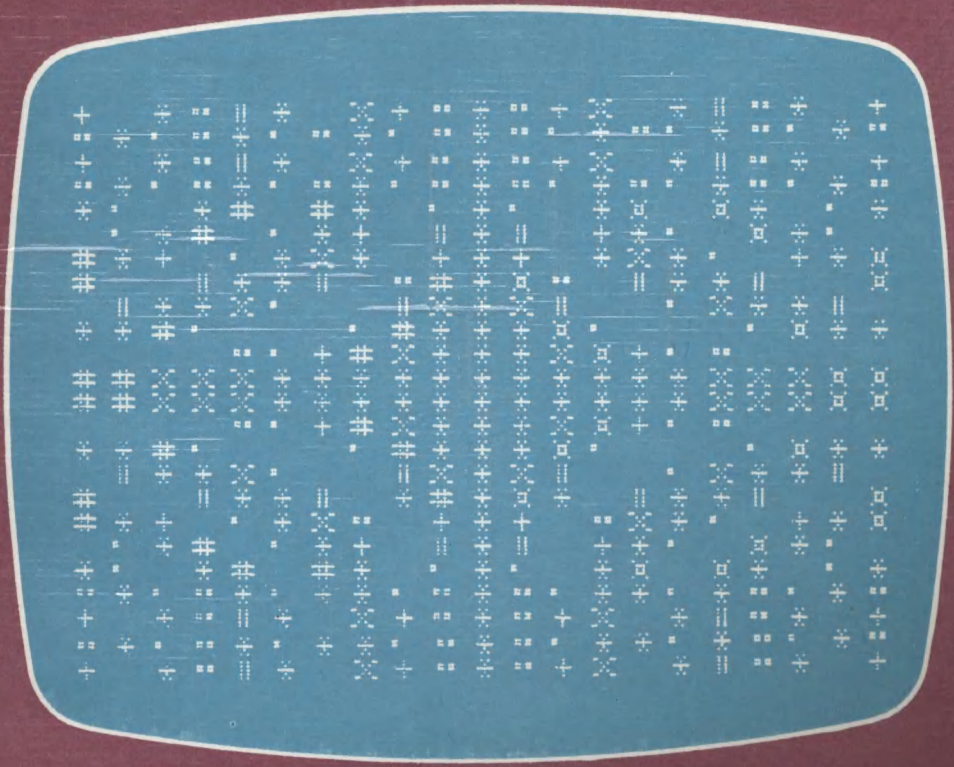
Ч. КОСНЁВСКИ

Ч. КОСНЁВСКИ

СОВРЕМЕННАЯ МАТЕМАТИКА
ПОПУЛЯРНАЯ СЕРИЯ

ЗАНИМАТЕЛЬНАЯ МАТЕМАТИКА И ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР



ЗАНИМАТЕЛЬНАЯ МАТЕМАТИКА И ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

Fun mathematics on your microcomputer

CZES KOSNIOWSKI

UNIVERSITY OF NEWCASTLE UPON TYNE

Cambridge University Press
Cambridge

London New York New Rochelle
Melbourne Sydney
1984

СОВРЕМЕННАЯ МАТЕМАТИКА

ПОПУЛЯРНАЯ СЕРИЯ

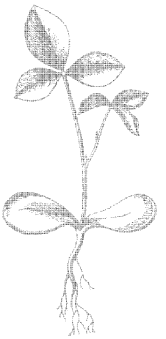
Ч. КОСНЁВСКИ

ЗАНИМАТЕЛЬНАЯ МАТЕМАТИКА И ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

Перевод с английского
И.А. МАХОВОЙ

под редакцией
Ю.М. БАЯКОВСКОГО

МОСКВА «МИР» 1987



ББК 22.1 + 32.97
К71
УДК 51 + 681.32

Коснёвски Ч.

К71 Занимательная математика и персональный компьютер. — Пер. с англ. — М.: Мир, 1987. — 192 с., ил.

Небольшая книга английского математика, знакомого советским читателям по переводу его «Начального курса алгебраической топологии» (М.: Мир, 1983). В ней в увлекательной форме предложена методика изучения различных разделов математики с помощью персональных компьютеров. Кратко излагаются теоретические сведения, формулируются занимательные задачи, для которых приводятся законченные программы на Бейсике с подробными пояснениями.

Для всех, кто осваивает программирование на Бейсике и изучает математику с помощью ЭВМ.

К $\frac{1702070000 - 243}{041(01) - 87}$ 47 — 87, ч. 1 **ББК 22.1 + 32.97**

Редакция литературы по математическим наукам

Оглавление

От редактора перевода 6
От автора 8

НАЧНЕМ ОТСЮДА
Введение 9

И ТАК ДАЛЕЕ, ДО БЕСКОНЕЧНОСТИ
О последовательностях и рядах 15

ВВЕРХ ДА ВНИЗ, ВОКРУГ ДА ОКОЛО
О функциях, графиках и полярных координатах 36

5 КМ НА СЕВЕР, 4 КМ НА ВОСТОК
О геометрии 56

ТЯНЕМ-ПОТЯНЕМ
О матрицах 62

ПО ПРАВИЛАМ ИГРЫ
О теории игр 70

НАВЕДЕМ ПОРЯДОК
О теории групп 92

ОЖИДАНИЕ
О теории массового обслуживания 109

ПРЕЛЕСТНЫЕ КАРТИНКИ
О функциях двух переменных 130

В ДВИЖЕНИИ
О дифференциальных уравнениях 145

ВСЕ БОЛЬШЕ И БОЛЬШЕ
Еще о дифференциальных уравнениях 165

ПРИЛОЖЕНИЕ. НОВЫЙ ОБЛИК ВАШЕЙ ПРОГРАММЫ
Замечания по изменению программ 182

ГДЕ ЭТО?
Указатель 191

От редактора перевода

Современные персональные компьютеры, легко умещающиеся на письменном столе, по своим рабочим характеристикам (объему оперативной и внешней памяти, скорости выполнения операций, эргономичности устройств ввода-вывода) нередко превосходят многотонные вычислительные гиганты конца 50-х, начала 60-х годов. Появление ПЭВМ, начало их массового производства, стремительное снижение стоимости оказалось для многих специалистов неожиданным и было воспринято как революция в информатике. Однако в массовом применении персональных компьютеров встретились серьезные затруднения.

Собственно компьютер, лишенный программного обеспечения, не более чем «железка», интересующая лишь профессионального программиста. Создание же развитого программного обеспечения требует серьезных затрат, порой во много крат превышающих стоимость самого компьютера. Даже современное образование ориентировано на обучение программированию, а не применению машин.

Таким образом, персональный компьютер нужен не сам по себе, а только вместе с удобными, надежными программами, позволяющими решать разнообразные задачи. Первый большой отклик персональные компьютеры получили в так называемой индустрии развлечений. Появилось огромное количество игровых программ. Однако не вся наша жизнь — игра. Ведется интенсивный поиск места для персональных компьютеров в профессиональных, инженерных применениях (этому посвящен, например, один из номеров журнала ТИИЭР: Малый тематический выпуск. Инженерные применения персональных компьютеров. — ТИИЭР, т. 73, № 12, 1985.). Книга Ч. Коснёвски показывает еще одну область применения ПЭВМ. Это — системы обучения. В данном случае речь идет об изучении некоторых разделов математики. На интересных, живых примерах читатель может познакомиться с последовательностями и рядами, матрицами и перестановками, теорией массового обслуживания и дифференциальными уравнениями.

Книга, безусловно, интересна и с другой точки зрения. С появлением ПЭВМ большое распространение получил язык программирования Бейсик. В книге имеется большое число реально работающих программ, которые будут полезны при изучении этого языка. Кстати, оби-

лие программ создало определенные затруднения при переводе. Тексты программ русского издания подготовлены переводчиком на ПЭВМ ДВК-2М в ВЦ Всесоюзного института повышения квалификации работников печати.

Материал книги в основном по силам любому старшекласснику, если у него есть доступ к машине. Поэтому можно надеяться, что книга будет полезна всем, кто хочет научиться работе на персональном компьютере, и, в частности, школьникам, изучающим информатику и вычислительную технику.

Ю. Баяковский

От автора

Автор весьма обязан фирмам ACT (Pulsar) и ACT (Sirius) за помощь и консультации. Поистине приятно найти компанию, которая не только поставляет прекрасную продукцию, но и снабжает ее великолепными сервисными программами. Как говорится, видеть — значит верить.

Приношу также благодарность многим предприятиям, охотно приславшим фотографии своих компьютеров.

Автор также хочет выразить признательность факультету вычислительной техники и информатики университета Ньюкасла-апон-Тайн за разрешение пользоваться компьютером и воспроизвести некоторые полученные на нем графики и диаграммы.

И наконец, приношу тысячи благодарностей Анне, Коре и Инге за неизменную поддержку и готовность выступать в качестве подопытных кроликов для многих программ этой книги.

Начнем отсюда

Введение

Из этой книги вы узнаете о том, сколь увлекательным может стать изучение чудесного мира математики с помощью персонального компьютера. Каждая глава знакомит с тем или иным разделом математики. Сначала излагаются основные идеи, затем они оформляются в виде программ для ЭВМ. Программы обычно представляют какую-нибудь занимательную игру. Иногда такая программа выдает яркие красивые картинки.

Если вы не возьмете в толк, как использовать свой компьютер, чтобы научиться чему-нибудь полезному и получить при этом удовольствие, то книга просто создана для вас. Если вы малосведущи и в математике, и в компьютерах, книга вас развлечет. Если вы имеете доступ к простенькому или достаточно совершенному персональному компьютеру, вычислительной машине для учебных целей или ЭВМ для решения экономических задач, во всех случаях книга будет вам полезна.

Все программы написаны на языке Бейсик, причем они составлены так, что их почти без переделок можно выполнять на вашем компьютере. Некоторые замечания по изменению программ приводятся в приложении.

Распечатки программ (их листинги) были получены непосредственно при прогоне исправной (отлаженной) программы. Программы проверялись на персональных компьютерах различных марок. Повсюду в книге приводятся фрагменты вычислений, выведенные на экран дисплея. Такое же изображение должно быть на вашем экране при выполнении программы.

Главы практически не зависят одна от другой, так что совсем необязательно читать их по порядку.

Эта книга научит вас некоторым полезным вещам как в информатике, так и в математике. Она станет для вас неисчерпаемым источником идей. Знания, которые вы приобретете, позволят вам самостоятельно писать и более сложные программы, чем помещенные в книге.

Далее вкратце изложим содержание глав.

И так далее, до бесконечности. Последовательности и ряды неожиданного возникают повсюду. Вы обнаружите их, изучая гусеницу,

ползущую по куску резины, процесс образования снежинок и поведение нерешительного, но любящего мужа. Игра, основанная на проверке умственных способностей (тесте интеллектуальности), займет вас на долгие часы.

Вверх да вниз, вокруг да около. В этой главе изучается построение графиков. В ней помещена программа, позволяющая «высветить» любую интересующую вас часть графика. Вы узнаете о «полярных координатах пчелы медоносной» и о том, как создавать нескончаемые количества прелестных картинок.

5 км на Север, 4 км на Восток. Здесь кратко излагаются сведения о геометрии и расстоянии. Приводится задача В ПОИСКАХ СОКРОВИЩ.

Тянем-потянем. Отсюда вы узнаете, как можно вытянуть или сплющить любую картинку на экране. Все это проделывается при помощи матриц.

По правилам игры. Один из разделов математики называется «теория игр». Здесь речь пойдет о стратегических играх — прекрасном источнике игр для персонального компьютера. Вы можете поиграть во вложение капитала и, может быть, даже заработать небольшую сумму в процессе одной игры.

Наведем порядок. Благодаря кубику Рубика теперь многим знакомо слово «группа». В этой главе представлены некоторые основные идеи теории групп, например, понятия перестановки, транспозиции и цикла. Включены задачи, основанные на теории групп. Все это столь же занимательно, сколь и познавательно.

Ожидание. Нам всем приходилось ждать обслуживания, стоя в очереди. Здесь вы познакомитесь с некоторыми идеями, таящимися в теории массового обслуживания, и получите большое удовольствие от игр, основанных на этих идеях.

Прелестные картинки. Это глава о представлении на плоскости трехмерных изображений и получении изолиний. Вы научитесь рисовать всевозможные красивые картинки — плоские и объемные.

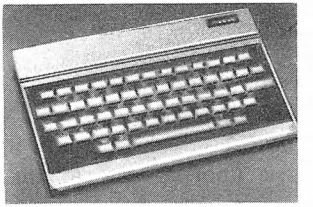
В движении. Все больше и больше. Что общего между подпрыгивающими мячами, самолетами и бактериями? Ответ: дифференциальные уравнения. Две эти главы посвящены дифференциальным уравнениям. Моделирование полета самолета или ракеты должно вас развлечь. Может быть, вам также удастся спасти целую палату больных, погибающих от жестокого вируса.

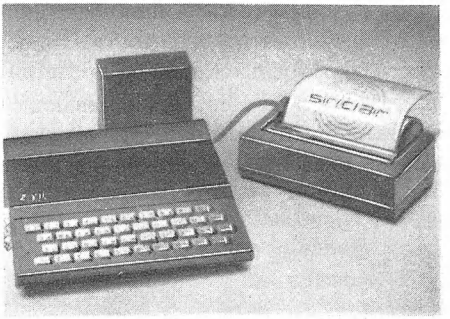
Приложение. Новый облик вашей программы. Все программы в книге составлены так, что их легко приспособить к вашему компьютеру. Эта глава поможет вам в случае необходимости кое-что переделать.

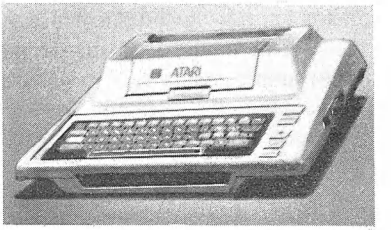
Замечания. Поскольку программы должны подходить для любого микрокомпьютера, они не столь «опрятны» и «коротки», как это могло быть, если бы они предназначались для конкретной машины. Многие программы можно сократить или даже улучшить. Переписывая программы и внося в них свои штрихи, вы получите дополнительную дозу развлечений.

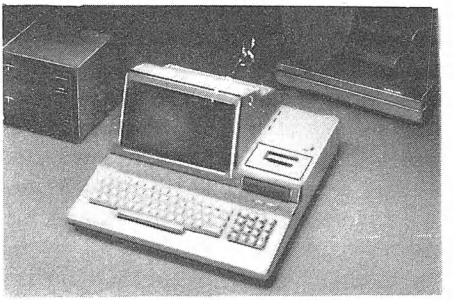
Несколько слов о моделировании при помощи компьютера. Некоторые программы книги в той или иной мере основаны на реальных ситуациях: они их *моделируют.* Однако *очень важно* не путать модель с самим явлением. Очевидно, что модель полезна лишь тогда, когда она хорошо согласуется с реальностью. Но модели всегда будут предсказывать и те вещи, которые не произойдут. А события, которые в действительности происходят, модель может и не прогнозировать. Польза от модели заключается в том, что она помогает понять, почему происходят те или иные явления: вам как бы предлагается ряд гипотез. Но не следует позволять модели сбить вас с толку.

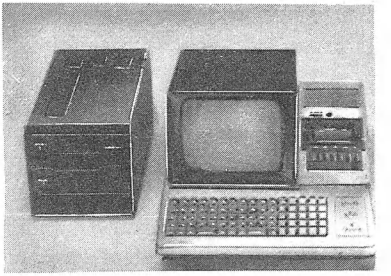
О некоторых популярных марках микрокомпьютеров. Персональные компьютеры бывают разной степени сложности и, соответственно, разной цены. Далее помещены фотографии некоторых марок персональных микрокомпьютеров. Они разбиты (весьма приблизительно) на три категории в зависимости от сложности и стоимости. Поместить фотографии всех существующих микрокомпьютеров не представляется возможным. Наши фотографии, хотя и не дают исчерпывающей информации, позволяют получить общее представление. Не огорчайтесь, если вашего компьютера здесь не оказалось — все равно книга будет полезной и заслуживающей внимания.

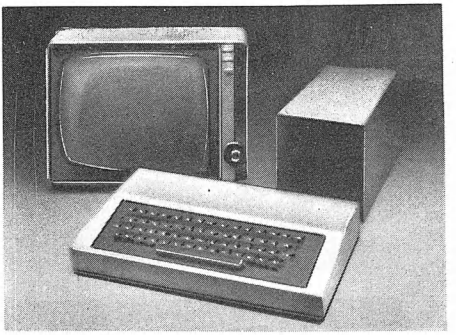


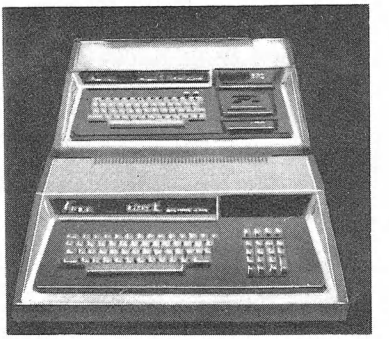

ZX81

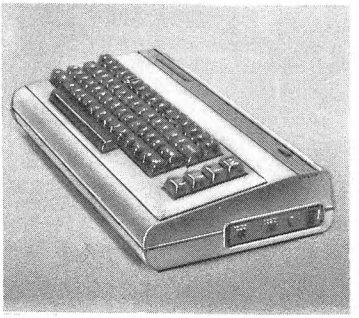

ATARI

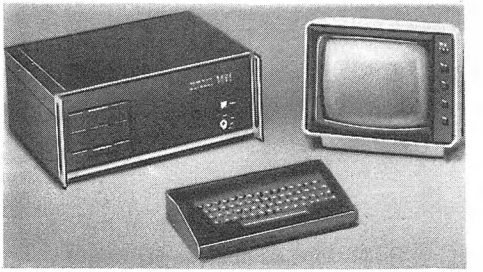


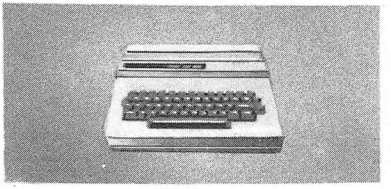


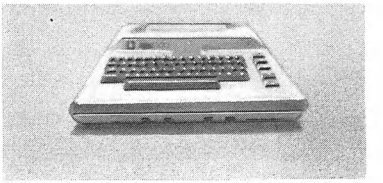


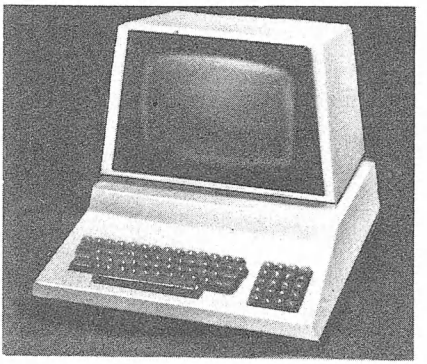


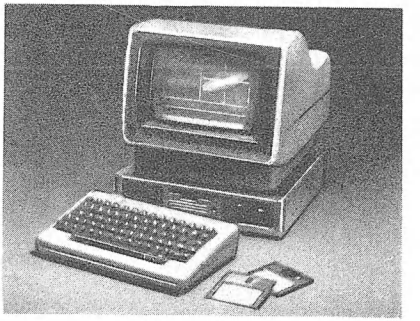


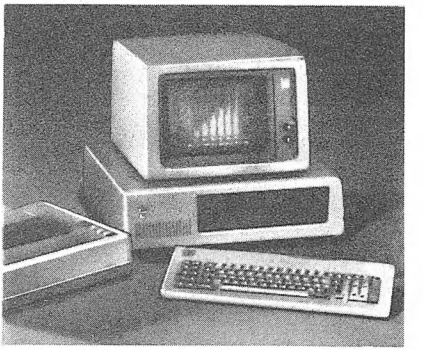


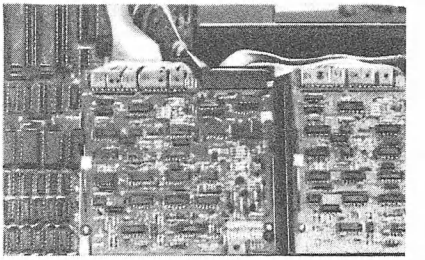












И так далее, до бесконечности

О последовательностях и рядах

Предположим, что на этой неделе вам выдали 10 фунтов с обещанием выплачивать каждую неделю на 2 фунта больше. Сколько вы получите на пятнадцатой неделе? Какую сумму вы заработаете за пятнадцать недель?

