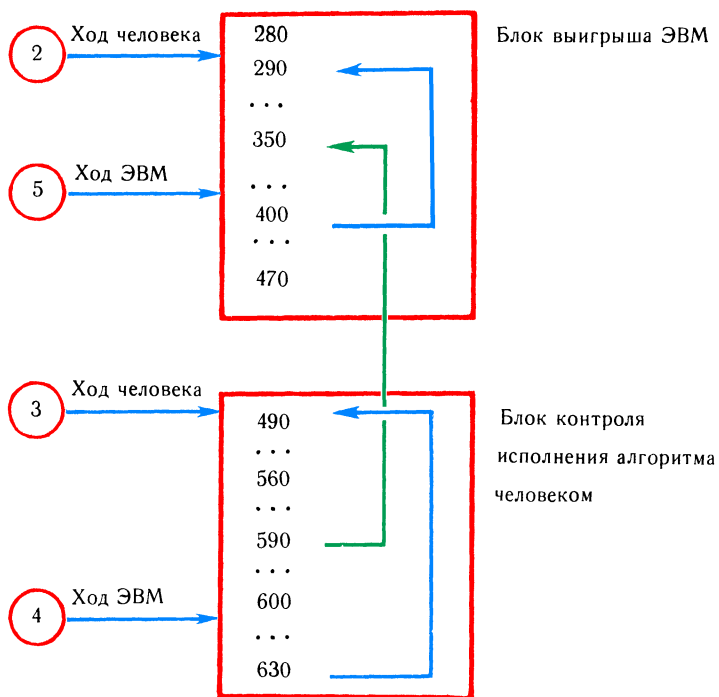
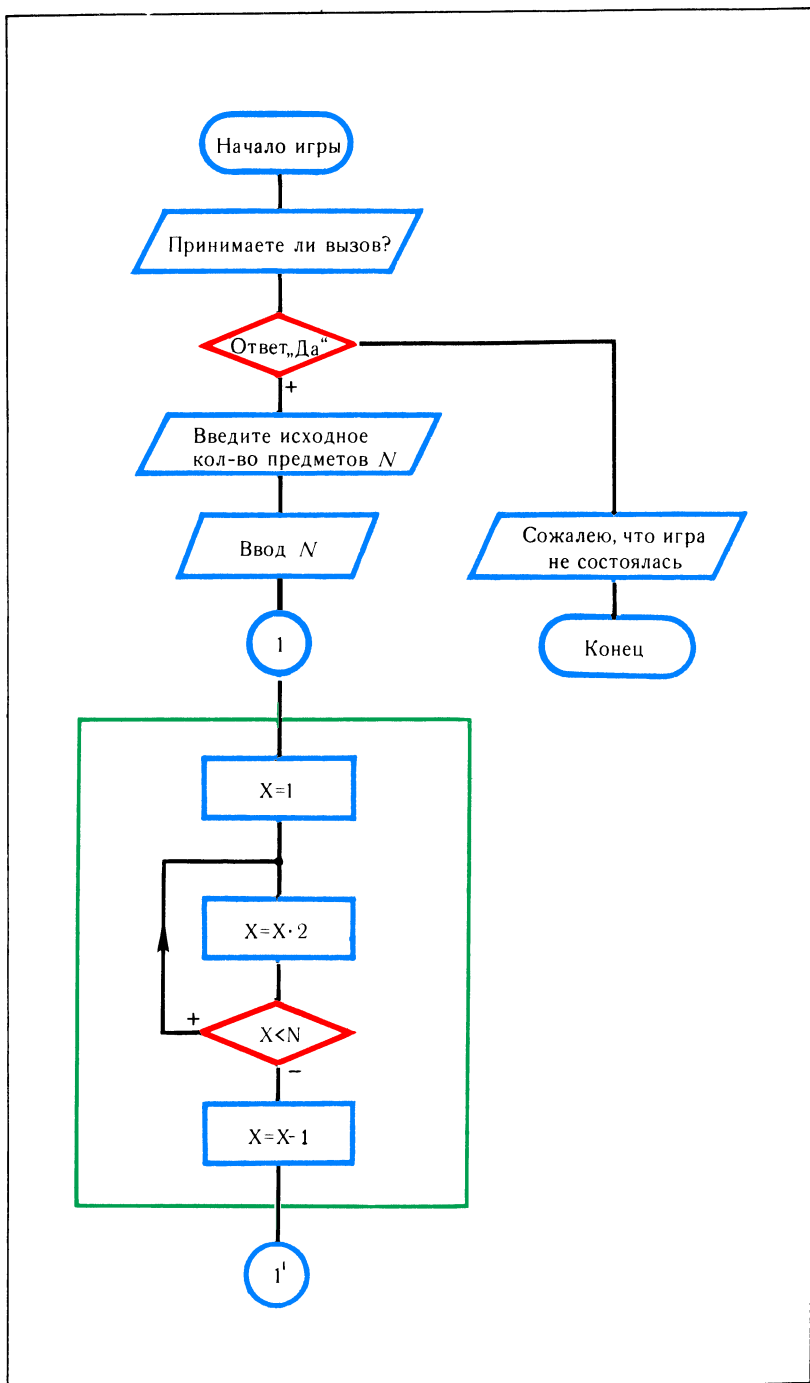