Не так уж трудно выписать список (или *последовательность*) чисел, обозначающих денежные суммы, полученные на каждой неделе:

10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38.

Такой список называется *арифметической последовательностью*, или *арифметической прогрессией*; числа последовательности называются ее *членами*. Продолжить этот список дальше вам, по-видимому, не составит труда.

Вообще говоря, арифметической последовательностью называется такая последовательность, в которой разность между каждым ее членом и членом, ему предшествующим, есть величина постоянная. Арифметическую последовательность можно представить в символьной записи:

$$A,\quad A + D,\quad A + 2*D,\quad A + 3*D,\quad \ldots,\quad A + (N - 1)*D.$$

Здесь A — первый член последовательности, D — разность и N — число членов последовательности. Знак многоточия используется для того, чтобы избавить нас от выписывания всех членов последовательности. Его следует читать: «и так далее, до». Приведем еще несколько примеров арифметических последовательностей:

5, 2, −1, −4, −7 (A = 5, D = −3, N = 5),
0, 0.5, 1, 1.5, 2, 2.5, 3 (A = 0, D = 0.5, N = 7),
1, 1.3, 1.6, 1.9, 2.2, 2.5 (A = 1, D = 0.3, N = 6),
1, 3, 5, 7, 9, ... , 99 (A = 1, D = 2, N = 50).

Нас часто будет интересовать сумма членов последовательности. Например,

$5 + 2 + -1 + -4 + -7$ и
$1 + 3 + 5 + 7 + 9 + \ldots + 99.$

Эти выражения называются *арифметическими рядами*. Существует формула для суммы арифметических рядов, позволяющая сократить вычисления. Вам, наверное, знакомо соотношение

$$1 + 2 + 3 + 4 + \ldots + N = N * (N + 1)/2.$$

Таким образом, сумма, скажем, первых 10 членов этого ряда равна 55. Ниже приводится общее выражение для сумм арифметических рядов:

$$A + (A + D) + (A + 2*D) + \ldots + (A + (N - 1)*D) = A*N + D*(N - 1)*N/2.$$

Возвращаясь к нашей исходной задаче, видим, что общая сумма, полученная в течение 15 недель, составит 10 * 15 + 2 * 14 * 15/2, что равно 360 фунтам.

При помощи программы АРИФМЕТИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ получать такие последовательности не составляет никакого труда. По команде INPUT вводятся первый член, разность и число членов последовательности, которое вам необходимо. На печать будет выдаваться сама последовательность и сумма всех ее членов. Напомним, что в этой программе каждый член последовательности вычисляется прибавлением разности D к предыдущему члену.

Нумерация команд в программе АРИФМЕТИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ и в следующих трех программах такова, что их все можно объединить в единую программу, озаглавленную ПОСЛЕДОВАТЕЛЬНОСТИ И РЯДЫ. Каждая из четырех программ не зависит одна от другой и будет исполняться без использования остальных.

```
1010 REM          ************************************
1020 REM          *                                  *
1030 REM          * АРИФМЕТИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ *
1040 REM          *                                  *
1050 REM          ************************************
1060 REM
1070 REM
1100 REM XXXXXXXXXXXXXXXXXXXX ВВОД ДАННЫХ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1110 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
1120 PRINT "АРИФМЕТИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ"
1130 PRINT
1140 PRINT "ПЕРВЫЙ ЧЛЕН";
1150 INPUT A
1160 PRINT
1170 PRINT "РАЗНОСТЬ";
1180 INPUT D
1190 PRINT
1200 PRINT "ЧИСЛО ЧЛЕНОВ";
1210 INPUT N
1300 REM XXXXXXXXXXXXXX СЧЕТ И ВЫВОД НА ЭКРАН XXXXXXXXXXXXXXXXXXXXXXXXXX
1310 PRINT
1320 PRINT "ПОСЛЕДОВАТЕЛЬНОСТЬ:"
1330 PRINT
```

Общие замечания по программированию см. в приложении.

```
1340 LET TERM=A
1350 LET SUM=0
1360 FOR I=1 TO N
1370 PRINT TERM;
1380 LET SUM=SUM+TERM
1390 LET TERM=TERM+D
1400 NEXT I
1410 PRINT
1420 PRINT
1430 PRINT "СУММА РАВНА";SUM
1440 PRINT
1500 REM XXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXXXX
1510 PRINT "ПОВТОРИТЬ? Y ИЛИ N"
1520 GET G$ : REM LET G$=INKEY$
1530 IF G$<>"Y" AND G$<>"N" THEN GOTO 1520
1540 IF G$="Y" THEN GOTO 1110
```

```
АРИФМЕТИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ

ПЕРВЫЙ ЧЛЕН ? -7

РАЗНОСТЬ ? 3

ЧИСЛО ЧЛЕНОВ ? 5

ПОСЛЕДОВАТЕЛЬНОСТЬ:

-7 -4 -1  2  5

СУММА РАВНА -5

ПОВТОРИТЬ?  Y ИЛИ N
```

Другую разновидность последовательностей представляют собой *геометрические последовательности*, или *геометрические прогрессии*. Приведем иллюстративный пример. Снова предположим, что на первой неделе вам выплатили 10 фунтов, но теперь надбавка за каждую неделю составляет 10% от полученного на предыдущей. Сколько вы получите на пятнадцатой неделе? Какую сумму вы заработаете за пятнадцать недель?

Часть последовательности чисел, представляющих собой денежные суммы, полученные на каждой неделе, приведены ниже (если мы хотим иметь ответ в фунтах, каждое число следует округлить до двух десятичных знаков):

10, 11, 12.1, 13.31, 14.641, 16.1051, 17.71561, ...

На пятнадцатой неделе будет выплачено 37.97 фунтов.

Геометрической последовательностью называется последовательность, в которой отношение между каждым ее членом и членом, ему предшествующим, есть величина постоянная. В нашем примере каждое число получается из предыдущего умножением на 1.1. В символьной записи геометрическую прогрессию можно представить следующим образом:

$$A,\ A*R,\ A*R^2,\ A*R^3,\ \ldots,\ A*R^{N-1}.$$

Здесь буквой A обозначен первый член последовательности, буквой R — ее знаменатель и буквой N — число членов последовательности. Как и обычно, R^2 (читается «R квадрат») означает R*R и R^3 (читается «R куб») означает R*R*R и т. д.

Вот еще несколько примеров геометрической прогрессии:

1, 2, 4, 8, 16, 32, 64	(A = 1, R = 2, N = 7),
4, 2, 1, 0.5, 0.25, 0.125, 0.0625	(A = 4, R = 0.5, N = 7),
1, −2, 4, −8, 16, −32, 64, −128	(A = 1, R = −2, N = 8).

Выражение в виде суммы членов геометрической последовательности называется *геометрическим рядом*. Приведем формулу для вычисления суммы геометрического ряда:

$$A + A*R + A*R^2 + \ldots + A*R^{N-1} = A*(1 - R^N)/(1 - R).$$

Таким образом, сумма денег, полученная за пятнадцать недель (начиная с 10 фунтов при 10-ти процентной еженедельной надбавке), составит $10*(1 - 1.1^{15})/(1 - 1.1)$, т. е. приблизительно 317.72 фунтов.

Вывод формулы для суммы геометрического ряда довольно прост. Обозначим сумму символом SUM; таким образом,

$$SUM = A + A*R + A*R^2 + A*R^3 + \ldots + A*R^{N-1}.$$

Умножив левую и правую части на R, получим второе соотношение:

$$R*SUM = A*R + A*R^2 + A*R^3 + \ldots + A*R^N.$$

Вычитая из первого равенства второе, придем к следующему:

$$SUM - R*SUM = A - A*R^N \quad \text{или}$$
$$(1 - R)*SUM = A*(1 - R^N).$$

Таким образом, мы получим, что $SUM = A*(1 - R^N)/(1 - R)$ при том условии, что R не равно единице (на нуль делить нельзя).

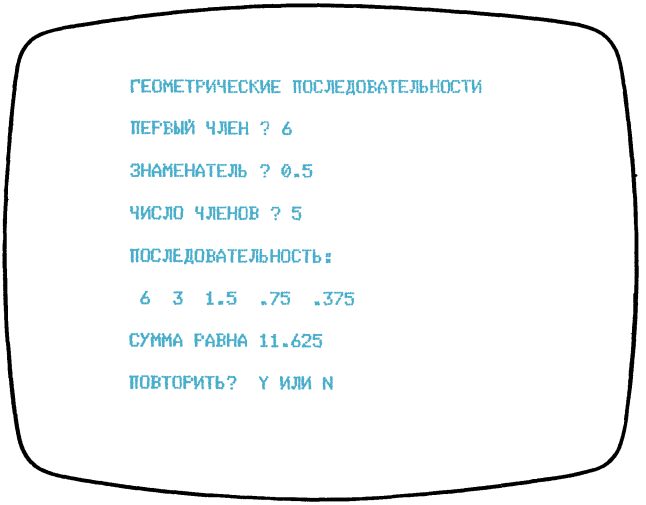
При помощи программы ГЕОМЕТРИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ можно получить сколько угодно геометрических последовательностей. По команде INPUT вводятся первый член, знаменатель и нужное вам число членов последовательности. На экран будет выводиться сама последовательность и сумма ее членов.

ГЕОМЕТРИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ — независимая программа, которая в то же время является второй частью программы ПОСЛЕДОВАТЕЛЬНОСТИ И РЯДЫ.

```
2010 REM            ************************************
2020 REM            *                                  *
2030 REM            * ГЕОМЕТРИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ *
2040 REM            *                                  *
2050 REM            ************************************
2060 REM
2070 REM
2100 REM XXXXXXXXXXXXXXXXXXXX ВВОД ДАННЫХ XXXXXXXXXXXXXXXXXXXXXXXXXXX
2110 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
2120 PRINT "ГЕОМЕТРИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ"
2130 PRINT
2140 PRINT "ПЕРВЫЙ ЧЛЕН";
2150 INPUT A
2160 PRINT
2170 PRINT "ЗНАМЕНАТЕЛЬ";
2180 INPUT R
2190 PRINT
2200 PRINT "ЧИСЛО ЧЛЕНОВ";
2210 INPUT N
2300 REM XXXXXXXXXXXXXXX СЧЕТ И ВЫВОД НА ЭКРАН XXXXXXXXXXXXXXXXXXXXXXXX
2310 PRINT
2320 PRINT "ПОСЛЕДОВАТЕЛЬНОСТЬ:"
2330 PRINT
2340 LET TERM=A
2350 LET SUM=0
2360 FOR I=1 TO N
2370 PRINT TERM;
2380 LET SUM=SUM+TERM
2390 LET TERM=TERM*R
2400 NEXT I
2410 PRINT
2420 PRINT
2430 PRINT "СУММА РАВНА "; SUM
2440 PRINT
2500 REM XXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXXXXXXX
2510 PRINT  "ПОВТОРИТЬ? Y ИЛИ N"
2520 GET G$ : REM LET G$=INKEY$
2530 IF G$<>"Y" AND G$<>"N" THEN GOTO 2520
2540 IF G$="Y" THEN GOTO 2110
```

Общие замечания по программированию см. в приложении.

Если вы попытаетесь исполнить (RUN) программу ГЕОМЕТРИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ с достаточно большим количеством членов последовательностей (при разных значениях N), то заметите, что довольно часто будет происходить переполнение (сообщение OVERFLOW). Однако при значениях R, заключенных между -1 и $+1$, переполнения никогда не будет. В самом деле, если взять бесконечно много членов геометрического ряда при значении R, заключенном меж-



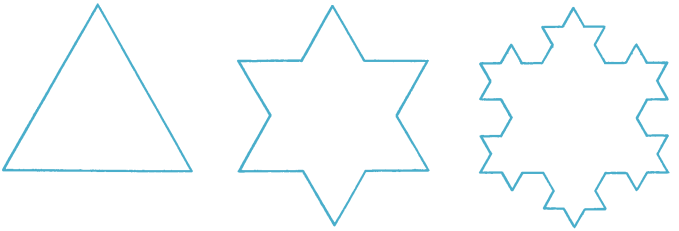
ду -1 и $+1$, то сумма будет равна $A/(1 - R)$, или, другими словами,

$$A + A*R + A*R^2 + A*R^3 + \ldots = A/(1 - R)$$
(если $-1 < R < 1$),

где «...» означает «и так далее, до бесконечности». Причина, грубо говоря, заключается в том, что R^N становится все меньше и меньше, когда N становится больше и больше. Ниже приводятся два классических примера:

$$1 + 1/2 + 1/4 + 1/8 + 1/16 + 1/32 + \ldots = 2 \quad (A = 1, R = 1/2),$$
$$1/10 + 1/100 + 1/1000 + 1/10000 + \ldots = 1/9$$
$$(A = 1/10, R = 1/10).$$

Снежинки. Правильный треугольник — это треугольник, все стороны которого имеют одну и ту же длину. Возьмем правильный треугольник с площадью в 1 $см^2$. Его можно следующим образом использовать для получения «снежинки». Пририсуем три меньших равносторонних треугольника по одному на каждой стороне (на средней трети). На каждой из полученных 12 сторон пририсуем по одному еще меньшему треугольнику (снова на средней трети стороны).



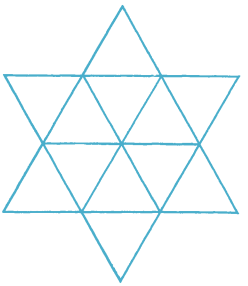
Этот процесс будем продолжать бесконечно долго. Какими будут площадь получившейся снежинки и ее периметр?

Прежде чем отвечать на эти вопросы, выясним, сколько сторон имеет снежинка на каждом этапе. Сначала у нее было 3 стороны. После первого добавления треугольников вместо одной стороны стало 4. Следовательно, каждый раз число сторон учетверяется. Итак, число сторон можно выразить такой последовательностью:

$$3,\ 3*4,\ 3*4^2,\ 3*4^3,\ 3*4^4,\ 3*4^5,\ \ldots\ .$$

Сразу же убеждаемся, что число сторон интересующей нас снежинки бесконечно велико.

Снежинка образуется добавлением треугольника к каждой стороне, так что выписанная выше последовательность дает также и число треугольников, добавляемое на каждом этапе. Начиная со второго этапа количество добавляемых треугольников каждый раз учетверяется. Площадь каждого нового треугольника составляет 1/9 от площади



предыдущего треугольника. Площадь первоначального треугольника была равна 1. После добавления трех треугольников площадь увеличилась на 3/9, т. е. на 1/3. Затем каждый раз будет добавляться вчетверо больше треугольников, чем на предыдущем этапе. Следовательно, площадь, добавляемая на каждом этапе, будет составлять 4/9 от площади, добавленной на предыдущем этапе. Общую площадь снежинки можно

выразить геометрическим рядом

$$1 + 1/3 + (1/3)*(4/9) + (1/3)*(4/9)^2 + (1/3)*(4/9)^3 + \ldots .$$

Если на время забыть про первую единицу, то перед нами геометрический ряд, первый член которого равен 1/3, а знаменатель 4/9. Сумма такого ряда равна (1/3)/(1 − 4/9), т. е. 3/5, или 0.6. Таким образом, площадь нашей снежинки выражается 1.6 квадратных сантиметров. Несмотря на то, что в построении снежинки приняло участие бесконечно много треугольников, ее площадь конечна (а не бесконечно велика).

Периметр снежинки, напротив, бесконечен. Чтобы убедиться в этом, обратим внимание на способ построения нашей снежинки. На каждом этапе количество сторон учетверяется. Но каждая новая сторона составляет 1/3 от прежней стороны (по построению). Таким образом, от этапа к этапу периметр увеличивается на 4/3. Пусть периметр исходного треугольника равен L. Длины периметров можно представить следующей (геометрической) последовательностью:

$$L,\ L*(4/3),\ L*(4/3)^2,\ L*(4/3)^3,\ L*(4/3)^4,\ \ldots .$$

Понятно, что периметр интересующей нас фигуры будет бесконечно велик.

Гармонический ряд. Знаменитая последовательность

1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, ...

не является ни арифметической, ни геометрической последовательностью. При суммировании членов этой последовательности получится так называемый *гармонический ряд*:

$$1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + \ldots .$$

Гармонический ряд вызывает у нас особый интерес, потому что его сумма бесконечна, несмотря на то, что его члены с каждым разом уменьшаются. В этом можно убедиться на основании следующих рассуждений:

Первый член 1.
Второй член 1/2.
Сумма следующих двух членов больше 1/2, поскольку каждый из них больше 1/4.
Сумма следующих четырех членов больше 1/2, поскольку каждый из них больше 1/8.
Сумма следующих восьми членов больше 1/2, поскольку каждый из них больше 1/16.
И т. д.

Таким образом, сумма больше, чем 1 + 1/2 + 1/2 + 1/2 + ... , и она становится все больше и больше при возрастании числа членов.

При помощи программы ГАРМОНИЧЕСКИЙ РЯД можно получать сумму произвольного числа членов гармонического ряда. Стóит лишь по команде INPUT ввести число членов, как программа определит сумму.

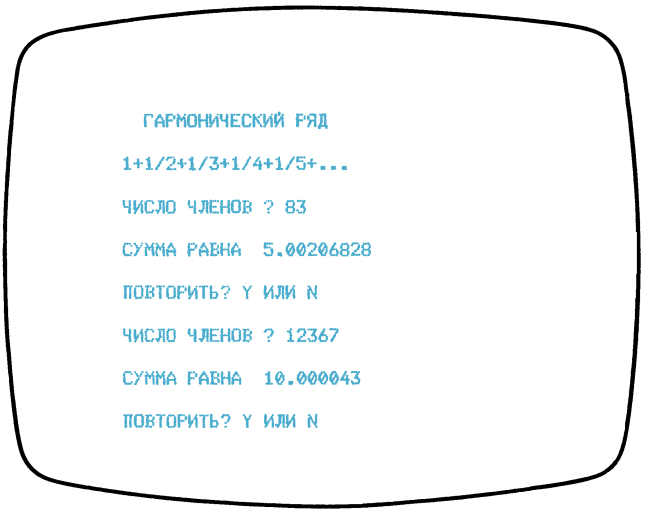
ГАРМОНИЧЕСКИЙ РЯД — независимая программа, которая представляет собой также и третью часть программы ПОСЛЕДОВАТЕЛЬНОСТИ И РЯДЫ.

```
3010 REM                    **********************
3020 REM                    *                    *
3030 REM                    * ГАРМОНИЧЕСКИЙ РЯД *
3040 REM                    *                    *
3050 REM                    **********************
3060 REM
3070 REM
3100 REM XXXXXXXXXXXXXXXXXXX ВВОД ДАННЫХ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
3110 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
3120 PRINT "  ГАРМОНИЧЕСКИЙ РЯД"
3130 PRINT
3140 PRINT "1+1/2+1/3+1/4+1/5+..."
3150 PRINT
3160 PRINT "ЧИСЛО ЧЛЕНОВ";
3170 INPUT N
3180 PRINT
3200 REM XXXXXXXXXXXXXXXXX СЧЕТ И ВЫВОД НА ЭКРАН XXXXXXXXXXXXXXXXXXXXXX
3210 LET SUM=0
3220 IF N<=0 THEN GOTO 3260
3230 FOR I=1 TO N
3240 LET SUM=SUM+1/I
3250 NEXT I
3260 PRINT "СУММА РАВНА "; SUM
3270 PRINT
3500 REM XXXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXXXX
3510 PRINT "ПОВТОРИТЬ? Y ИЛИ N"
3520 GET G$ : REM LET G$=INKEY$
3530 IF G$<>"Y" AND G$<>"N" THEN GOTO 3520
3540 IF G$="Y" THEN GOTO 3150
```

Общие замечания по программированию см. в приложении.

Заметим, что вычисление значения SUM в программе ГАРМОНИЧЕСКИЙ РЯД мы начинаем с наименьшего числа и затем прибавляем бóльшие числа. Это полезная подсказка, особенно при сложении огромных количеств чисел. Если вы начнете с бóльших чисел, то неизбежно столкнетесь с разного рода ошибками округления.

Сколько членов гармонического ряда нужно сложить, чтобы сумма превысила 5? А сколько, чтобы получить не меньше 10? Используя свой компьютер, вы легко получите ответ. Чтобы добраться до 5, вам понадобится 83 члена, а до 10 — целых 12 367. Значение 20 вы получите, сложив всего лишь 272 404 867 чисел. Разумеется, вам не стоит себя затруднять, проверяя это на своем компьютере.



Существует формула для вычисления приблизительного количества членов, необходимого для достижения некоторого заданного числа. Предположим, вы хотите узнать, сколько членов гармонического ряда нужно взять, чтобы получить в сумме число N. Возьмите число 2.71828 и умножьте его само на себя N раз, а затем разделите ответ на 1.781. Значение, полученное в результате, и будет числом членов, необходимым для получения в сумме N. Понятно, что с увеличением N это число очень быстро растет. На самом деле оно *увеличивается экспоненциально* (*растет по экспоненте*) — выражение, часто используемое для описания быстрого роста. Значение 2.71828 имеет непосредственное отношение к экспоненте — функции EXP(X), с которой вы, наверное, уже встречались на своем компьютере. Подробнее об этой функции будет рассказано несколько позже. Если вы попросите компьютер напечатать EXP(1) по команде PRINT EXP(1), то увидите число 2.71828183. Умножение этого числа само на себя N раз — не что иное, как вычисление EXP(N), которое компьютер проведет для вас без всякого труда. Таким образом, число членов гармонического ряда, необходимое для того, чтобы в сумме получалось N, приблизительно равно EXP(N)/1.781.

Ряд, у которого, подобно гармоническому, сумма выражается бесконечно большим числом, называется *расходящимся*. Если же сумма конечна, то ряд называется *сходящимся*. Например, ряд, составленный из членов геометрической прогрессии со знаменателем, заключенным между -1 и $+1$, сходящийся.

Гусеница на резине. Вот задача, в которой мы имеем дело с гармоническим рядом. Гусеница ползет по куску резины, стремясь достичь противоположного конца. Ползет она со скоростью 1 см/мин. Кусок резины имеет длину 7 см и может растягиваться до любой длины. Через минуту вы вытягиваете резину так, чтобы она удлинилась вдвое (т. е. стала 14 см в длину). Гусеница прочно держится на поверхности и продолжает двигаться, когда вы тянете резину. Она ползет все с той же скоростью. Еще через минуту вы снова вытягиваете резину так, что ее первоначальная длина утраивается (т. е. она становится равной 21 см). Гусеница продолжает ползти, а вы продолжаете каждую минуту тянуть резину, в четвертый раз уже удлинив ее в четыре раза. Доберется ли гусеница когда-нибудь до противоположного конца? Если да, то когда?

Гусеница все-таки достигнет противоположного конца через 10 часов с четвертью. А доберется она до конца благодаря тому, что гармонический ряд расходится! Чтобы убедиться в этом, посмотрим, что же происходит. За первую минуту наша гусеница проползла 1 см, что составляет 1/7 длины пути. В течение следующей минуты она проползает еще 1 см, что составляет 1/14 длины резины. Таким образом, в общей сложности пройдено 1/7 + 1/14 длины пути, что можно переписать в виде (1 + 1/2)/7. После удлинения резины до 21 см гусеница преодолеет еще 1 см, что составит 1/21 часть длины. Теперь гусеница в общей сложности проползла (1 + 1/2 + 1/3)/7 часть пути.

Нетрудно понять, что за N минут гусеница проползет

$$(1 + 1/2 + 1/3 + 1/4 + \ldots + 1/N)/7$$

часть пути. Здесь явно просматривается гармонический ряд. При значении N, приблизительно равном EXP(7)/1.781, выписанное выше выражение становится равным 7/7, т. е. к этому времени гусеница преодолеет весь путь.

Подобным вопросом мы можем задаться при любой конечной длине куска резины; в каждом случае гусеница может добраться до конца. Если длина равна 10 см, то гусеница преодолеет все расстояние через 8 дней. Если же первоначальная длина составляла 1 км (100 000 см), то ползти придется довольно долго. На самом деле даже дольше, чем предполагаемый возраст Вселенной, а резина за это время растянется до длины, превышающей размеры Вселенной. Понятно, что и гусенице, и вам подобная задача не по силам.

Любящий муж. Некий мужчина отправляется на службу, которая находится на расстоянии 1 км от дома. Дойдя до места работы, он вдруг вспоминает, что забыл поцеловать жену, и поворачивает назад. Пройдя полпути, он меняет решение, посчитав, что правильнее вер-

нуться на работу. Пройдя 1/3 км по направлению к конторе, он вдруг осознает, что будет настоящим подлецом, если так и не поцелует жену. На этот раз, прежде чем снова изменить мнение, он проходит 1/4 км. Так он продолжает метаться, и после N-го этапа, пройдя 1/N км, снова меняет решение. Где этот человек должен остановиться, чтобы окончательно не свихнуться?

Произойдет это на расстоянии 0.6931471806 км (приблизительно) от его дома — хотя до этого места ему придется добираться вечно. Ряд для вычисления расстояния выглядит так:

$$1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 - 1/8 + \ldots .$$

Он называется *знакочередующимся рядом*, поскольку его члены меняют знак. Этот ряд сходится и его сумма приблизительно равна 0.6931471806. Напишите программу для ее вычисления.

Ряд для вычисления PI. Число пи широко известно — по-видимому, это одно из самых знаменитых чисел, вычисленных приближенно. Оно определяется как отношение длины окружности к ее диаметру. Вот значение пи до 16-го десятичного знака: 3.1415926535897932. Это число равно приблизительно 22/7, но лишь приблизительно. Уже в 200 г. до н. э. греческий математик Архимед (тот самый, что закричал «Эврика!» во время купания) знал, что пи меньше чем 22/7, но больше, чем 223/71.

Значение пи (далее обозначаем его PI) можно вычислять с помощью рядов. Имеется несколько разных рядов, которые мы сейчас и рассмотрим; подробности, касающиеся их получения, опустим.

Около 1673 г. Готтфрид Лейбниц вывел следующую формулу для пи:

$$PI = 4 - 4/3 + 4/5 - 4/7 + 4/9 - 4/11 + 4/13 - \ldots .$$

Однако эта формула не очень удобна для вычислений. Пришлось бы использовать не одну тысячу членов, чтобы получить достаточно разумное приближение.

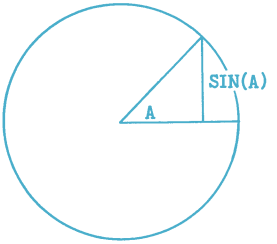
Позже два других математика — Шарп (около 1699 г.) и Эйлер (около 1736 г.) открыли значительно более полезные выражения:

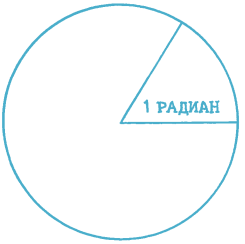
$$PI = 2*SQR(3)*[1 - 1/(3*3) + 1/(3^2*5) - 1/(3^3*7) + \ldots],$$
$$PI = SQR(6 + 6/1^2 + 6/2^2 + 6/3^2 + 6/4^2 + 6/5^2 + \ldots).$$

Вероятно, в памяти вашего микрокомпьютера есть значение PI, но все же интересно написать программу его вычисления при помощи одного из приведенных выше рядов. Обратите внимание, сколько членов ряда вам понадобится, чтобы получить значение, хранящееся в памяти вашего компьютера.

Некоторые другие ряды. Ряды совершенно неожиданно встречаются повсюду. Вот еще несколько примеров.

Дадим геометрическое определение синуса. Начертим окружность радиуса 1. Отложим угол A, как показано на рисунке слева. Вертикальный отрезок прямой H называется *синусом* угла A. Обозначим его SIN(A).





Обычно углы измеряют в градусах; в окружности содержится 360 градусов. Однако ваш компьютер (как и все математики) предпочитает радианы. Что же такое радиан? Отложим на нашей окружности дугу, равную по длине радиусу. Угол, стягиваемый этой дугой, и будет углом, равным 1 радиану; см. рисунок справа.

Длина окружности радиуса 1 равна 2*PI, так что 2*PI то же самое, что 360 градусов. По следующим формулам очень легко градусы превращать в радианы и обратно, радианы в градусы:

A градусов = A*PI/180 радиан
X радиан = X*180/PI градусов.

Когда вы встретите в книге SIN(X), знайте, что X выражено в радианах.

Геометрическая интерпретация SIN(X) очень удобна для нас, но совсем неудобна для компьютера. Если вы зададите команду PRINT SIN(0.5) и нажмете клавишу RETURN (ENTER или NEW LINE), то на экране появится значение SIN(0.5). Как узнаёт ваш компьютер, чему равен SIN(X)? Разумеется, он не может помнить все значения SIN(X) (для каждого X) и, ясное дело, он не вычерчивает окружности и не откладывает расстояния. Конечно же, он использует ряды. SIN(X) можно получить при помощи такого ряда:

$$SIN(X) = X - X^3/3! + X^5/5! - X^7/7! + X^9/9! - \ldots$$

для $-PI \leqslant X \leqslant PI$.

Обозначения 3!, 5! и т. д. читаются как «три факториал», «пять факториал» и т. д. Так обозначается произведение всех целых чисел от 1 до указанного целого включительно. Например, 2! = 1*2 = 2, 3! = = 1*2*3 = 6, 4! = 1*2*3*4 = 24 и т. д. Вообще говоря, факториалы — очень большие числа. Например, 35! выражается 40 цифрами: ваш компьютер переполнится и сообщит вам: OVERFLOW, если вы попытаетесь это число вычислить. Если мы нашли ответ, не означает ли это, что мы смогли вычислить факториал столь большого числа? Нет, конечно! Даже если бы мы все-таки как-то ухитрились его вычислить, то во всем мире не нашлось бы такого количества бумаги, чтобы записать это число. Понадобилось бы больше бумаги, чем во всех книгах всех библиотек мира.

Программа ВЫЧИСЛЕНИЕ СИНУСА — прекрасная иллюстрация того, как при помощи ряда можно вычислить SIN(X). Запускаем программу и вводим значение X и количество членов N. Обратите внимание на то, что обычно после пятого или шестого члена ответ не меняется.

ВЫЧИСЛЕНИЕ СИНУСА — независимая программа, которая представляет собой также и четвертую часть программы ПОСЛЕДОВАТЕЛЬНОСТИ И РЯДЫ.

```
4010 REM                  **********************
4020 REM                  *                    *
4030 REM                  * ВЫЧИСЛЕНИЕ СИНУСА *
4040 REM                  *                    *
4050 REM                  **********************
4060 REM
4070 REM
4100 REM XXXXXXXXXXXXXXXXXXXX ВВОД ДАННЫХ XXXXXXXXXXXXXXXXXXXXXXXXXXXX
4110 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
4120 PRINT "   ВЫЧИСЛЕНИЕ СИНУСА"
4130 PRINT
4140 PRINT "X-X↑3/3!+X↑5/5!-..."
4150 PRINT
4160 PRINT "ЗНАЧЕНИЕ X ";
4170 INPUT X
4180 PRINT
4190 PRINT "ЧИСЛО ЧЛЕНОВ"
4200 INPUT N
4300 REM XXXXXXXXXXXXXXXX СЧЕТ И ВЫВОД НА ЭКРАН XXXXXXXXXXXXXXXXXXXXXXXX
4310 LET R=INT(X/π )
4320 IF R<0 THEN LET R=R+1
4330 LET Y=X-R*π
4340 LET SUM=Y
4350 LET TERM=Y
4360 LET Y=Y*Y
4370 IF N<=0 THEN LET SUM=0
4380 LET M=3
4390 REM XXXXXXXXX ЦИКЛ XXXXXXXXXXXXX
4400 IF M>2*N THEN GOTO 4460
4410 LET DENOM=M*(M-1)
```

Из X вычитается число, кратное пи, до тех пор, пока результат не окажется между 0 и PI.

```
4420 LET TERM=-TERM*Y/DENOM
4430 LET SUM=SUM+TERM
4440 LET M=M+2
4450 GOTO 4400
4460 IF INT(R/2)<>R/2 THEN LET SUM=-SUM
4470 PRINT
4480 PRINT "СУММА РАВНА "; SUM
4490 PRINT
4500 REM XXXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXXXX
4510 PRINT "ПОВТОРИТЬ? Y ИЛИ N"
4520 GET G$ : REM LET G$=INKEY$
4530 IF G$<>"Y" AND G$<>"N" THEN GOTO 4520
4540 IF G$="Y" THEN GOTO 4150
```

Если X — нечетное, то знак суммы изменяется. Это следует из формул

SIN(PI + X) = −SIN(X),
SIN(2*PI + X) = SIN(X).

Дополнительные замечания см. в приложении.

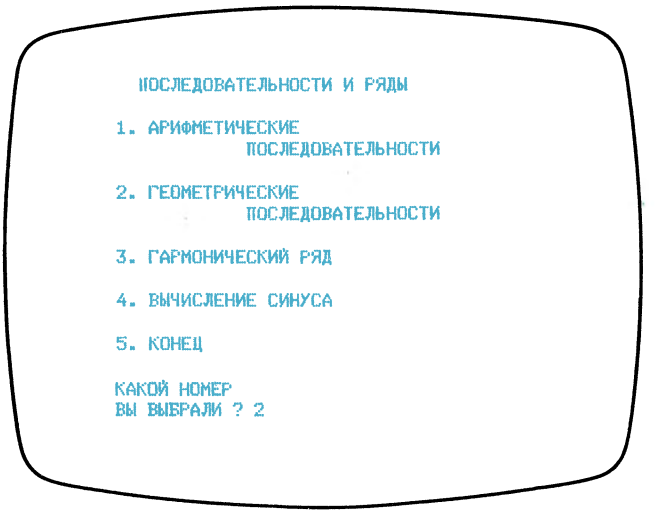
```
   ВЫЧИСЛЕНИЕ СИНУСА

X-X↑3/3!+X↑5/5!-...

ЗНАЧЕНИЕ X?  0.523599

ЧИСЛО ЧЛЕНОВ?  5

СУММА РАВНА   .500000194

ПОВТОРИТЬ?  Y ИЛИ N

ЗНАЧЕНИЕ X?  0.523599

ЧИСЛО ЧЛЕНОВ?  10

СУММА РАВНА   .500000194

ПОВТОРИТЬ? Y ИЛИ N
```

Заметьте, что если X очень мало, то X/3!, X/5!, и т. д. — совсем маленькие числа, так что SIN(X) приблизительно равно X. Можете проверить этот факт и на своем компьютере, вычисляя SIN(0.01) и SIN(0.001). В этом также можно убедиться, обратившись к геометрическому определению SIN(X). В круге радиуса 1 угол X (в радианах) — это длина дуги, тогда как SIN(X) — вертикальный отрезок (см. рисунок выше). Для очень маленьких углов эти две линии почти совпадают.

Мы уже упоминали, что в предыдущих четырех программах нумерация команд осуществлена таким образом, что все они могут быть объединены в одну программу, озаглавленную ПОСЛЕДОВАТЕЛЬНОСТИ И РЯДЫ. Чтобы можно было пользоваться любой из четырех программ по своему выбору, надо добавить всего несколько строк:

```
10 REM              **************************
20 REM              *                        *
30 REM              * ПОСЛЕДОВАТЕЛЬНОСТИ И РЯДЫ *
40 REM              *                        *
50 REM              **************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXX
110 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
120 PRINT " ПОСЛЕДОВАТЕЛЬНОСТИ И РЯДЫ "
130 PRINT
140 PRINT "1. АРИФМЕТИЧЕСКИЕ
150 PRINT "            ПОСЛЕДОВАТЕЛЬНОСТИ"
160 PRINT
170 PRINT "2. ГЕОМЕТРИЧЕСКИЕ
180 PRINT "            ПОСЛЕДОВАТЕЛЬНОСТИ"
190 PRINT
200 PRINT "3. ГАРМОНИЧЕСКИЙ РЯД"
210 PRINT
220 PRINT "4. ВЫЧИСЛЕНИЕ СИНУСА"
230 PRINT
240 PRINT "5. КОНЕЦ"
250 PRINT
260 PRINT "КАКОЙ НОМЕР"
270 PRINT "ВЫ ВЫБРАЛИ ";
280 INPUT X
290 IF X=5 THEN END : REM STOP
300 IF X<1 OR X>4 THEN GOTO 260
310 ON X GOTO 1110,2110,3110,4110
320 REM GOTO 1110+X*1000


1550 GOTO 110
2550 GOTO 110
3550 GOTO 110
4550 GOTO 110
```

Общие замечания по программированию см. в приложении.



Ранее мы вскользь упомянули об экспоненте. Ряд для ее вычисления выглядит так:

$$EXP(X) = 1 + X + X^2/2! + X^3/3! + X^4/4! + X^5/5! + \ldots .$$

Почему бы не написать программу для EXP(X) аналогично программе ВЫЧИСЛЕНИЕ СИНУСА и не добавить ее к программе ПОСЛЕДОВАТЕЛЬНОСТИ И РЯДЫ?

Вот еще несколько рядов, для которых можно написать программы и добавить их к ПОСЛЕДОВАТЕЛЬНОСТЯМ И РЯДАМ:

$$COS(X) = 1 - X^2/2! + X^4/4! - X^6/6! + X^8/8! - \ldots ,$$

$$ATN(X) = X - X^3/3! + X^5/5! - X^7/7! + X^9/9! - \ldots ,$$

$$-1 < X < 1,$$

$$LN(X) = 2^*(Y + Y^3/3 + Y^5/5 + Y^7/7 + \ldots),$$

где $Y = (X - 1)/(X + 1)$.

Заметим, что символом ATN(X) обозначен арктангенс угла X (на некоторых компьютерах используется обозначение ATAN(X)), а LN(X) — это натуральный логарифм числа X (на некоторых компьютерах — LOG(X)).