Рис. 45



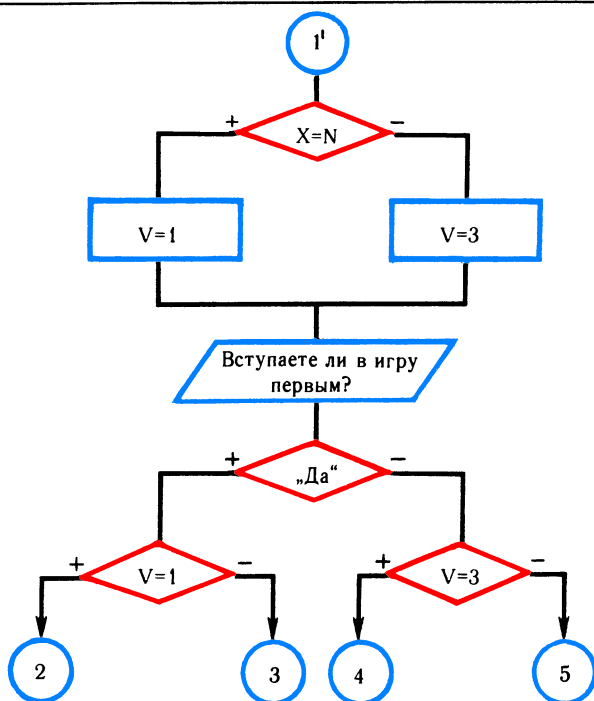


Рис. 46

```

280 REM "БЛОК ВЫИГРЫША ЭВМ"
290 INPUT "ВАШ ХОД. БЕРИТЕ R ПРЕДМЕТОВ R";R
300 IF R>N/2 THEN PRINT "НАРУШЕНЫ ПРАВИЛА":GOTO 290
310 LET N=N-R
320 PRINT "ОСТАЛОСЬ N=";N;"ПРЕДМЕТОВ"
330 IF N<=1 THEN 410
340 REM "РАСЧЕТ НАИЛУЧШЕГО ХОДА ЭВМ"
350 LET L=1
360 LET L=L*2
370 IF L<N THEN 360 ELSE LET M=N-(L/2-1)
380 PRINT "БЕРУ M ПРЕДМЕТОВ M=";M
390 LET N=N-M:PRINT "ОСТАЛОСЬ N=";N;"ПРЕДМЕТОВ"
400 IF N>1 THEN 290
410 PRINT "ВЫ ПРОИГРАЛИ. БЛАГОДАРИЮ ЗА ИГРУ"
420 PRINT "СДАЮСЬ. БЛАГОДАРИЮ ЗА ИГРУ"
430 PRINT "НЕ ХОТИТЕ ЛИ ПОВТОРИТЬ ИГРУ?"
440 PRINT "ДА ИЛИ НЕТ?"
450 INPUT W$
460 IF W$="ДА" THEN 110 ELSE END
470 IF W$="НЕТ" AND V=3 THEN 350
  