Тест интеллектуальности. В тестах проверки умственных способностей очень популярны вопросы о последовательностях чисел или букв. Например, такие:

15, 12, 9, 6 или Л, Й, З. Ё

Затем вас просят написать следующий член последовательности. Используя микрокомпьютер, можно получать всякого рода последовательности для интересных игр. Такие последовательности вам предоставит программа ТЕСТ ИНТЕЛЛЕКТУАЛЬНОСТИ. В программе предусмотрено достаточное количество последовательностей и организован случайный их выбор. В тесте не установлено каких-либо ограничений по времени. Их вы можете определять по своему усмотрению.

```
10 REM            ***************************
20 REM            *                         *
30 REM            * ТЕСТ ИНТЕЛЛЕКТУАЛЬНОСТИ *
40 REM            *                         *
50 REM            ***************************
60 REM
70 REM
100 REM XXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXX
110 LET J=1
120 LET SC=0                                    J — счетчик туров,
130 LET S=ASC("A") : REM LET S=CODE("A")        SC — счетчик баллов.
140 GOTO 290
200 REM XXXXXXXXXXXXXXXX ОСНОВНОЙ ЦИКЛ XXXXXXXXXXXXXXXXXXXXXXXXX
210 IF J>10 THEN GOTO 710
220 PRINT
```

```
230 PRINT
240 PRINT "ЧТОБЫ ПРОДОЛЖИТЬ,НАЖМИТЕ Y"
250 PRINT
260 GET G$ : REM LET G$=INKEY$
270 IF G$<>"Y" THEN GOTO 260
280 REM XXXXXXXXXXXX СЛУЧАЙНЫЕ ЧИСЛА XXXXXXXX
290 LET A=INT(RND(1)*10+1)
300 LET N=INT(RND(1)*4)
310 LET B=INT(RND(1)*10)
320 LET C=1
330 IF RND(1)<.5 THEN LET C=-1
340 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
350 PRINT "   ТЕСТ ИНТЕЛЛЕКТУАЛЬНОСТИ"
360 PRINT "    ПОСЛЕДОВАТЕЛЬНОСТЬ ";J
370 PRINT
380 PRINT "    КАКОЙ СЛЕДУЮЩИЙ ЧЛЕН?"
390 REM XXXXXXXX ВЫБОР ПОСЛЕДОВАТЕЛЬНОСТИ XXXXXX
400 ON INT(RND(1)*8)+1 GOSUB 1110,1310,1510,1710,1910,2110,2310,2510
410 REM GOSUB 1110+200*INT(RND*8)
420 PRINT
430 LET J=J+1
440 INPUT Y$
450 IF Y$=X$ OR " "+Y$=X$ THEN GOTO 560
460 GOSUB 911 : REM ЗВУКОВОЙ СИГНАЛ (ОШИБКА)
470 PRINT
480 PRINT "    НЕТ - ПОПРОБУЙТЕ ЕЩЕ"
490 PRINT
500 INPUT Y$
510 IF Y$=X$ OR " "+Y$=X$ THEN GOTO 560
520 GOSUB 911 : REM ЗВУКОВОЙ СИГНАЛ (ОШИБКА)
530 PRINT
540 PRINT "   ОТВЕТ ";X$
550 GOTO 210
560 GOSUB 921 : REM ЗВУКОВОЙ СИГНАЛ (ПРАВИЛЬНО)
570 PRINT
580 PRINT "    ПРАВИЛЬНО"
590 LET SC=SC+1
600 GOTO 210
700 REM XXXXXXXXXXXXXXX ОКОНЧАНИЕ XXXXXXXXXXXXXXXXXXXX
710 PRINT
720 PRINT
730 PRINT "ВАША ОЦЕНКА В 10-БАЛЛЬНОЙ"
740 PRINT
750 PRINT "ШКАЛЕ  ";SC
760 GOSUB 931 : REM ЗВУКОВОЙ СИГНАЛ (ОЦЕНКА)
770 PRINT
780 PRINT
790 PRINT "ПОВТОРИТЬ? Y ИЛИ N"
800 GET G$ : REM LET G$=INKEY$
810 IF G$<>"Y" AND G$<>"N" THEN GOTO 800
820 IF G$="Y" THEN GOTO 110
830 END : REM STOP
900 REM XXXXXXXXXXXX ЗВУКОВЫЕ ЭФФЕКТЫ XXXXXXXXXXXXXX
910 REM XXXXXXXXXXXXXXX ОШИБКА XXXXXXXXXX
911 POKE 36879,26+RND(1)*6
912 POKE 36878,15
913 T=150:GOSUB 941:T=130:GOSUB 941
914 POKE 36878,0
915 RETURN
920 REM XXXXXXXXXXXXX ПРАВИЛЬНО XXXXXXXXX
921 POKE 36879,26+RND(1)*6
```

A, N, B, C — различные случайные числа, используемые в последовательностях.

В строке 400 происходит выбор последовательности. Пользователям машины SINCLAIR следует обращаться к строке 410.

Вам предоставляется две возможности дать правильный ответ.

Здесь выставляется оценка в 10-балльной шкале, и вас спрашивают, не хотите ли вы еще поиграть.

```
922 POKE 36878,15
923 T=220:GOSUB 941:T=200:GOSUB 941
924 T=220:GOSUB 941
925 POKE 36878,0
926 RETURN
930 REM XXXXXXXXXXXXXXXX ОЦЕНКА XXXXXXXXXXX
931 POKE 36878,15
932 FOR K=1 TO SC
933 POKE 36879,26+RND(1)*6
934 T=220:GOSUB 941:T=0:GOSUB 941
935 NEXT K
936 POKE 36878,0:RETURN
940 REM XXXXXXXXXXXX ЗВУКОВОЙ СИГНАЛ XXXXXXX
941 POKE 36876,T+J
942 FOR I=1 TO 125:NEXT I
943 POKE 36876,0
944 RETURN
```

Дайте свои звуковые эффекты. Здесь рекомендовано три вида сигналов. Первый используется для указания неправильного ответа, второй — правильного, третий — для указания оценки.

```
1000 REM XXXXXXXXXXXXXXXX ПОДПРОГРАММЫ XXXXXXXXXXXXXXXXXXXXXXXXXXX
1100 REM XXXXXXXXXX АРИФМЕТИЧЕСКАЯ XXXXXXXXXX
1110 IF C<0 THEN LET B=B+80+INT(RND(1)*10)
1120 FOR I=1 TO 6
1130 PRINT STR$(A*I*C+B); : REM PRINT " ";A*I*C+B;
1140 NEXT I
1150 LET X$=STR$(A*I*C+B)
1160 PRINT
1170 RETURN
1300 REM XXXXXXXX БУКВЕННАЯ АРИФМЕТИЧЕСКАЯ XXXX
1310 FOR I=N TO N+6
1320 LET W=A*C*I+B
1330 LET W=W+S-26*INT(W/26)
1340 PRINT " ";CHR$(W);
1350 NEXT I
1360 LET W=A*C*I+B
1370 LET W=W+S-26*INT(W/26)
1380 LET X$=CHR$(W)
1390 PRINT
1400 RETURN
1500 REM XXXXXXXXXXX ГЕОМЕТРИЧЕСКАЯ XXXXXXXXX
1510 LET W=1
1520 LET N=N+2
1530 FOR I=1 TO 4
1540 LET W=W*N
1550 PRINT STR$(W+A) ; : REM PRINT " ";W+A;
1560 NEXT I
1570 LET X$=STR$(W*N+A)
1580 PRINT
1590 RETURN
```

На некоторых компьютерах перед и после числа дается пробел. Если на вашей машине это не предусмотрено, то более красивое представление можно получить при помощи команды PRINT, следующей за REM.

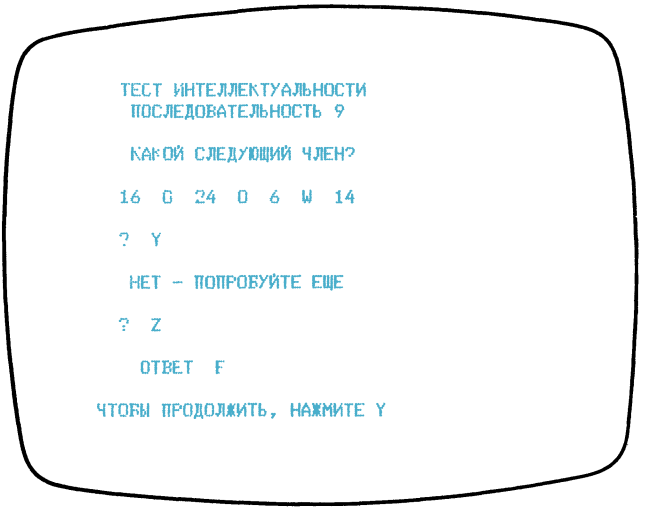
```
1700 REM XXXXXX ФИБОНАЧЧИ XXXXXXXX
1710 PRINT STR$(B);STR$(A); : REM PRINT " ";B;" ";A;
1720 PRINT STR$(B+A);STR$(B+2*A); : REM PRINT " ";B+A;" ";B+2*A;
1730 PRINT STR$(2*B+3*A);STR$(3*B+5*A):REM PRINT " ";2*B+3*A;" ";3*B+5*A
1740 LET X$=STR$(5*B+8*A)
1750 RETURN
1900 REM XXXXXX ФАКТОРИАЛЫ XXXXXXX
1910 LET W=A
1920 PRINT STR$(W+C); : REM PRINT " ";W+C;
1930 FOR I=1 TO 4
1940 LET W=W*I
1950 PRINT STR$(W+C); : REM PRINT " ";W+C;
1960 NEXT I
```

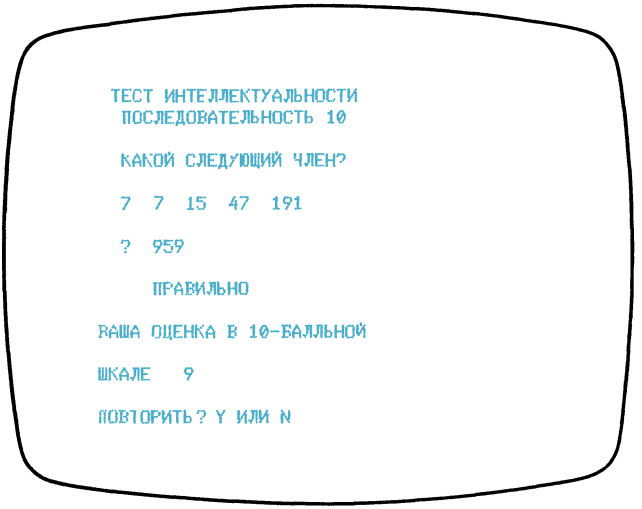
Комментарии (REM) можно благополучно опустить.

```
1970 PRINT
1980 LET X$=STR$(5*W+C)
1990 RETURN
2100 REM % БУКВЕННО-ЧИСЛОВАЯ АРИФМЕТИЧЕСКАЯ %
2110 FOR I=N TO N+6
2120 LET W=A*C*I+B
2130 LET W=W+S-26*INT(W/26)
2140 LET Y$=" "+CHR$(W)
2150 IF 2*INT(I/2)=I THEN LET Y$=STR$(W-S+1) : REM Y$=" "+STR$(W-S+1)
2160 PRINT Y$;
2170 NEXT I
2180 LET W=A*C*I+B
2190 LET W=W+S-26*INT(W/26)
2200 LET X$=CHR$(W)
2210 IF 2*INT(I/2)=I THEN LET X$=STR$(W-S+1)
2220 PRINT
2230 RETURN
2300 REM %%%%% ДВЕ АРИФМЕТИЧЕСКИЕ %%%%%%%%%%
2310 LET W=80+INT(RND(1)*10)
2320 FOR I=1 TO 6
2330 PRINT STR$(W); : REM PRINT " ";W;
2340 LET W=W+C*A
2350 LET C=-C
2360 LET A=A+B+1
2370 NEXT I
2380 PRINT
2390 LET X$=STR$(W)
2400 RETURN
2500 REM %%%%% ВОЗРАСТАЮЩАЯ АРИФМЕТИЧЕСКАЯ %%%%
2510 LET W=85+INT(RND(1)*10)
2520 LET D=1
2530 IF RND(1)<.5 THEN LET D=-1
2540 FOR I=1 TO 6
2550 PRINT STR$(W); : REM PRINT " ";W;
2560 LET W=W+C*A
2570 LET C=C*D
2580 LET A=A+I
2590 NEXT I
2600 PRINT
2610 LET X$=STR$(W)
2620 RETURN
```

Дополнительные замечания см. в приложении.

```
ТЕСТ ИНТЕЛЛЕКТУАЛЬНОСТИ
 ПОСЛЕДОВАТЕЛЬНОСТЬ 1

 КАКОЙ СЛЕДУЮЩИЙ ЧЛЕН?

 81  77  85  73  89  69

 ?  93

     ПРАВИЛЬНО

ЧТОБЫ ПРОДОЛЖИТЬ, НАЖМИТЕ Y
```


ТЕСТ ИНТЕЛЛЕКТУАЛЬНОСТИ
ПОСЛЕДОВАТЕЛЬНОСТЬ 9
КАКОЙ СЛЕДУЮЩИЙ ЧЛЕН?
16 G 24 O 6 W 14
? Y
НЕТ - ПОПРОБУЙТЕ ЕЩЕ
? Z
ОТВЕТ F
ЧТОБЫ ПРОДОЛЖИТЬ, НАЖМИТЕ Y


ТЕСТ ИНТЕЛЛЕКТУАЛЬНОСТИ
ПОСЛЕДОВАТЕЛЬНОСТЬ 10
КАКОЙ СЛЕДУЮЩИЙ ЧЛЕН?
7 7 15 47 191
? 959
ПРАВИЛЬНО
ВАША ОЦЕНКА В 10-БАЛЛЬНОЙ
ШКАЛЕ 9
ПОВТОРИТЬ? Y ИЛИ N

Вверх да вниз, вокруг да около

О функциях, графиках и полярных координатах

Площадь A круга радиуса R определяется по формуле A = = PI*R*R. Каждому значению R соответствует некоторое значение A. Мы говорим, что A есть *функция* аргумента R.

Вероятно, вы уже встречали выражение вида

Y = X*X, Y = SIN(X), Y = SQR(X), Y = SQR(X*X + 2), ...

Все это примеры функций. В каждом случае Y есть функция от X. Вводится значение X и на выходе получается значение Y. Заметим, что для некоторых функций не все значения X можно вводить. Например, для функции SQR(X) нельзя вводить отрицательные числа.

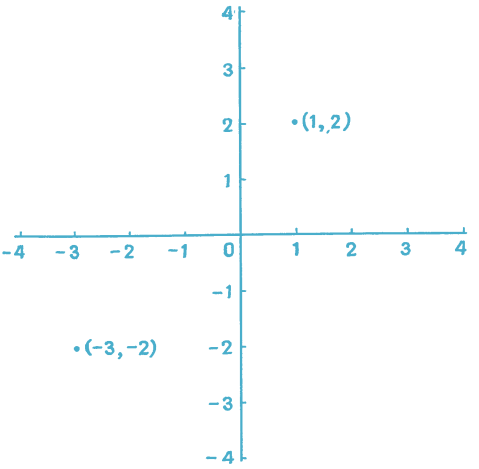
Абсолютно безразлично, какие буквы использовать для обозначения функции и аргумента. Так, каждая из приведенных ниже записей

B = A*A, W = U*U, Y = X*X

соответствует одной и той же функции возведения числа в квадрат. Как правило, используют буквы Y и X.

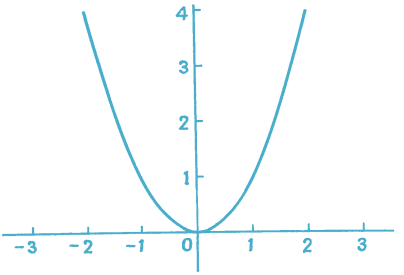
Представление функции в виде формулы само по себе не очень иллюстративно. Например, вот функция Y = X*X*SIN(1/X). Что можно сказать о ее поведении при больших значениях X? А при малых X? «Лучше один раз увидеть, чем сто раз услышать», поэтому для лучшего понимания функций мы часто рисуем «картинку» — график функции.

Графики и координаты. Точку на плоскости (или на экране дисплея вашего компьютера) обычно представляют парой чисел (X, Y). Эта пара означает расстояние вашей точки от неких осей. Число X задает расстояние по горизонтали, а Y — по вертикали. Расстояние по горизонтали принято отсчитывать слева направо, а по вертикали — снизу вверх. В противоположных направлениях откладываются отрицательные значения (см. рисунок).



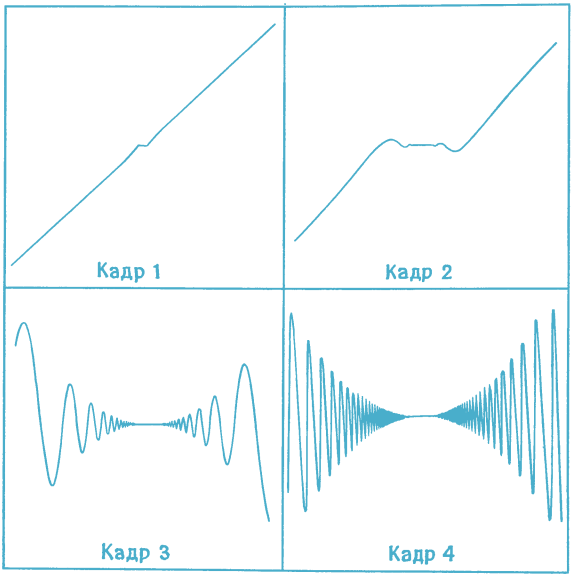
Пара (X, Y) называется *декартовыми координатами*, или просто *координатами* точки. Свое название эти координаты получили по имени французского математика Рене Декарта (1596—1650) («Я мыслю, значит, я существую», — его высказывание), кому мы обязаны этим методом описания местонахождения точки.

Построить *график* функции — это значит: для каждого значения X из некой заданной области определения вычислить соответствующее значение Y и построить точку с координатами (X, Y). На рисунке показан график функции Y = X*X.



Довольно интересно ведет себя функция Y = X*X*SIN(1/X). Ниже представлен ее график для четырех различных областей определения. Если вы построите график для значений X, заключенных между −10 и 10, то он будет очень похож на прямую, проведенную по линей-

ке (см. первый кадр). Однако вблизи 0 создается такое впечатление, что линейка слегка сместилась. Но как раз в этом месте происходит нечто интересное. На следующем кадре сделано увеличение и построен график для X между −1 и 1. Здесь уже начинает проясняться, что происходит в окрестности X = 0. Дальнейшее увеличение дают кадры 3 и 4, на которых показаны соответственно области от −0.1 до 0.1 и от −0.025 до 0.025.



Почему при больших значениях X график функции X*X**SIN(1/X) выглядит почти как прямая? Чтобы ответить на этот вопрос, представим нашу функцию в виде X*SIN(1/X)/(1/X) и обозначим 1/X буквой U. Теперь мы имеем дело с функцией X*SIN(U)/U, и при больших X величина 1/X (или U) мала. Вспомним, что при малых значениях U значение SIN(U)/U приблизительно равно 1, так что X*SIN(U)/U приблизительно равно X. Таким образом, при больших значениях X получаем (приблизительно) график Y = X, т. е. прямую линию.

Если вы взглянете на графики для малых значений X, то заметите интересную особенность (лучше прищуриться!). Самые верхние точки графика выстраиваются в линию X*X, самые нижние — в линию −X*X, а весь остальной график пульсирует вверх-вниз между этими кривыми. Дело в том, что приближаясь к нулю, X проходит через точки −1/PI, −1/(2PI), −1/(3PI) и т. д. В этих точках SIN(1/X) принимает значения ±1, а функция X*X*SIN(1/X) — либо X*X, либо −X*X.

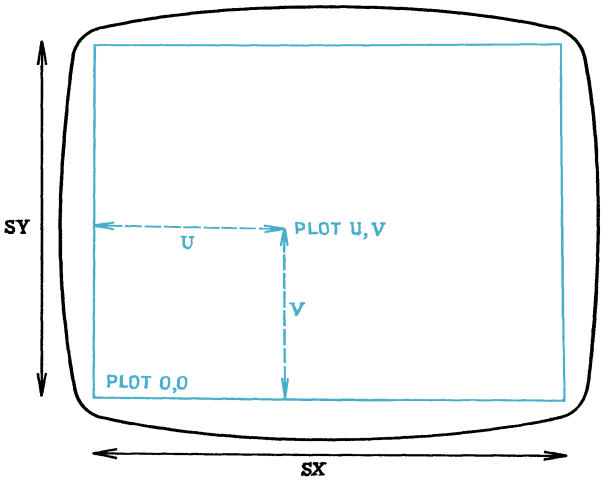
При помощи своего компьютера вы сможете весьма эффективно рисовать графики функций. Более того, компьютер поможет вам «как через лупу» посмотреть на любую интересующую вас часть графика, если вы хотите узнать о функции больше. Даже графики, построенные при помощи компьютера ZX 81, обнаруживают удивительные вещи.

Вооружившись программой ПОСТРОЕНИЕ ГРАФИКОВ, вы сможете изучать графики функций при каких угодно значениях X. Вас может заинтересовать «глобальный вид» графика, т. е. достаточно обширная область его определения, или же вы захотите исследовать поведение графика «локально», т. е. в некоторой узкой области значений X.

Возможности Бейсика на разных машинах различные, так что написать общую программу представляется чрезвычайно затруднительным. А когда дело доходит до составления программы построения графиков, положение становится просто плачевным. На некоторых микрокомпьютерах имеются такие команды, как PLOT и DRAW, тогда как на многих машинах они отсутствуют. На последних графики строятся посредством POKE, и способ, при помощи которого это делается, зависит исключительно от марки машины. Программа ПОСТРОЕНИЕ ГРАФИКОВ была написана таким образом, чтобы ее можно было приспособить для вашего персонального компьютера.

Первое, что надо сделать, — это определить разрешающую способность вашего экрана в терминах элементов изображения или точек, которые можно будет построить. Например, машина ZX 81 имеет разрешающую способность 64 элемента изображения по горизонтали и 44 по вертикали, тогда как экран ZX Spectrum имеет разрешающую способность 256 × 176. Эти числа, уменьшенные на 1 или на 2, обозначаются в программе построения графиков соответственно символами SX и SY (см. рисунок).

Чтобы вам проще было адаптировать программу ПОСТРОЕНИЕ ГРАФИКОВ для вашей машины, в книгу включены два дополнительных листинга. SINCLAIR GRAPH PLOTTING предназначается для иллюстрации того, как приспособить ПОСТРОЕНИЕ ГРАФИКОВ к машине, в которой предусмотрена команда PLOT. Программа VIC 20 GRAPH PLOTTING демонстрирует, как приспособить ПОСТРОЕНИЕ ГРАФИКОВ к машине, в Бейсике которой отсутствует команда PLOT. Дополнительные разъяснения можно найти в приложении.




```
10 REM             ***********************
20 REM             *                     *
30 REM             * ПОСТРОЕНИЕ ГРАФИКОВ *
40 REM             *                     *
50 REM             ***********************
60 REM
70 REM
100 REM XXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET SX=176 : REM РАЗМЕР ЭКРАНА ПО ГОРИЗОНТАЛИ
130 LET SY=160 : REM РАЗМЕР ЭКРАНА ПО ВЕРТИКАЛИ
140 LET HY=SY/2
150 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
160 PRINT "     ПОСТРОЕНИЕ ГРАФИКОВ"
170 PRINT
180 PRINT "1.Y = X*X*SIN(1/X)"
190 PRINT
200 PRINT "2.Y = X*SIN(1/X)"
210 PRINT
220 PRINT "3.Y = SQR(X*X+2)"
230 PRINT
240 PRINT "4.Y = COS(X*EXP(-X/5))"
250 PRINT
260 PRINT "5.Y = 6+2*X*X-X*X*X*X"
270 PRINT
280 PRINT "УКАЖИТЕ НОМЕР"
290 PRINT "УРАВНЕНИЯ";
300 INPUT N
310 IF N=1 THEN DEF FNA(X)=X*X*SIN(1/X)
320 IF N=2 THEN DEF FNA(X)=X*SIN(1/X)
330 IF N=3 THEN DEF FNA(X)=SQR(X*X+2)
340 IF N=4 THEN DEF FNA(X)=COS(X*EXP(-X/5))
```

SX и SY обозначают количество точек, которое можно изобразить на экране. Укажите значения, соответствующие вашему экрану.

```
350 IF N=5 THEN DEF FNA(X)=6+2*X*X-X*X*X*X
360 PRINT
370 PRINT "ОБЛАСТЬ ИЗМЕНЕНИЯ X"
380 PRINT
390 PRINT "НАИМЕНЬШЕЕ ЗНАЧЕНИЕ";
400 INPUT A
410 PRINT
420 PRINT "НАИБОЛЬШЕЕ ЗНАЧЕНИЕ";
430 INPUT B
440 PRINT
450 IF A)=B THEN PRINT "ОШИБКА - ПОПРОБУЙТЕ ЕЩЕ"
460 IF A)=B THEN GOTO 360
500 REM XXXXXXXX ВЫЧИСЛЕНИЕ ОБЛАСТИ ЗНАЧЕНИЙ Y XXXXXXXXXXXXXXXXXXX
510 PRINT "ВЫЧИСЛЕНИЕ ОБЛАСТИ ЗНАЧЕНИЙ Y"
520 LET C=(B-A)/100
530 LET M=1.0E-30
540 FOR X=A TO B STEP C
550 IF X=0 THEN GOTO 580
560 LEI Y=ABS(FNA(X))
570 IF M<Y THEN LET M=Y
580 NEXT X
590 PRINT "ГОТОВ К ПОСТРОЕНИЮ ГРАФИКОВ"
600 FOR I=1 TO 1000
610 NEXT I
620 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
630 GOSUB 1010 : REM ПОДГОТОВИТЬ ЭКРАН, ЕСЛИ НУЖНО
700 REM XXXXXXXXXXXXXX ПОСТРОЕНИЕ ГРАФИКОВ XXXXXXXXXXXXXXXXXXXXX
710 LET C=C/10 : REM ПОПРОБУЙТЕ C/5 ИЛИ C/20
720 FOR X=A TO B STEP C
730 IF X=0 THEN GOTO 790
740 LET Y=FNA(X)
750 LET U=SX*(X-A)/(B-A)
760 LET V=HY+HY*Y/M
770 IF V<0 OR V)SY THEN GOTO 790
780 GOSUB 1110 : REM PLOT U,V
790 NEXT X
800 REM XXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXX
810 GET G$ : REM LET G$=INKEY$
820 IF G$="" THEN GOTO 810
830 GOSUB 1210 : REM ВОССТАНОВИТЬ ЭКРАН, ЕСЛИ НУЖНО
840 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
850 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
860 GET G$ : REM LET G$=INKEY$
870 IF G$<>"Y" AND G$<>"N" THEN GOTO 860
880 IF G$="Y" THEN GOTO 150
890 END : REM STOP
1000 REM XXXXXXXXXXXXXXXXX ПОДГОТОВКА ЭКРАНА XXXXXXXXXXXXXXXXXXXXXXXX
1010 REM RETURN
1100 REM XXXXXXXXXXX ПОСТРОЕНИЕ ГРАФИКОВ ПРИ ПОМОЩИ POKE XXXXXXXXXXXXXX
1110 RETURN
1200 REM XXXXXXXXXXXXXXXX ВОССТАНОВЛЕНИЕ ЭКРАНА XXXXXXXXXXXXXXXXXXXXX
1210 RETURN
```

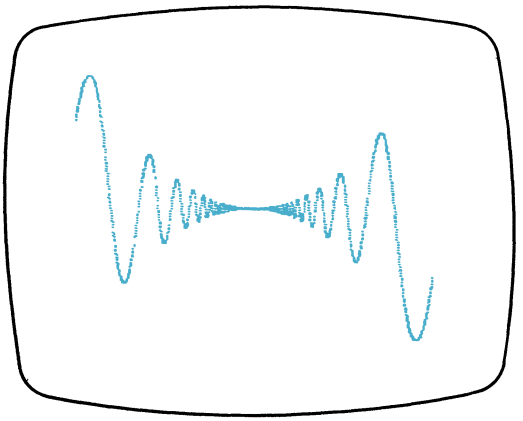
Вычисляется область значений Y (приблизительно). Затем выбирается такой масштаб, чтобы экран был заполнен целиком.

Здесь происходит непосредственно построение (PLOT). Проверка делается для того, чтобы убедиться, что будут построены только необходимые точки.

Сначала испробуйте программу на функции X*X*SIN(1/X). Затем поэкспериментируйте с другими функциями. Может быть, вам захочется включить в программу такие функции:

$$Y = SQR(X*X + 2)*SIN(X), \quad Y = COS(X*EXP(-X/5))*SIN(X),$$
$$Y = X*SIN(1/X)*SIN(1/X), \quad Y = SIN(X/LN(ABS(X) + 1.1)).$$

```
        ПОСТРОЕНИЕ ГРАФИКОВ

1.Y = X*X*SIN(1/X)

2.Y = X*SIN(1/X)

3.Y = SQR(X*X+2)

4.Y = COS(X*EXP(-X/5))

5.Y = 6+2*X*X-X*X*X*X

УКАЖИТЕ НОМЕР
УРАВНЕНИЯ?  1

ОБЛАСТЬ ИЗМЕНЕНИЯ X

НАИМЕНЬШЕЕ ЗНАЧЕНИЕ?  -0.1

НАИБОЛЬШЕЕ ЗНАЧЕНИЕ?   0.1

ВЫЧИСЛЕНИЕ ОБЛАСТИ ЗНАЧЕНИЙ Y
```



```
10 REM                  ******************
20 REM                  *    SINCLAIR    *
30 REM                  * GRAPH PLOTTING *
40 REM                  *                *
50 REM                  ******************
60 REM
100 REM %%%%%%%%%%%%%%%%%%%% SETTING UP %%%%%%%%%%%%%%%%%%%%
120 LET SX=62
130 LET SY=42
140 LET HY=SY/2
150 CLS
160 PRINT "    GRAPH PLOTTING"
170 PRINT
```

SX и SY обозначают количество точек, которое можно изобразить на экране. Укажи-

```
180 PRINT "1.Y = X*X*SIN(1/X)"
190 PRINT
200 PRINT "2.Y = X*SIN(1/X)"
210 PRINT
220 PRINT "3.Y = SQR(X*X+2)"
230 PRINT
240 PRINT "4.Y = COS(X*EXP(-X/5))"
250 PRINT
260 PRINT "5.Y = 6+2*X*X-X*X*X*X"
270 PRINT
280 PRINT "TYPE IN THE NUMBER OF"
290 PRINT "THE EQUATION ";
300 INPUT N
305 PRINT N
310 IF N=1 THEN LET A$="X*X*SIN(1/X)"
320 IF N=2 THEN LET A$="X*SIN(1/X)"
330 IF N=3 THEN LET A$="SQR(X*X+2)"
340 IF N=4 THEN LET A$="COS(X*EXP(-X/5))"
350 IF N=5 THEN LET A$="6+2*X*X-X*X*X*X"
360 PRINT
370 PRINT "VALUES OF X FOR PLOT"
380 PRINT
390 PRINT "LOWEST VALUE ";
400 INPUT A
405 PRINT A
410 PRINT
420 PRINT "HIGHEST VALUE ";
430 INPUT B
435 PRINT B
440 PRINT
450 IF A>=B THEN PRINT "ERROR - TRY AGAIN"
460 IF A>=B THEN GOTO 360
500 REM XXXXXXXXXXXXXXXX CALCULATING RANGE OF Y XXXXXXXXXXXXXXX
510 PRINT "CALCULATING RANGE OF Y"
520 LET C=(B-A)/100
530 LET M=1.0E-30
535 REM FAST FOR ZX81
540 FOR X=A TO B STEP C
550 IF X=0 THEN GOTO 580
560 LET Y=ABS(VAL(A$))
570 IF M<Y THEN LET M=Y
580 NEXT X
585 REM SLOW FOR ZX81
590 PRINT "READY FOR PLOTTING"
600 FOR I=1 TO 100
610 NEXT I
620 CLS
700 REM XXXXXXXXXXXXXXXXXXXX PLOTTING XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
710 LET C=C/10 : REM TRY C/5 OR C/20
720 FOR X=A TO B STEP C
730 IF X=0 THEN GOTO 790
740 LET Y=VAL(A$)
750 LET U=SX*(X-A)/(B-A)
760 LET V=HY+HY*Y/M
770 IF V<0 OR V>SY THEN GOTO 790
780 PLOT U,V
790 NEXT X
800 REM XXXXXXXXXXXXXXXX ENDING AND ANOTHER GO XXXXXXXXXXXXXXXXXXXXXXXX
810 LET G$=INKEY$
820 IF G$="" THEN GOTO 810
```

те соответствующие вашему экрану значения. Для ZX Spectrum это SX = 255, SY = 175.

Вычисляется область значений Y (приблизительно). Затем выбирается такой масштаб, чтобы экран был заполнен целиком.

Здесь происходит непосредственно построение (PLOT). Проверка делается для того, чтобы убедиться, что будут построены только необходимые точки.

```
850 PRINT " ANOTHER GO?  Y OR N"
860 LET G$=INKEY$
870 IF G$<>"Y" AND G$<>"N" THEN GOTO 860
880 IF G$="Y" THEN GOTO 150
890 STOP


10 REM                 ******************
20 REM                 *     VIC 20     *
30 REM                 * GRAPH PLOTTING *
40 REM                 *                *
50 REM                 ******************
60 REM
100 REM %%%%%%%%%%%%%%%%% SETTING UP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110 LET CS$=CHR$(147) : REM CODE FOR CLEAR SCREEN
120 LET SX=176 : REM SCREEN SIZE HORIZ
130 LET SY=160 : REM SCREEN SIZE VERT
140 LET HY=SY/2
145 GOSUB 1310 : REM EXTRA SETTINGS
150 PRINT CS$
160 PRINT "      GRAPH PLOTTING"
170 PRINT
180 PRINT "1.Y = X*X*SIN(1/X)
190 PRINT
200 PRINT "2.Y = X*SIN(1/X)"
210 PRINT
220 PRINT "3.Y = SQR(X*X+2)"
230 PRINT
240 PRINT "4.Y = COS(X*EXP(-X/5))"
250 PRINT
260 PRINT "5.Y = 6+2*X*X-X*X*X*X"
270 PRINT
280 PRINT "TYPE IN THE NUMBER OF"
290 PRINT "THE EQUATION";
300 INPUT N
310 IF N=1 THEN DEF FNA(X)=X*X*SIN(1/X)
320 IF N=2 THEN DEF FNA(X)=X*SIN(1/X)
330 IF N=3 THEN DEF FNA(X)=SQR(X*X+2)
340 IF N=4 THEN DEF FNA(X)=COS(X*EXP(-X/5))
350 IF N=5 THEN DEF FNA(X)=6+2*X*X-X*X*X*X
360 PRINT
370 PRINT "VALUES OF X FOR PLOT"
380 PRINT
390 PRINT "LOWEST VALUE";
400 INPUT A
410 PRINT
420 PRINT "HIGHEST VALUE";
430 INPUT B
440 PRINT
450 IF A)=B THEN PRINT "ERROR - TRY AGAIN"
460 IF A)=B TNEN GOTO 360
500 REM %%%%%%%%%%%%%%%%%% CALCULATING RANGE OF Y %%%%%%%%%%%%%%%%%%%%%%
510 PRINT "CALCULATING RANGE OF Y"
520 LET S=(B-A)/100
530 LET M=1.0E-30
540 FOR X=A TO B STEP C
550 IF X=0 THEN GOTO 580
560 LET Y=ABS(FNA(X))
570 IF M<Y THEN LET M=Y
580 NEXT X
```

SX и SY обозначают количество точек, которое можно изобразить на экране. Укажите соответствующие вашему экрану значения.

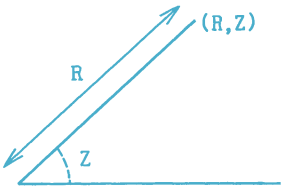
Вычисляется область значений Y (приблизительно). Затем выбирается такой масштаб, чтобы экран был заполнен целиком.

```
590 PRINT "READY FOR PLOTTING"
600 FOR I=1 TO 1000
610 NEXT I
620 PRINT CS%
630 GOSUB 1010 : REM PREPARE SCREEN
700 REM XXXXXXXXXXXXXXXXXXXX PLOTTING XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
710 LET C=C/10 : REM TRY C/5 OR C/20
720 FOR X=A TO B STEP C
730 IF X=0 THEN GOTO 790
740 LET Y=FNA(X)
750 LET U=SX*(X-A)/(B-A)
760 LET V=HY+HY*Y/M
770 IF V<0 OR V>SY THEN GOTO 790
780 GOSUB 1110 : REM PLOT U,V
790 NEXT X
800 REM XXXXXXXXXXXXXXXXXX ENDING AND ANOTHER GO XXXXXXXXXXXXXXXXXXXXXX
810 GET G$ : REM LET G$=INKEY$
820 IF G$="" THEN GOTO 810
830 GOSUB 1210 : REM RESTORE SCREEN
840 PRINT CS$
850 PRINT " ANOTHER GO? Y OR N"
860 GET G$ : REM LET G$=INKEY$
870 IF G$<>"Y" AND G$<>"N" THEN GOTO 860
880 IF G$="Y" THEN GOTO 150
890 END : REM STOP
1000 REM XXXXXXXXXXXXXXXX PREPARE HI-RES SCREEN XXXXXXXXXXXXXXXXXXXXX
1010 POKE 36869,R8+12:POKE 36867,(PEEK(36867) AND 128) OR 21
1020 FOR I=RR TO SS:POKE I,0:NEXT I
1030 FOR I=0 TO 219:POKE FF+I,I-32*Q:POKE QQ+I,2:NEXT I
1040 RETURN
1100 REM XXXXXXXXXXXXXXXXXXXX PLOT VIA POKE XXXXXXXXXXXXXXXXXXXXXXXXX
1110 V=SY-V
1120 J=INT(U/8)
1130 L=INT(U-8*J)
1140 I=INT V/16)
1150 K=INT(V-16*I)
1160 W=RR+I*352+J*16+K
1170 POKE W,PEEK(W) OR 2↑(7-L)
1180 RETURN
1200 REM XXXXXXXXXXXXXXXXXXXX RESTORE SCREEN XXXXXXXXXXXXXXXXXXXXXXX
1210 REM POKE 36869,R8:POKE 36867,R6:POKE 198,0
1220 RETURN
1300 REM XXXXXXXXXXXXXXXXXXXX VIC 20 SETTINGS XXXXXXXXXXXXXXXXXXXXXXX
1310 Q=PEEK(44)+18:PP=7680+Q*3584:QQ=38400+Q*512
1320 IFQ=-1ANDPEEK(44)<32THEN PRINT "PROGRAM ABORTED - SEE APPENDIX":END
1330 RR=4096-Q*512:SS=RR+3583:R8=PEEK(36869):R6=PEEK(36867)
1340 RETURN
```

Здесь происходит непосредственно построение (PLOT). Проверка делается для того, чтобы убедиться, что будут построены только необходимые точки.

Используйте соответствующую вашему компьютеру команду POKE.

Полярные координаты. Обычно точки на плоскости представляют их декартовыми координатами. Но есть и другой способ определения расположения точек на плоскости — задание *полярных координат*.