```

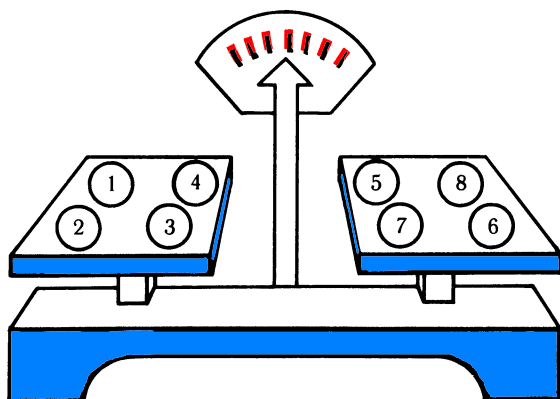


Рис. 47

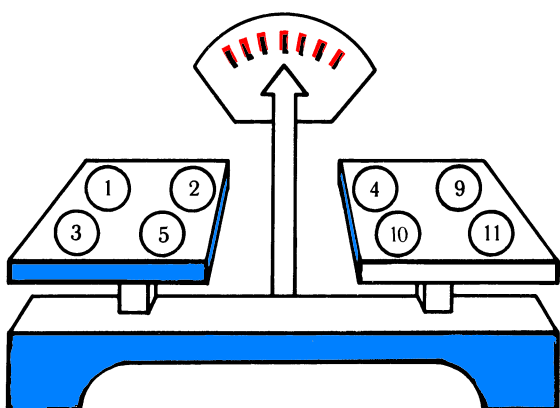


Рис. 48

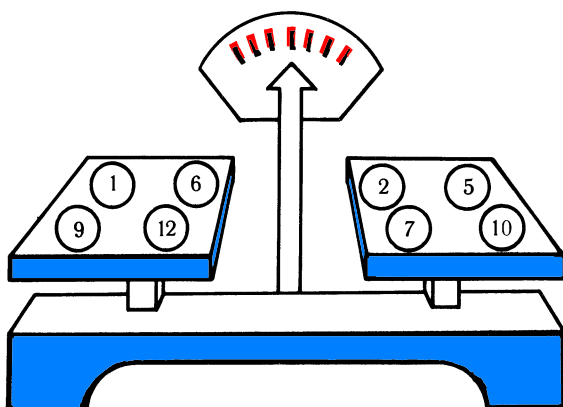
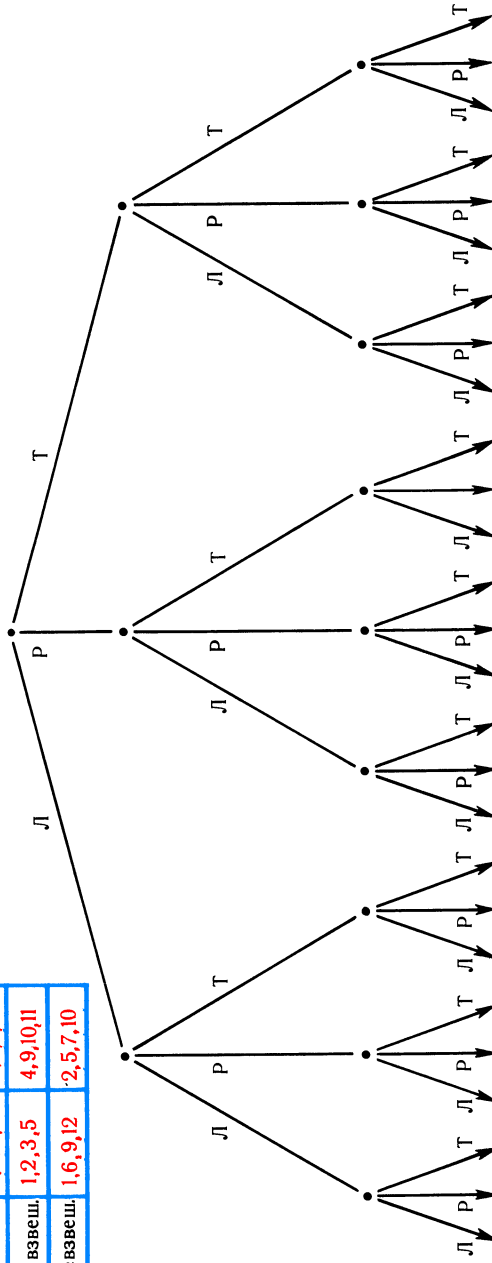


Рис. 49

Возможные результаты взвешиваний

	Левая	Правая
1-е взвеш.	1,2,3,4	5,6,7,8
2-е взвеш.	1,2,3,5	4,9,10,11
3-е взвеш.	1,6,9,12	2,5,7,10

[illegible][illegible]

- **Фальшивая монета легче**

- **Фальшивая монета тяжелее**

Рис. 50

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

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

Рис. 51

1	823	821	809	811	797	119	29	313	31	23	37
89	83	211	79	641	631	619	709	617	53	43	739
97	227	103	107	193	557	719	727	607	139	757	281
223	653	499	197	109	113	563	479	173	761	587	157
367	379	521	383	241	467	257	263	269	167	601	599
349	359	353	647	389	331	317	311	409	307	293	449
503	523	233	337	547	397	421	17	401	271	431	433
229	491	373	487	461	251	443	463	137	439	457	283
509	199	73	541	347	191	181	569	577	571	163	593
661	101	643	239	691	701	127	131	179	613	277	151
659	673	677	683	71	67	61	47	59	743	733	41
827	3	7	5	13	11	787	769	773	419	149	751

Рис. 52

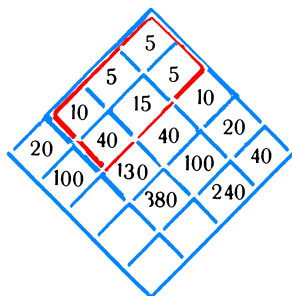


Рис. 53

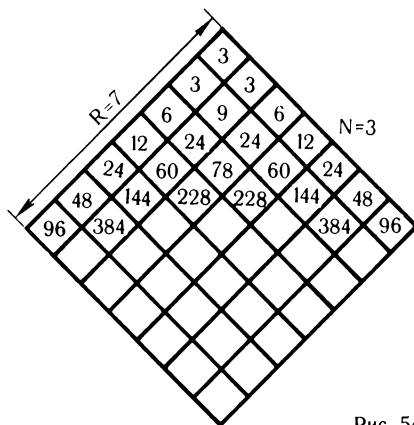


Рис. 54

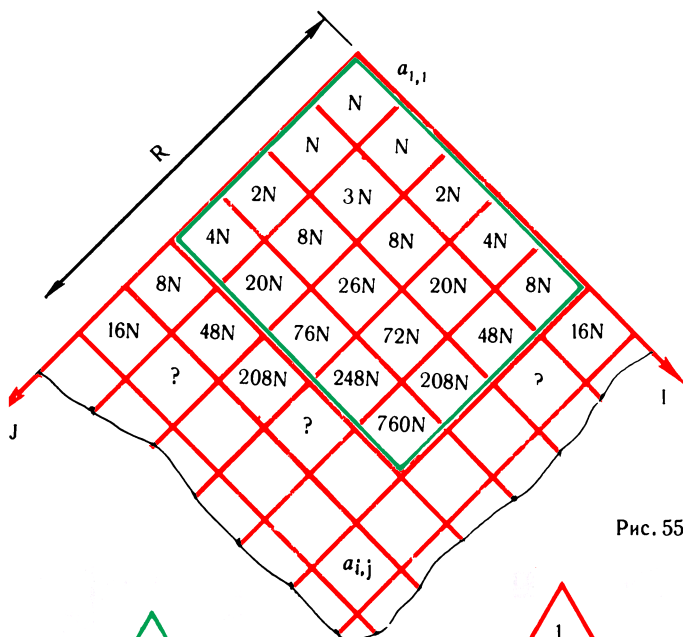


Рис. 55

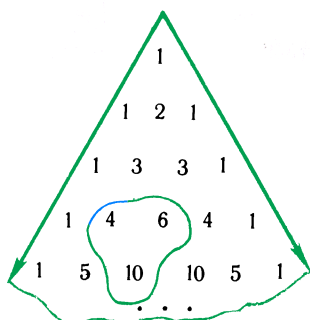


Рис. 56

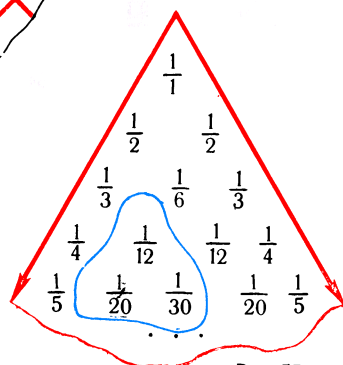


Рис. 57

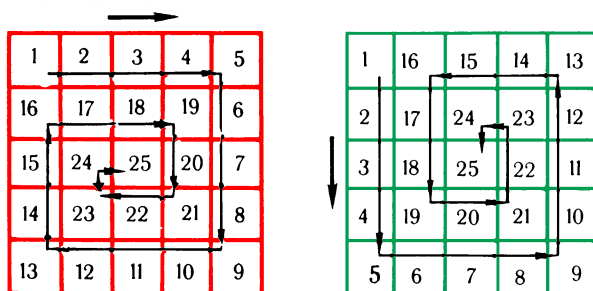


Рис. 58

+	0	1	2
0	0	1	2
1	1	2	10
2	2	10	11

q=3