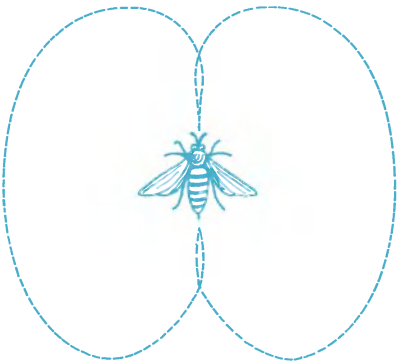


В этом случае имеется единственная ось и некая точка на ней, называемая полюсом. Любую точку на плоскости теперь можно определить парой чисел (R, Z), где R — расстояние от полюса и Z — угол между осью и прямой, соединяющей полюс и данную точку (угол изменяется в направлении против часовой стрелки от оси).

Удобно считать, что R и Z могут быть отрицательными. В этом случае углы и отрезки откладываются в направлениях, противоположных принятым. Например, следующие пары задают одну и ту же точку:

(2,PI/2), (2,−3*PI/2), (−2,−PI/2).

Полярные координаты пчелы. Медоносные пчелы используют полярные координаты для обмена информацией об источниках пищи. Найдя новый источник пищи (цветочную клумбу), пчела-разведчица возвращается в улей, приносит образчик найденной пищи и исполняет танец, на языке которого рассказывает, где находится клумба. Танец состоит в том, что пчела, покачиваясь с боку на бок, прочерчивает прямую и затем, описав плавную кривую, возвращается в начальную точку. Участок прямой повторяется, но на этот раз она возвращается по кривой в другом направлении. Весь процесс повторяется несколько раз. Длина отрезка прямой указывает расстояние до цветочной клумбы, а направление этой прямой — направление, в котором надо лететь. Таким образом пчела-разведчица сообщает другим пчелам полярные координаты нового источника пищи.



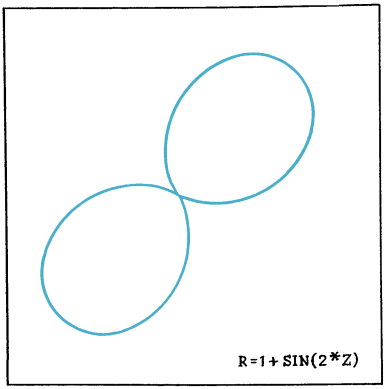
Графики в полярных координатах. Функции, в которых используются полярные координаты, будем называть *функциями в полярных*

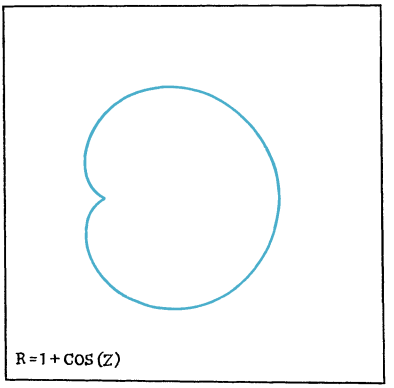
координатах. Например, R = SIN(Z) — функция в полярных координатах. Здесь для каждого значения Z из некоторой заданной области строится точка с полярными координатами (R, Z). Чтобы упростить построение, обратимся снова к декартовым координатам. Точка (R, Z) в полярных координатах — это то же самое что точка (R*COS(Z), R*SIN(Z)) в декартовых координатах, и именно ее мы строим. График функции R = SIN(Z) в полярных координатах совсем не похож на график функции Y = SIN(X) в декартовых координатах. Первый график — это окружность.

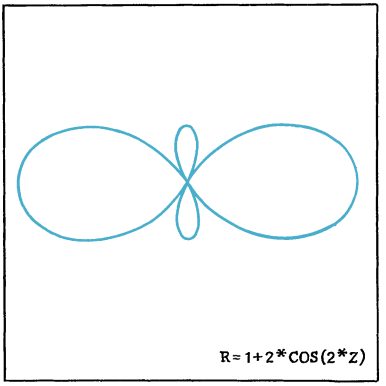
При помощи полярных координат очень просто представляются многие красивые кривые. Вот несколько примеров:

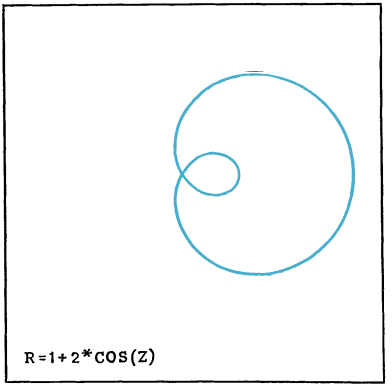
R = 1	окружность
R = SIN(2*Z)	четырехлепестковая роза
R = SIN(7*Z)	семилепестковая роза
R = 1 + 2*COS(Z)	улитка Паскаля
R = 1 + COS(Z)	кардиоида
R = Z/4	спираль
R = 1 + SIN(2*Z)	двухлепестковая роза
R = 1 + 2*COS(2*Z)	петельное сцепление

Некоторые из этих графиков в полярных координатах изображены на рисунках.









Программа ГРАФИКИ В ПОЛЯРНЫХ КООРДИНАТАХ представляющая собой некое видоизменение программы ПОСТРОЕНИЕ ГРАФИКОВ, позволяет строить графики функций в полярных координатах. Имеется библиотека из нескольких функций в полярных координатах, с которыми вы можете экспериментировать. Приводится также версия этой программы для микрокомпьютеров марки Sinclair.

```
10 REM                  **************************
20 REM                  *         ГРАФИКИ        *
30 REM                  * В ПОЛЯРНЫХ КООРДИНАТАХ *
40 REM                  *                        *
50 REM                  **************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET SX=144 : REM РАЗМЕР ЭКРАНА ПО ГОРИЗОНТАЛИ
130 LET SY=176 : REM РАЗМЕР ЭКРАНА ПО ВЕРТИКАЛИ
140 LET RATIO=0.6 : REM СДЕЛАТЬ СТРОКИ ПО ГОР И ВЕРТ ОДНОЙ ДЛИНЫ
150 LET HY=SY/2
160 LET HX=SX/2
165 GOSUB 1310 : REM ДОПОЛНИТЕЛЬНАЯ УСТАНОВКА ДЛЯ VIC 20
170 PRINT CS; : REM ОЧИСТИТЬ ЭКРАН
180 PRINT "ГРАФИКИ В ПОЛЯРНЫХ КООРДИНАТАХ"
190 PRINT
200 PRINT "1. R = 1"
210 PRINT
220 PRINT "2. R = SIN(2*Z)"
230 PRINT
240 PRINT "4. R = SIN(7*Z)"
250 PRINT
260 PRINT "4. R = 1+2*COS(Z)"
270 PRINT
280 PRINT "5. R = 1+COS(Z)"
```

SX и SY обозначают количество точек, которое можно изобразить на экране. Укажите значения, соответствующие вашему экрану. Величина RATIO введена для того, чтобы круг был похож на круг. Если значение 0.6 вас не устраивает, попробуйте другое число.

```
290 PRINT
300 PRINT "6. R = 1+SIN(2*Z)"
310 PRINT
320 PRINT "7. R = 1+2*COS(2*Z)"
330 PRINT
340 PRINT "УКАЖИТЕ НОМЕР"
350 PRINT "УРАВНЕНИЯ";
360 INPUT N
370 IF N=1 THEN DEF FNA(Z)=1
380 IF N=2 THEN DEF FNA(Z)=SIN(2*Z)
390 IF N=3 THEN DEF FNA(Z)=SIN(7*Z)
400 IF N=4 THEN DEF FNA(Z)=1+2*COS(Z)
410 IF N=5 THEN DEF FNA(Z)=1+COS(Z)
420 IF N=6 THEN DEF FNA(Z)=1+SIN(2*Z)
430 IF N=7 THEN DEF FNA(Z)=1+2*COS(2*Z)
440 PRINT
450 PRINT "ДЛЯ СТАНДАРТНОГО PLOT A=1 И B=1"
460 PRINT
470 PRINT "ЗНАЧЕНИЕ A"
480 INPUT A
490 PRINT
500 PRINT "ЗНАЧЕНИЕ B"
510 INPUT B
520 PRINT
600 REM XXXXXXXXXXXXXXXX ВЫЧИСЛЕНИЕ ОБЛАСТИ ЗНАЧЕНИЙ R XXXXXXXXXXXXXXXX
610 PRINT "ВЫЧИСЛЕНИЕ ОБЛАСТИ ЗНАЧЕНИЙ R"
620 LET M=1.0E-30
630 FOR Z=0 TO 2*π STEP 0.1 : REM  π=PI
640 LET R=ABS(FNA(Z))
650 IF M<R THEN LET M=R+0.1
660 NEXT Z
670 PRINT "ГОТОВ К ПОСТРОЕНИЮ ГРАФИКОВ"
680 FOR I=1 TO 1000
690 NEXT I
700 PRINT CS◦ : REM ОЧИСТИТЬ ЭКРАН
710 GOSUB 1010 : REM ПОДГОТОВИТЬ ЭКРАН,ЕСЛИ НУЖНО
800 REM XXXXXXXXXXXXXXXXXX ПОСТРОЕНИЕ ГРАФИКОВ XXXXXXXXXXXXXXXXXXXX
810 FOR Z=0 TO 2*π STEP 0.01 : REM  π= PI
820 LET R=FNA(Z)
830 LET U=HX+HY*RATIO*COS(A*Z)*R/M
840 IF U<0 OR U>SX THEN GOTO 880
850 LET V=HY+HY*SIN(B*Z)*R/M
860 IF V<0 OR V>SY THEN GOTO 880
870 GOSUB 1110 : REM PLOT U,V
880 NEXT Z
900 REM XXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXX
910 GET G◦ : REM LET G◦=INKEY◦920 IF G◦="" THEN GOTO 910
930 GOSUB 1210 REM : ВОССТАНОВИТЬ ЭКРАН, ЕСЛИ НУЖНО
940 PRINT CS◦ : REM ОЧИСТИТЬ ЭКРАН
950 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
960 GET G» : REM LET G◦=INKEY◦
970 IF G◦<>"Y" AND G◦<>"N" THEN GOTO 960
980 IF G◦="Y" THEN GOTO 170
990 END : REM STOP
1000 REM XXXXXXXXXXXXX ПОДГОТОВКА ЭКРАНА ДЛЯ VIC 20 XXXXXXXXXXXXXXXXXXX
1010 POKE 36869,R8+12:POKE 36867,(PEEK(36867) AND 128) OR 25
1020 POKE 36866,R5-4:POKE 36864,R3+4
1030 FOR I=RR TO SS:POKE I,0:NEXT I
1040 FOR I=0 TO 219:POKE PP+I,I-32*Q:POKE QQ+I,2:NEXT I
1050 RETURN
```

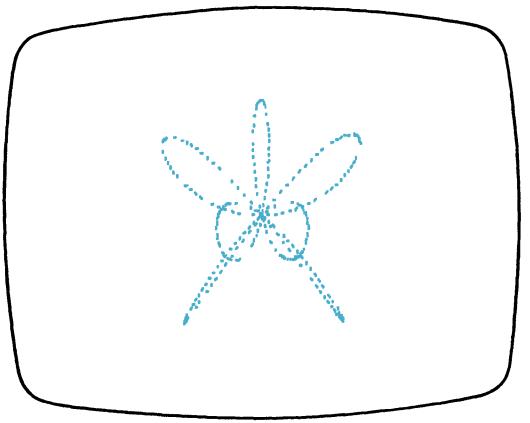
Здесь вычисляется область значений R (приблизительно). Затем выбирается такой масштаб, чтобы экран был заполнен целиком.

Здесь происходит непосредственно построение (PLOT). Проверка делается для того, чтобы убедиться, что будут построены только необходимые точки. Используйте либо PLOT, либо POKE.

```
1100 REM XXXXXXX ПОСТРОЕНИЕ ГРАФИКОВ ПОСРЕДСТВОМ POKE НА VIC 20 XXXXXXXXXX
1110 V=SY-V
1120 J=INT(U/8)
1130 L=INT(U-8*J)
1140 I=INT(V/16)
1150 K=INT(V-16*I)
1160 W=RR+J*288+J*16+K
1170 POKE W,PEEK(W) OR 2↑(7-L)
1180 RETURN
1200 REM XXXXXXXXXXXX ВОССТАНОВЛЕНИЕ ЭКРАНА ДЛЯ VIC 20 XXXXXXXXXXXXXXXXXX
1210 POKE 36869,R8:POKE 36867,R6
1220 POKE 36866,R5:POKE 36864,R3:POKE 198,0
1230 RETURN
1300 REM XXXXXXXXXXXXX УСТАНОВКА ДЛЯ VIC 20 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1310 Q=PEEK(44))=18:PP=7680+Q*3584:QQ=38400+Q*512
1320 IFQ=-1ANDPEEK(44)<32 THEN PRINT"ПРОГРАММА ПРЕРВАНА - СМ ПРИЛОЖЕНИЕ":END
1330 RR=4096-Q*512:SS=RR+3583:R8=PEEK(36869):R6=PEEK(36867)
1340 R5=PEEK(36866):R3=PEEK(36864)
1350 RETURN
```

Используйте соответствующую вашему компьютеру команду POKE.

```
ГРАФИКИ В ПОЛЯРНЫХ КООРДИНАТАХ

1. R = 1

2. R = SIN(2*Z)

3. R = SIN(7*Z)

4. R = 1+2*COS(Z)

5. R = 1+COS(Z)

6. R = 1+SIN(2*Z)

7. R = 1+2*COS(2*Z)

УКАЖИТЕ НОМЕР
УРАВНЕНИЯ?  3

ДЛЯ СТАНДАРТНОГО PLOT
A=1 И B=1

ЗНАЧЕНИЕ A?  1

ЗНАЧЕНИЕ B?  3
```



```
10 REM                    ******************
20 REM                    *    SINCLAIR    *
30 REM                    * POLAR GRAPHICS *
40 REM                    *                *
50 REM                    ******************
60 REM
70 REM
100 REM %%%%%%%%%%%%%%%%%%%% SETTING UP %%%%%%%%%%%%%%%%%%%%%%%%
120 LET SX=62
130 LET SY=42
140 LET RATIO=0.6
150 LET HY=SY/2
160 LET HX=SX/2
170 CLS
180 PRINT "   POLAR GRAPHICS"
190 PRINT
200 PRINT "1. R = 1"
210 PRINT
220 PRINT "2. R = SIN(2*Z)"
230 PRINT
240 PRINT "3. R = SIN(7*Z)"
250 PRINT
260 PRINT "4. R = 1+2*COS(Z)"
270 PRINT
280 PRINT "5. R = 1+COS(Z)"
290 PRINT
300 PRINT "6. R = 1+SIN(2*Z)"
310 PRINT
320 PRINT "7. R = 1+2*COS(2*Z)"
330 PRINT
340 PRINT "TYPE IN THE NUMBER OF"
350 PRINT "THE EQUATION ";
360 INPUT N
365 PRINT N
370 IF N=1 THEN LET A$="1"
380 IF N=2 THEN LET A$="SIN(2*Z)"
390 IF N=3 THEN LET A$="SIN(7*Z)"
```

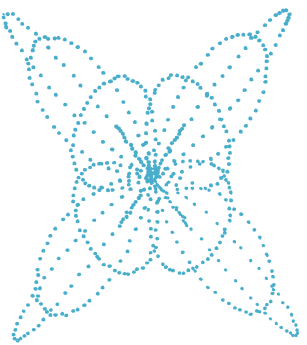
SX и SY обозначают количество точек, которое можно изобразить на экране. Укажите соответствующие вашему экрану значения. Величина RATIO введена для того, чтобы круг был похож на круг. Если значение 0.6 вас не устраивает, попробуйте другое число. Для ZX Spectrum подойдет 0.92.

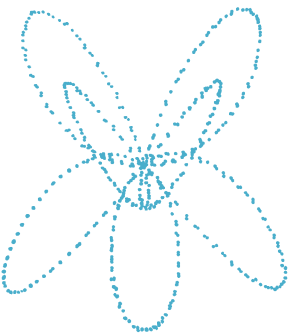
```
400 IF N=4 THEN LET A$="1+2*COS(Z)"
410 IF N=5 THEN LET A$="1+COS(Z)"
420 IF N=6 THEN LET A$="1+SIN(2*Z)"
430 IF N=7 THEN LET A$="1+2*COS(2*Z)"
440 PRINT
450 PRINT "FOR STANDARD PLOT USE A=1 AND B=1"
460 PRINT
470 PRINT "VALUE OF A ";
480 INPUT A
485 PRINT A
490 PRINT
500 PRINT "VALUE OF B ";
510 INPUT B
515 PRINT B
520 PRINT
600 REM %%%%%%%%%%%%%%%% CALCULATING RANGE OF R %%%%%%%%%%%%%%%%%%%%
610 PRINT "CALCULATING RANGE OF R"
620 LET M=1.0E-30
630 FOR Z=0 TO 2*PI STEP 0.1
640 LET R=ABS(VAL(A$))
650 IF M<R THEN LET M=R+0.1
660 NEXT Z
670 PRINT "READY FOR PLOTTING"
680 FOR I=1 TO 100
690 NEXT I
700 CLS
800 REM %%%%%%%%%%%%%%%%%%%% PLOTTING %%%%%%%%%%%%%%%%%%%%%%%%%%
810 FOR Z=0 TO 2*PI STEP 0.01
820 LET R=VAL(A$)
830 LET U=HX+HY*RATIO*COS(A*Z)*R/M
840 IF U<0 OR U>SX THEN GOTO 880
850 LET V=HY+HY*SIN(B*Z)*R/M
860 IF V<0 OR V>SY THEN GOTO 880
870 PLOT U,V
880 NEXT Z
900 REM %%%%%%%%%%%%%%%%%% ENDING AND ANOTHER GO %%%%%%%%%%%%%%%%
910 LET G$=INKEY$
920 IF G$="" THEN GOTO 910
950 PRINT " ANOTHER GO? Y OR N"
960 LET G$=INKEY$
970 IF G$<>"Y" AND G$<>"N" THEN GOTO 960
980 IF G$="Y" THEN GOTO 170
990 STOP
```

Вычисляется область значений R (приблизительно). Затем выбирается такой масштаб, чтобы экран был заполнен целиком.

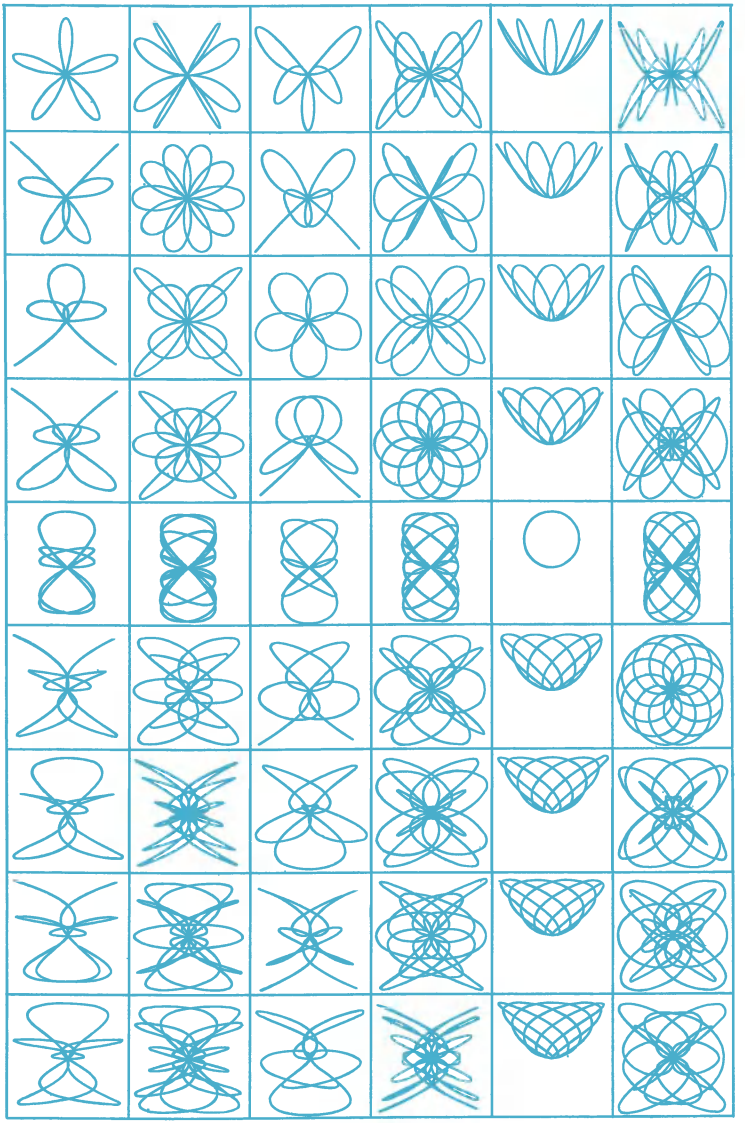
Здесь происходит непосредственно построение (PLOT). Проверка делается для того, чтобы убедиться, что будут построены только необходимые точки.