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	12
4	4	10	11	12	13

q=5

Рис. 59

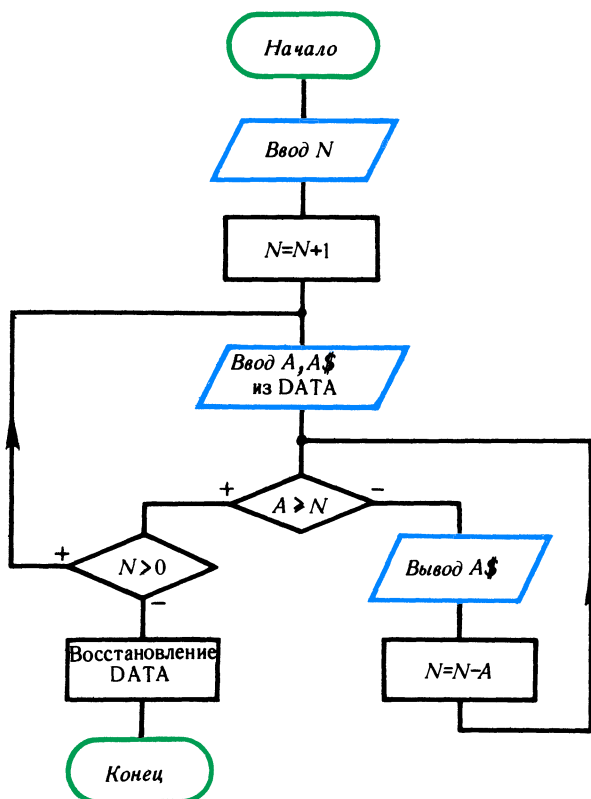


Рис. 60

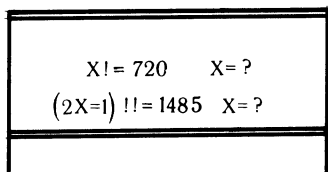


Рис. 61

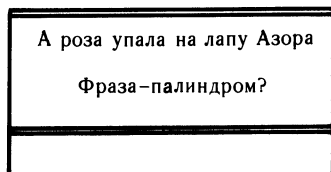


Рис. 62

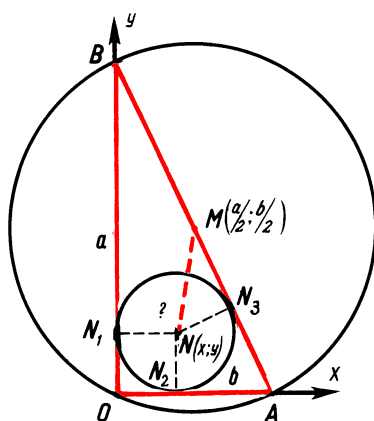


Рис. 63

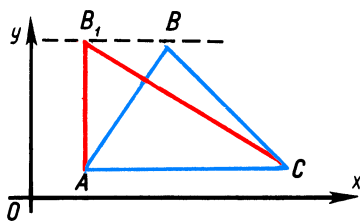


Рис. 64

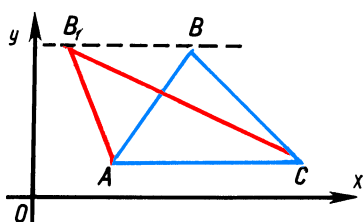


Рис. 65

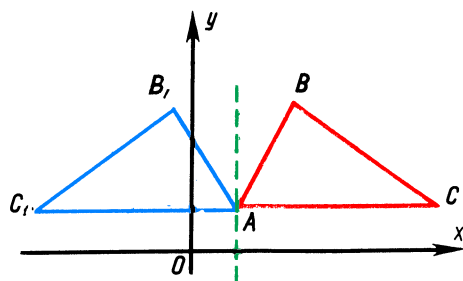


Рис. 66

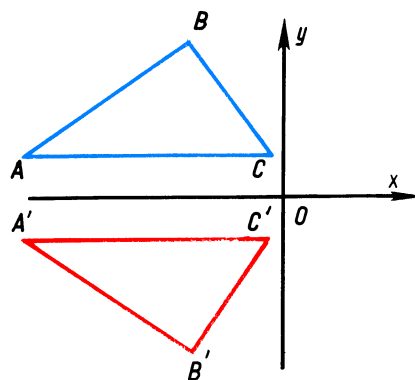


Рис. 67

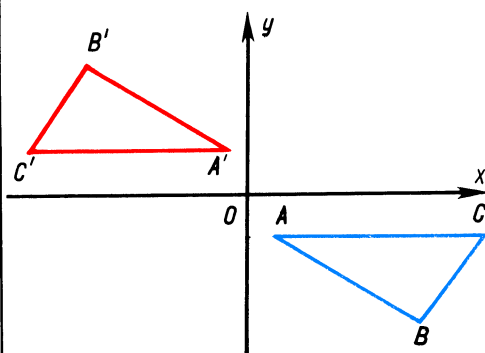


Рис. 68

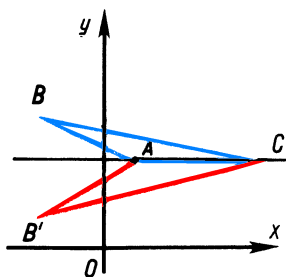


Рис. 69

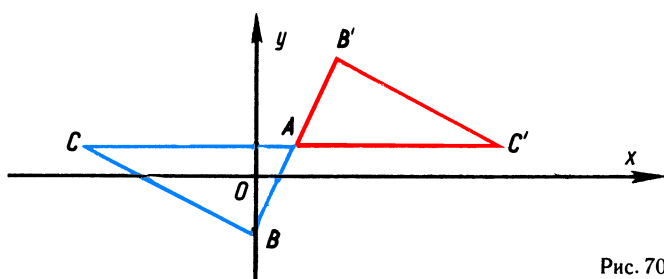


Рис. 70

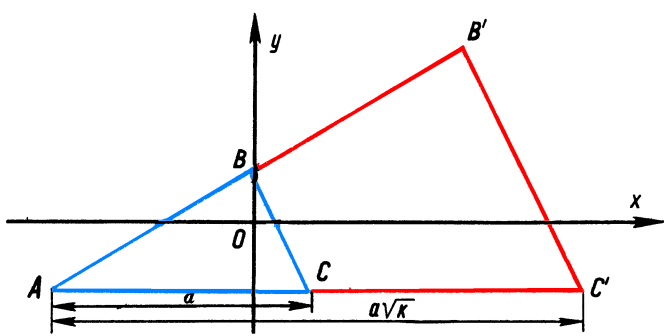


Рис. 71

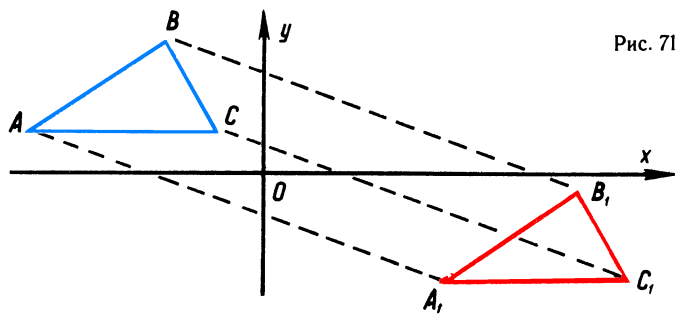


Рис. 72

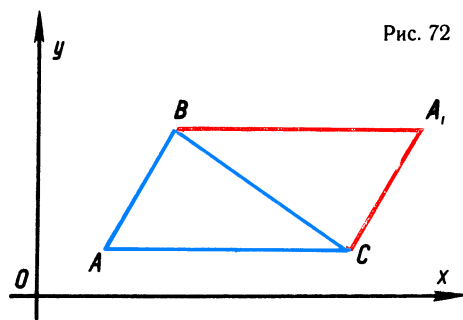


Рис. 73

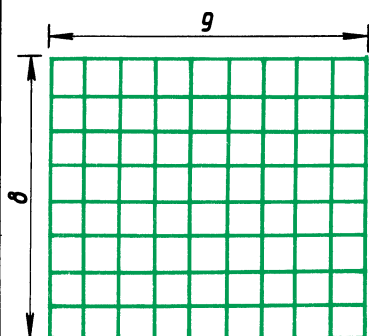


Рис. 74

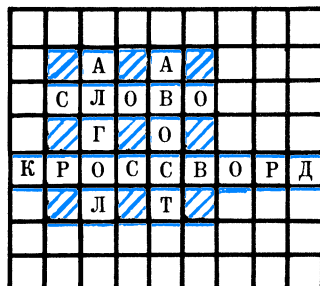


Рис. 75



Рис. 76



Рис. 77

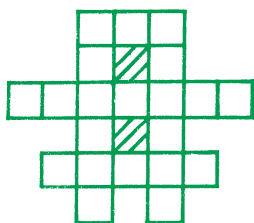


Рис. 78



Рис. 79

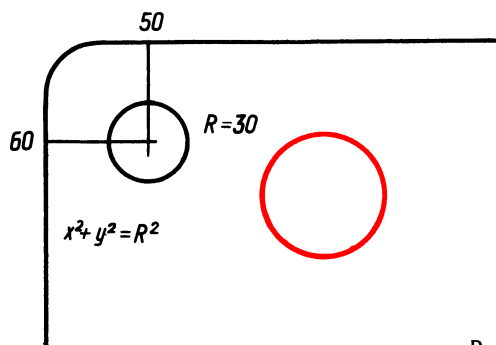


Рис. 80

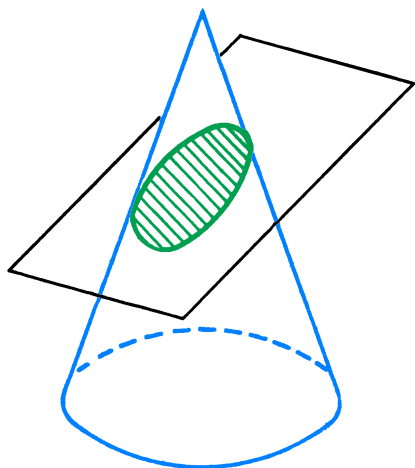
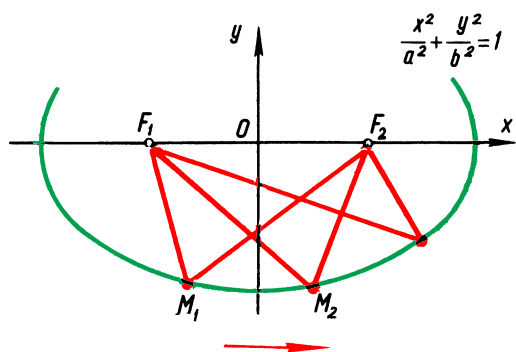