Дополнительные замечания см. в приложении.



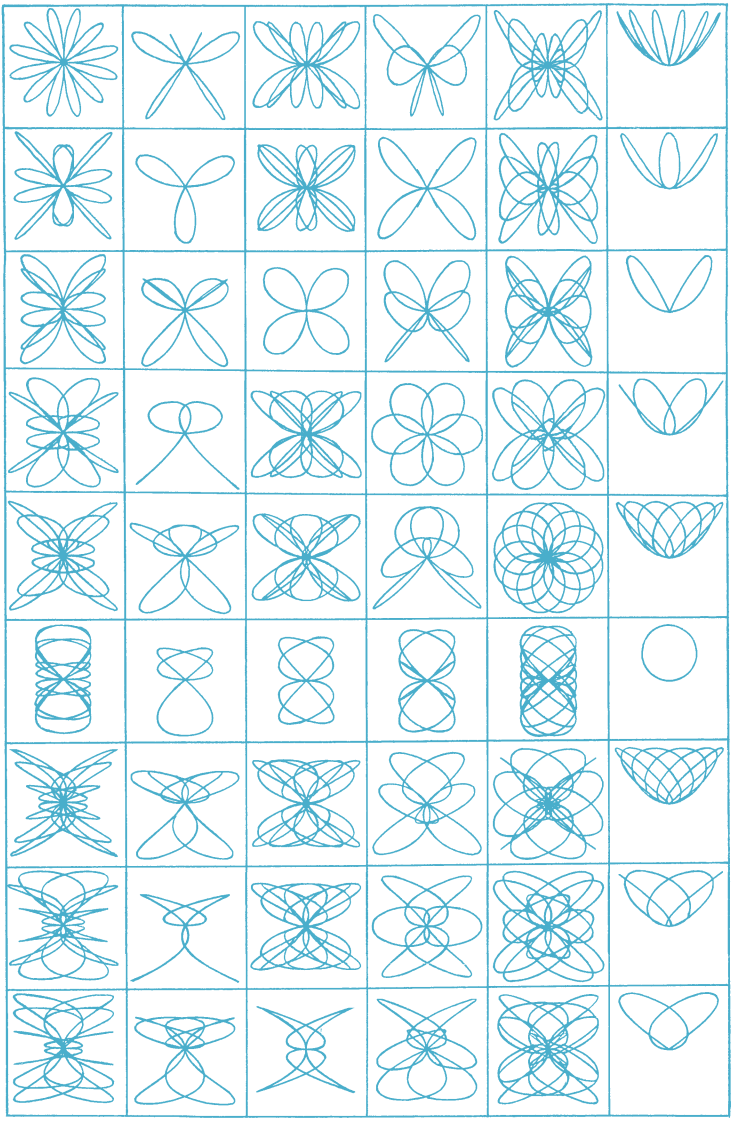


Проявив некоторую предприимчивость, вы сможете создать массу поистине замечательных картинок. Вместо использования при построении декартовых координат (R*COS(Z),R*SIN(Z)), введите два дополнительных параметра A и B и стройте (R*COS(A*Z),R*SIN(B*

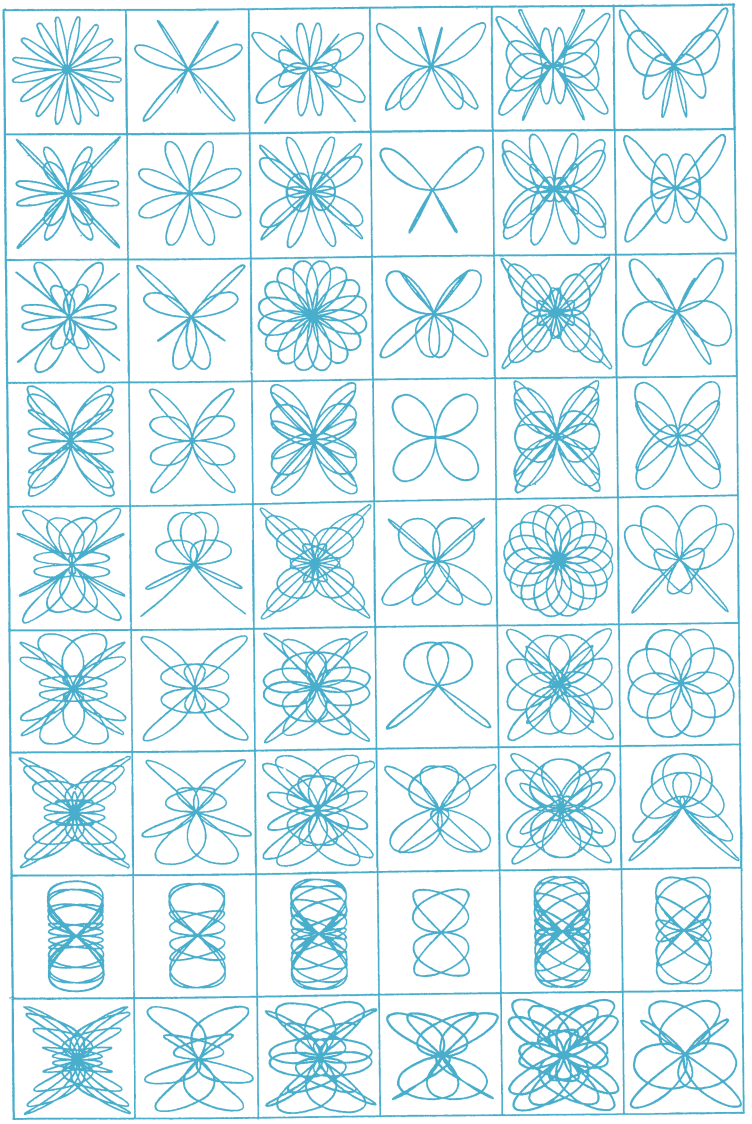


R=SIN(5*Z)

*Z)). Меняя значения A и B, вы получите великолепное зрелище, причем разнообразие этих прелестных картинок чрезвычайно велико. Здесь представлены некоторые из этих картинок: каждая страница соответствует одной формуле со значениями A от 1 до 9 и со значениями B от 1 до 6.



R = SIN(6*Z)

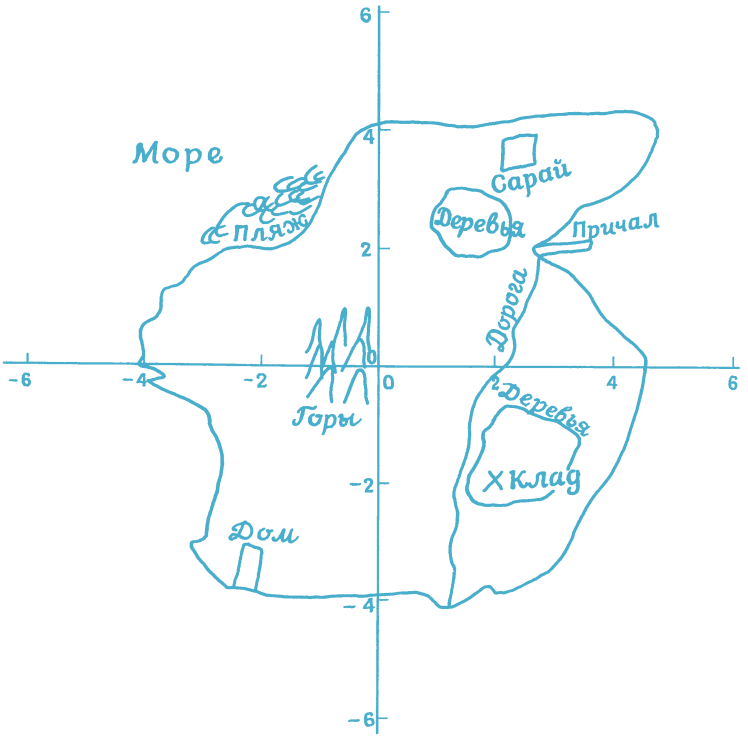


R = SIN(8*Z)

5 км на Север, 4 км на Восток

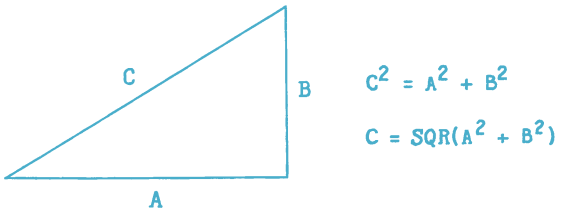
О геометрии

Декартовы координаты указывают нам местоположение точки на плоскости. Ниже вы видите карту Острова сокровищ, нарисованную одной из моих дочерей. Используя систему координат, можно указать разные места на карте. Например, место, где зарыт клад, имеет координаты (2, −2), тогда как причал — (3, 2).



На каком расстоянии от причала находятся сокровища? Мы можем пройти 1 км на Запад и 4 км на Юг, чтобы добраться от причала до места, где зарыт клад. В общей сложности это 5 км. Разумеется, путь по прямой был бы короче — горы и другие возможные препятствия во внимание не принимаются. Это расстояние (напрямик) есть SQR(4* *4 + 1*1), т. е. около 4.1 км. Оно подсчитано при помощи теоремы Пифагора. (Пифагор был древнегреческим математиком и философом, родившимся примерно в 582 г. до н. э.)

Прямоугольным называется треугольник, имеющий прямой угол, т. е. угол PI/2 радиан, или 90 градусов. Теорема Пифагора устанавливает, что квадрат самой длинной стороны (гипотенузы прямоугольного треугольника) равен сумме квадратов двух других сторон (катетов). Обозначив длины сторон треугольника буквами A, B и C, где C — длина гипотенузы, имеем $C^2 = A^2 + B^2$.



Используем теорему Пифагора для вычисления расстояния (по прямой) между любыми двумя точками. Пусть P — точка с координатами (XP, YP) и Q — точка с координатами (XQ, YQ). Расстояние между P и Q задается выражением

$$SQR((XQ - XP)^2 + (YQ - YP)^2).$$

Иначе говоря, расстояние между P и Q равно квадратному корню (SQR) из суммы квадрата разности X-координат и квадрата разности Y-координат.

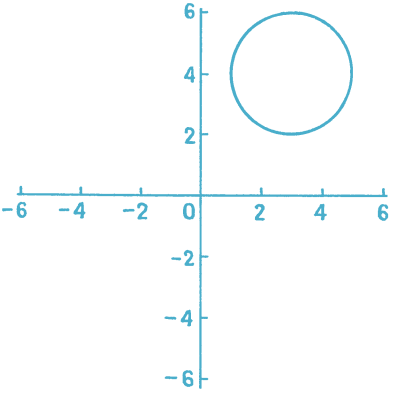
Представьте себе, что вы находитесь в точке с координатами (3,4), а ваш друг — на расстоянии 2 (скажем, метра) от вас. Сможете вы определить координаты вашего друга? Ответ — «нет», если только вы не получите дополнительную информацию. На самом деле точки, находящиеся на расстоянии 2 единицы от вас, образуют окружность радиуса 2. Чтобы убедиться в этом, будем считать, что ваш друг находится в точке (X, Y). Вам известно лишь, что он находится на расстоянии 2 единицы от вас. Используя формулу расстояния, получаем следу-

ющее уравнение:

$$SQR((X - 3)^2 + (Y - 4)^2) = 2.$$

Его можно переписать по-другому:

$$(X - 3)^2 + (Y - 4)^2 = 4.$$



Если вы построите все точки вида (X, Y), удовлетворяющие этому уравнению, то получите окружность радиуса 2. При каждом значении X можно определить, какое значение Y (если таковое имеется) удовлетворяет этому уравнению. Заметим, что X должно находиться между 1 и 5, иначе точка (X, Y) будет находиться дальше чем на 2 единицы от (3,4). Если, например, мы возьмем X = 3, то наше уравнение превратится в

$$(3 - 3)^2 + (Y - 4)^2 = 4,$$

или, проще, в $(Y - 4)^2 = 4$, из чего заключаем, что $(Y - 4) = SQR(4)$ или $(Y - 4) = -SQR(4)$. Иными словами, Y = 6 или Y = 2. Таким образом, две точки (3,5) и (3,2) лежат на нашей окружности. Выбирая различные значения X, можно получить другие точки окружности.

В более общем виде уравнение окружности радиуса R с центром в точке (XP, YP) задается уравнением

$$(X - XP)^2 + (Y - YP)^2 = R^2.$$

В компьютерной игре В ПОИСКАХ СОКРОВИЩ на вашем экране четко обозначаются ряды кустарника. Клад — под одним из них. Ваша задача — выяснить, где именно зарыт клад. Вы находитесь у мигающего куста и можете передвигаться от одного куста к другому, нажи-

мая клавиши U (вверх), D (вниз), R (вправо) или L (влево). Нажав клавишу *, вы узнаете приблизительное расстояние от того места, где вы находитесь, до места, где зарыт клад. Точнее, если D — расстояние между вами и кладом, то компьютер выведет на экран INT(D) — целую часть числа D, т. е. его значение с отброшенными знаками после десятичной точки. Таким образом, если на экране высветилось число D, то вы понимаете, что клад находится на расстоянии, большем или равном D, но меньшем, чем D + 1.

Вот еще две особенности программы В ПОИСКАХ СОКРОВИЩ. Во-первых, если расстояние между вами и кладом превышает или равно 10, то вместо числа на экран выводится буква X. Во-вторых, только в половине случаев на экран выводится расстояние. В остальное время появляются вопросительные знаки.

Призвав на помощь интуицию и геометрическое воображение или довольствуясь простыми догадками, постарайтесь определить место, где зарыты сокровища, за минимальное число перемещений. Если вы умны и удачливы, то сможете обойтись тремя — четырьмя попытками.

```
10 REM                    **********************
20 REM                    *                    *
30 REM                    * В ПОИСКАХ СОКРОВИЩ *
40 REM                    *                    *
50 REM                    **********************
60 REM
70 REM
100 REM %%%%%%%%%%%%%%%%%% НАЧАЛЬНАЯ УСТАНОВКА %%%%%%%%%%%%%%%%%%%%%%%%%
110 LET CS$=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET HM$=CHR$(19) : REM КОД ИСХОДНОГО ПОЛОЖЕНИЯ
130 LET S=ASC("0") : REM S=CODE("0")
140 LET P=10 : REM ЧИСЛО КУСТОВ ПО ВЕРТ
150 LET Q=9 : REM ЧИСЛО КУСТОВ ПО ГОРИЗ
160 DIM A$(P,Q)
170 FOR I=1 TO P
180 FOR J=1 TO Q
190 LET A$(I,J)="*" : REM LET A$(I,J)="▒"
200 NEXT J
210 NEXT I
220 LET M=5
230 LET N=5
240 LET MM=INT(RND(1)*P+1)
250 LET NN=INT(RND(1)*Q+1)
260 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
270 GOSUB 710
300 REM %%%%%%%%%%%%%%%%%%%%%% ЦИКЛ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
310 LET B$=A$(M,N)
320 LET A$(M,N)=" "
330 GOSUB 910
340 LET A$(M,N)=B$
350 GOSUB 910
400 REM %%%%%%%%%%%%%%%%%% ВАШИ ПЕРЕМЕЩЕНИЯ %%%%%%%%%%%%%%%%%%%%%%%%%%%
410 GET G$ : REM LET G$=INKEY$
420 IF G$="D" AND M<P THEN LET M=M+1
430 IF G$="U" AND M>1 THEN LET M=M-1
```

Количество кустов равно P*Q. В этой программе P = 10, Q = 9. Если хотите, можете взять другие значения. Соответствующая информация храниться в массиве A$(I, J).

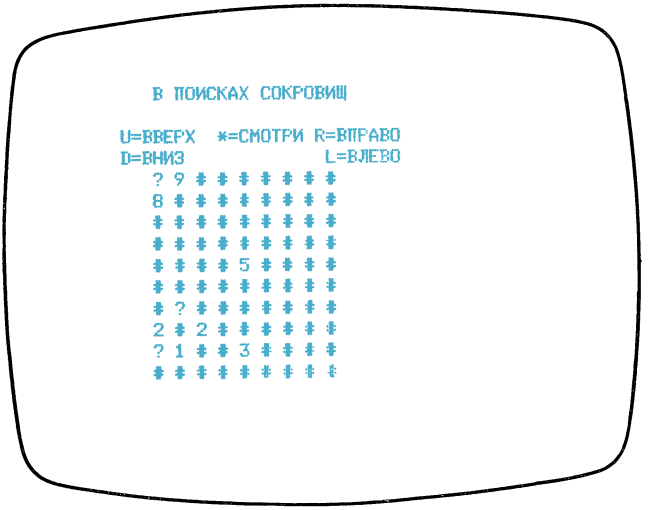
Клад зарыт под кустом A$(MM, NN).