Рис. 81

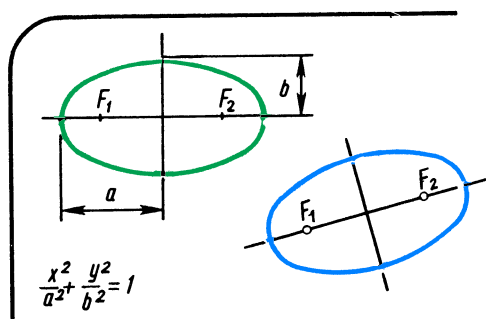


Рис. 82

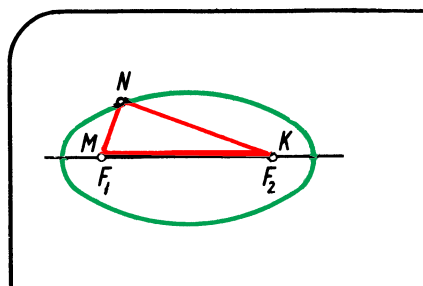


Рис. 83

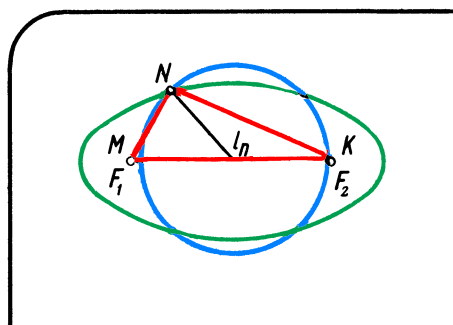


Рис. 84

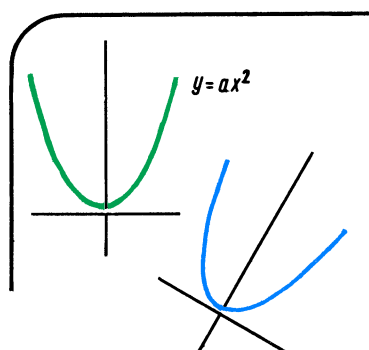


Рис. 85

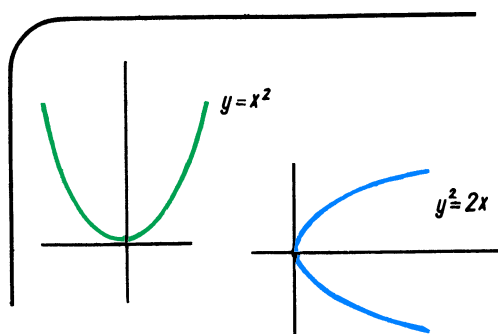


Рис. 86

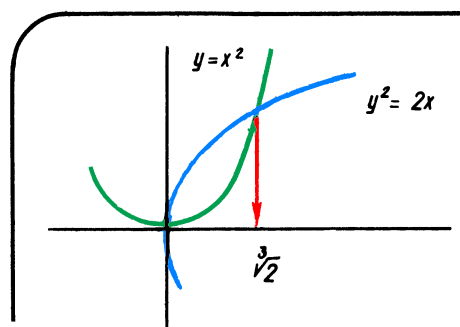


Рис. 87

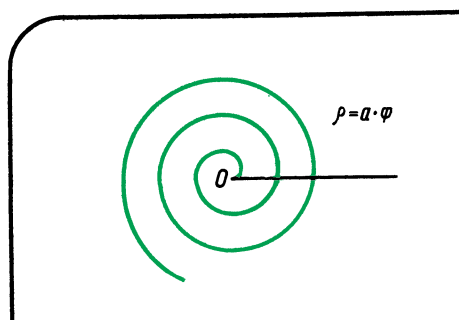


Рис. 88

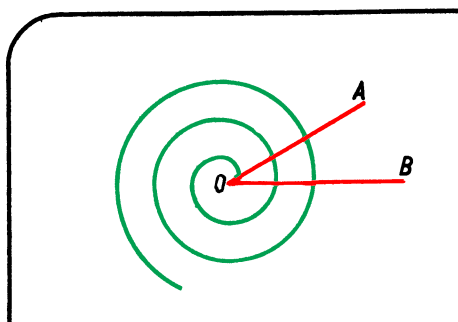


Рис. 89

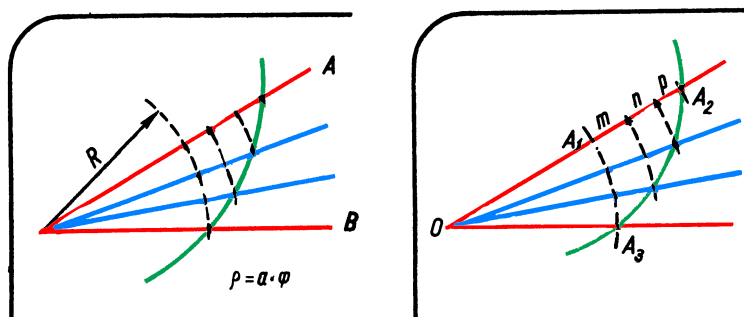


Рис. 90

УЧЕБНОЕ ИЗДАНИЕ

Серия «Когда сделаны уроки»

Касаткин Валентин Николаевич

Через задачи — к программированию

Для старшего школьного возраста

Художник *Кобрин Сергей Николаевич*

Заведующий редакцией математики *Н. Е. Зубченко*. Художественный редактор *В. А. Пузанкевич*. Технический редактор *А. Г. Фридман*. Корректор *Л. А. Волюнская*

ИБ № 5889

Сдано в набор 27.06.88. Подписано в печать 12.12.88. БФ 05719.
Формат 60×90/16 Бумага офсетн. № 1. Гарнитура литературная
Печать офсетная. Усл. печ. л. 8+2 вкл. Усл. кр.-отт. 20,30. Уч.-
изд. л. 7,44+1,97 вкл. Тираж 100 000 экз. Изд. № 39932
Заказ 8-161 Цена 55 к.

Издательство «Радянська школа»,
252053, Киев, Ю. Коцюбинского, 5.

Книжная фабрика «Коммунист», 310012, Харьков, Энгельса, 11

55 к.