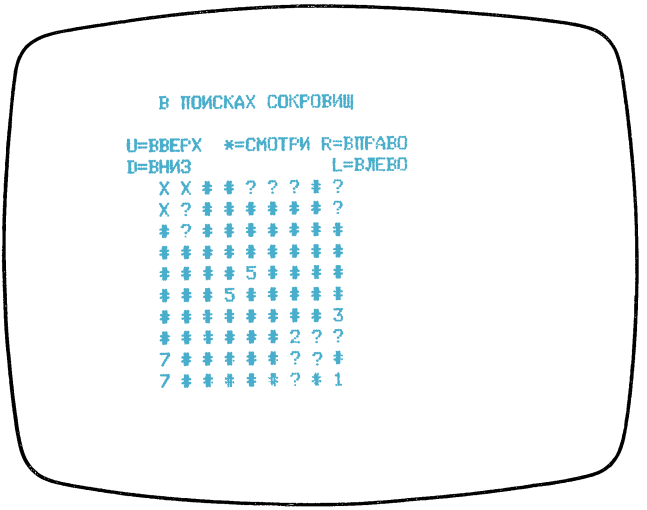
```
440 IF G$="R" AND N<Q THEN LET N=N+1
450 IF G$="L" AND N>1 THEN LET N=N-1
460 IF G$="*" AND A$(M,N)="$" THEN GOTO 510 : REM "$"
470 GOTO 310
500 REM XXXXXXXXXX ВЫВОД РАССТОЯНИЯ И ПРОВЕРКА НА ОКОНЧАНИЕ XXXXXXXXXXXXXXX
510 LET DIST=INT(SQR((MM-M)*(MM-M)+(NN-N)*(NN-N)))
520 LET A$(M,N)=CHR$(S+DIST)
530 IF DIST=0 THEN GOTO 610
540 IF DIST>9 THEN LET A$(M,N)="X"
550 IF RND(1)<0.5 THEN LET A$(M,N)="?"
560 GOTO 310
600 REM XXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXXXXXXX
610 GOSUB 710
620 GOSUB 1010 : REM ЗВУКОВЫЕ ЭФФЕКТЫ
630 PRINT " ПОВТОРИТЬ? Y ИЛИ N";
640 GET G$ : REM LET G$=INKEY$
650 IF G$<>"Y" AND G$<>"N" THEN GOTO 640
660 IF G$="Y" THEN GOTO 170
670 END : REM STOP
700 REM XXXXXXXXXXXXXXXXX ВЫВОД НА ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
710 PRINT HM$; : REM PRINT AT 0,0
720 PRINT " U=ВВЕРХ *=СМОТРИ R=ВПРАВО"
730 PRINT " D=ВНИЗ            L=ВЛЕВО"
740 FOR I=1 TO P
750 PRINT " ";
760 FOR J=1 TO Q
770 PRINT " ";A$(I,J);
780 NEXT J
790 PRINT
800 PRINT
810 NEXT I
820 RETURN
900 REM XXXXXXXXXXXXXXXXX  МИГАЮЩИЙ КУРСОР XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
905 REM ЕСЛИ ВОЗМОЖНО, ИСПОЛЬЗУЙТЕ PRINT AT 2*M,2*N;A$(M,N)
910 PRINT HM$;
920 FOR I=1 TO 2*M
930 PRINT
940 NEXT I
950 PRINT TAB(2*N) A$(M,N)
960 RETURN
1000 REM XXXXXXXXXXXXXXXX ЗВУКОВЫЕ ЭФФЕКТЫ XXXXXXXXXXXXXXXXXXXXXXXXXXXX
1010 POKE 36878,15 : K=1
1020 T=250-K*20 : K=K+1
1030 POKE 36876,1
1040 FOR I=1 TO 200 : NEXT
1050 POKE 36876,0
1060 IF K<4 THEN 1020
1070 POKE 36878,0
1080 RETURN
```

Вместо строк 910—950 можно воспользоваться командой PRINT AT, если она имеется.

Дайте свои звуковые эффекты.

Дополнительные замечания см. в приложении.

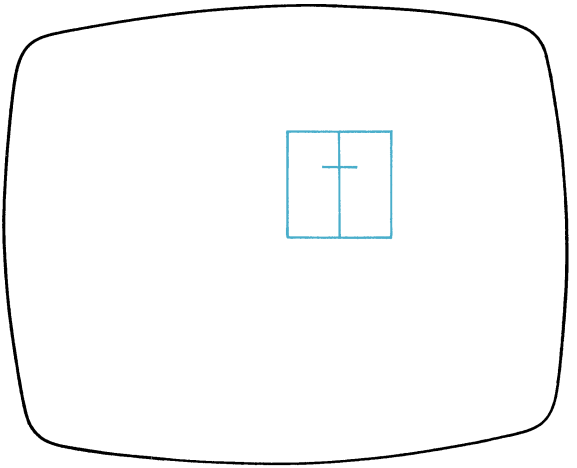

В ПОИСКАХ СОКРОВИЩ
U=ВВЕРХ *=СМОТРИ R=ВПРАВО
D=ВНИЗ L=ВЛЕВО
? 9 # # # # # # #
8 # # # # # # # #
# # # # # # # # #
# # # # # # # # #
5 # # #
# # # # # # # # #
? # # # # # #
2 # 2 # # # # # #
? 1 # # 3 # # # #
# # # # # # # # #


В ПОИСКАХ СОКРОВИЩ
U=ВВЕРХ *=СМОТРИ R=ВПРАВО
D=ВНИЗ L=ВЛЕВО
X X # # ? ? ? # ?
X ? # # # # # # ?
? # # # # # #
# # # # # # # # #
5 # # #
5 # # # #
3
2 ? ?
7 # # # # # ? ? #
7 # # # # # ? # 1

О матрицах

Рисовать картинки при помощи компьютера — неплохое развлечение. Если вам вдруг захочется изменить получившиеся картинки, вытягивая и сжимая их в разных направлениях, это легко осуществить при помощи *матриц*.

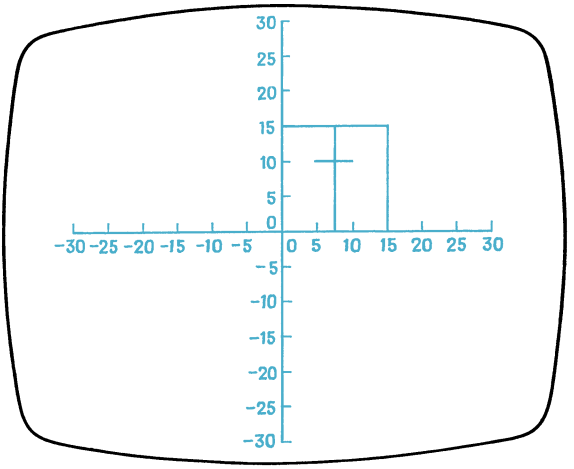
Предположим, что вам нужна вот такая картинка:



Получить ее на экране можно при помощи команд PLOT или POKE, примененных к подходящим точкам. Это сделает за вас программа ИССЛЕДОВАНИЕ МАТРИЦ, с которой вы познакомитесь позже в этой главе.

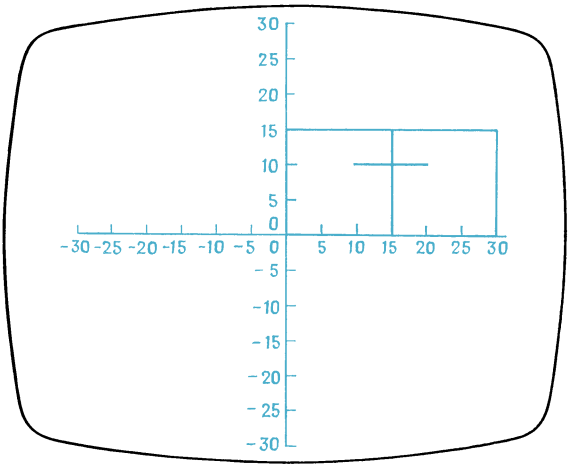
Программа поможет вам снабдить картинку системой координат:

```
1 Ø
Ø 1
```



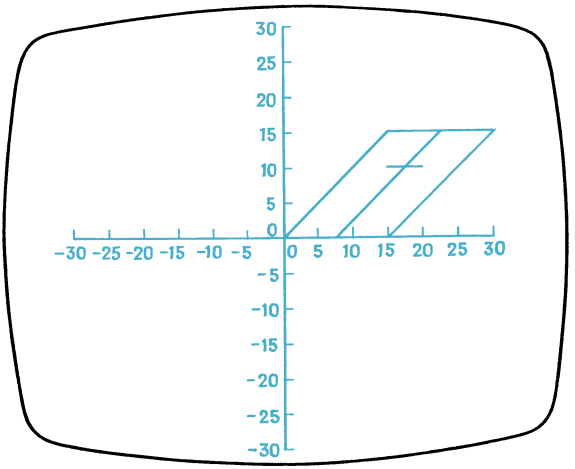
Что следует делать, если вы хотите вытянуть картинку по горизонтали так, чтобы она стала вдвое шире? Этого можно добиться простым удвоением значения координаты X, откладываемой по горизонтали. Иначе говоря, если первоначальным значением координат некой точки было (X,Y), то нужно построить точку с координатами (2*X,Y). Например, точка с координатами (15,15) теперь станет точкой (30,15).

```
2 Ø
Ø 1
```



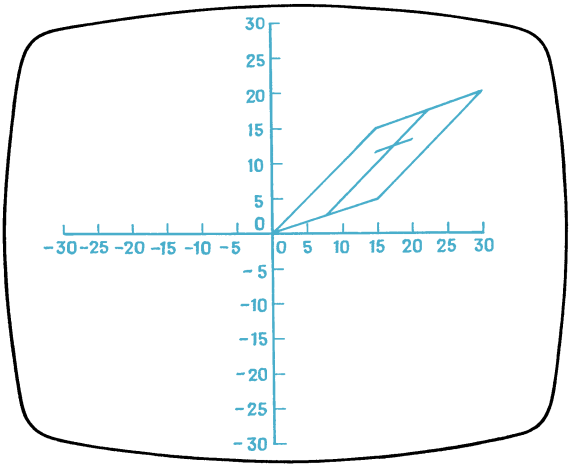
Вы можете поставить перед собой и более смелую задачу. Например, наклонить картинку вправо. Это достигается построением (X + + Y,Y) вместо (X,Y). Например, точка с координатами (0,15) на первоначальной картинке превратится в точку (15,15):

```
1  1
ø  1
```



Мы можем проявить даже еще большую изобретательность: заменить точки с координатами (X,Y) на (X + Y,Y + X/3). Например, вместо точки (15,15) исходной картинки теперь построим точку (30,20). То, что получилось, изображено ниже.

```
1   1
1/3 1
```



В общем случае это записывается так: мы можем строить точки (M1*X + M2*Y,M3*X + M4*Y) вместо точки (X,Y), где M1, M2, M3, M4 — произвольные числа. Разные наборы их значений дают разные видоизменения картинки. Числа M1, M2, M3 и M4 обычно хранятся в виде квадратного массива:

M1 M2
M3 M4

Матрицей называется такой квадратный массив чисел M(I,J), в котором

M(1,1) = M1, M(1,2) = M2,
M(2,1) = M3, M(2,2) = M4.

Матрица, соответствующая каждой картинке, помещается слева от нее.

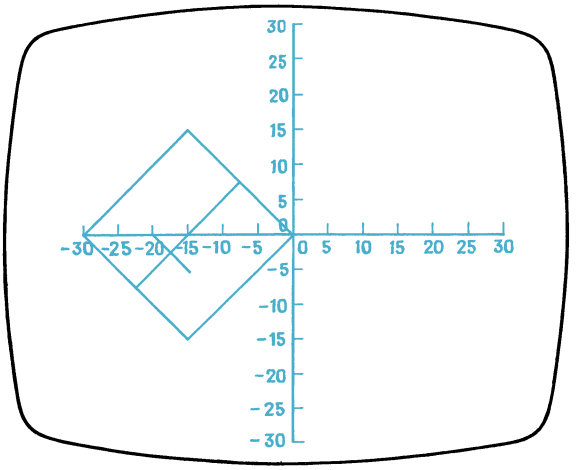
Заметим, что матрица

1 0
0 1

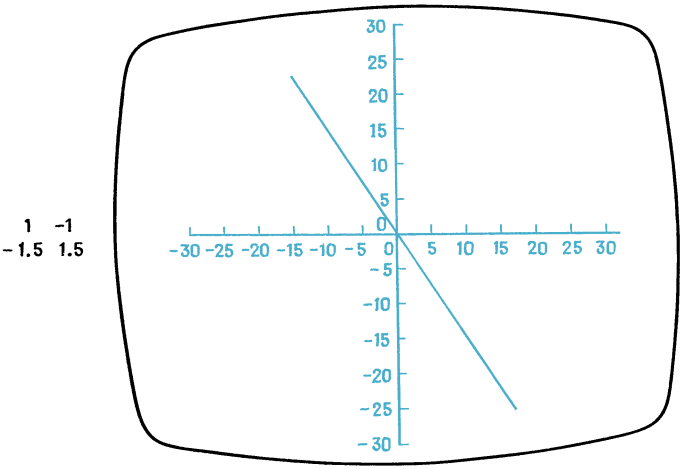
никак не меняет картинку; она называется матрицей *тождественного преобразования*, или *единичной* матрицей.

Ниже приводится еще один пример, демонстрирующий воздействие матрицы на нашу картинку.

-1 -1
1 -1



Каждую матрицу характеризует одно важное число, называемое *детерминантом* (*определителем*), которое вычисляется по формуле M1 * M4 − M2 * M3. По его значению можно судить, что произойдет с картинкой: растяжение или сжатие. Если значение определителя находится между −1 и +1, происходит сжатие, в противном случае — растяжение. Когда определитель равен нулю, наблюдается нечто странное: картинка сплющивается в прямую и становится совершенно неузнаваемой. Такой результат дает матрица, выписанная ниже; рядом — то, что она натворила.



При помощи программы ИССЛЕДОВАНИЕ МАТРИЦ вы можете изучать воздействие матриц на картинку. При выполнении программы вам нужно будет вводить (INPUT) матрицу, значение за значением. Затем на экран будет выведен определитель. После этого, прежде чем появится новое изображение, вам будет предоставлена возможность изменить матрицу.

```
10 REM                    ***********************
20 REM                    *                     *
30 REM                    * ИССЛЕДОВАНИЕ МАТРИЦ *
40 REM                    *                     *
50 REM                    ***********************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНАЕ
120 LET SX=176 : REM РАЗМЕР ЭКРАНА ПО ГОРИЗ
130 LET SY=160 : REM РАЗМЕР ЭКРАНА ПО ВЕРТ
140 LET RATIO =0.6 : REM СДЕЛАТЬ СТРОКИ ПО ВЕРТ И ГОРИЗ ОДНОЙ ДЛИНЫ
150 LET HX=SX/2
160 LET HY=SY/2
```

```
170 REM XXXXXXXX ВВОД МАТРИЦЫ XXXXXXX
180 PRINT CS¥ : REM ОЧИСТИТЬ ЭКРАН
190 PRINT " ИССЛЕДОВАНИЕ МАТРИЦ"
200 PRINT
210 PRINT "ВВОДИТЕ СВОЮ МАТРИЦУ"
220 PRINT
230 PRINT "СТРОКА 1 СТОЛБЕЦ 1 ";
240 INPUT M1
250 PRINT "СТРОКА 1 СТОЛБЕЦ 2 ";
260 INPUT M2
270 PRINT
280 PRINT "СТРОКА 2 СТОЛБЕЦ 1 ";
290 INPUT M3
300 PRINT "СТРОКА 2 СТОЛБЕЦ 2 ";
310 INPUT M4
320 PRINT
330 PRINT "ВАША МАТРИЦА:"
340 PRINT TAB(4);M1;TAB(10);M2
350 PRINT TAB(4);M3;TAB(10);M4
360 PRINT
370 LET DET=M1*M4-M2*M3
380 PRINT "ОПРЕДЕЛИТЕЛЬ:";DET
390 PRINT
400 PRINT "ВЫ ХОТИТЕ"
410 PRINT "ПРОДОЛЖАТЬ^ Y ИЛИ N"
420 PRINT
430 GET G¥ : REM LET G>=INKEY◦
440 IF G¤<()"Y" AND G¤<()"N" THEN GOTO 430
450 IF G¤="N" THEN GOTO 180
460 PRINT " ЛАДНО, НО НЕМНОГО"
470 PRINT " ТЕРПЕНИЯ, ПОЖАЛУЙСТА "
480 FOR I=1 TO 1000
490 NEXT I
500 LET DD=ABS(DET)
510 IF DD<1 THEN LET DD=1
520 PRINT CS¥ : REM ОЧИСТИТ> ЭКРАН
530 GOSUB 1010 : REM ПОДГОТОВИТЬ ЭКРАН, ЕСЛИ НУЖНО
600 REM XXXXXXXXXXXXXXXXXX КАРТИНКА XXXXXXXXXXXXXXXXXXXXXXXXX
610 FOR X=0 TO 15 STEP 7.5
620 FOR Y=0 TO 15 STEP 1/DD
630 GOSUB 910
640 NEXT Y
650 NEXT X
660 FOR Y=0 TO 15 STEP 15
670 FOR X=0 TO 15 STEP 1/DD
680 GOSUB 910
690 NEXT X
700 NEXT Y
710 LET Y=12
720 FOR X=5 TO 10 STEP 1/DD
730 GOSUB 910
740 NEXT X
800 REM XXXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXX
810 GOSUB 1210 : REM ВОССТАНОВИТЬ ЭКРАН, ЕСЛИ НУЖНО
820 PRINT CS¤ : REM ОЧИСТИТЬ ЭКРАН
830 PRINT " ПОВТОРИТЬ?  Y ИЛИ N"
840 GET G¥ : REM LET G>=INKEY¤
850 IF G¤<()"Y" AND G¥<()"N" THEN GOTO 840
860 IF G¤="Y" THEN GOTO 180
870 END : REM STOP
```

Используйте подходящие для вашего компьютера значения SX, SY и RATIO.

В этой части вводится матрица.

Матрица: M1 M2
M3 M4

Дополнительные замечания см. в приложении. См. также замечания к программе ПОСТРОЕНИЕ ГРАФИКОВ.

Если предпочитаете свою картинку, пожалуйста.

```
900 REM XXXXXXXXXXXXXXXX ВЫЧЕРЧИВАНИЕ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
910 LET U=(M1*X+M2*Y)*RATIO+HX
920 IF U<0 OR U>SX THEN RETURN
930 LET V=HY+M3*X+M4*Y
940 IF V<0 OR V>SY THEN RETURN
950 GOSUB 1110 : REM PLOT U,V
960 RETURN
1000 REM XXXXXXXXXXXXX ПОДГОТОВКА ЭКРАНА ДЛЯ VIC 20 XXXXXXXXXXXXXXXXXX
1010 POKE 36869,R8+12:POKE 36867,(PEEK(36867) AND 128) OR 21
1020 FOR I=RR TO SS:POKE I,0:NEXT I
1030 FOR I=0 TO 219:POKE PP+I,J-32*Q:POKE QQ+I 2:NEXT I
1040 RETURN
1100 REM XXXXXXXXXXXXX ИСПОЛЬЗОВАНИЕ POKE НА VIC 20 XXXXXXXXXXXXXXXXX
1110 V=SY-V
1120 J=INT(U/8)
1130 L=INT(U-8*J)
1140 I=INT(V/16)
1150 K=INT(V-16*I)
1160 W=RR+I*352+J*16+K
1170 POKE W,PEEK(W) OR 2↑(7-L)
1180 RETURN
1200 REM XXXXXXXXXXXXX ВОССТАНОВЛЕНИЕ ЭКРАНА ДЛЯ VIC 20 XXXXXXXXXXXXXXXXXX
1210 POKE 198,0
1220 GET G$:IF G$="" THEN 1220
1230 POKE 36869,R8:POKE 36867,R6:POKE 198,0
1240 RETURN
1300 REM XXXXXXXXXXXXX УСТАНОВКА ДЛЯ VIC 20 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1310 Q=PEEK(44)>18:PP=7680+Q*3584:QQ=38400+Q*512
1320 IFQ=-1ANDPEEK(44)<32THEN PRINT "ПРОГРАММА ПРЕРВАНА - СМ ПРИЛОЖЕНИЕ":END
1330 RR=4096-Q*512:SS=RR+3583:R8=PEEK(36869):R6=PEEK(36867)
1340 RETURN
```

Обратите внимание на M1*X + M2*Y и M3*X + M4*Y.

Используйте подходящую команду POKE, если PLOT на вашей машине отсутствует.

```
          ИССЛЕДОВАНИЕ МАТРИЦ

ВВОДИТЕ СВОЮ МАТРИЦУ

СТРОКА 1 СТОЛБЕЦ 1 ? 1
СТРОКА 1 СТОЛБЕЦ 2 ? 1

СТРОКА 2 СТОЛБЕЦ 1 ? .333
СТРОКА 2 СТОЛБЕЦ 2 ? 1

ВАША МАТРИЦА:
       1      1
       .333   1

ОПРЕДЕЛИТЕЛЬ: .667

ВЫ ХОТИТЕ
ПРОДОЛЖАТЬ? Y ИЛИ N

ЛАДНО, НО НЕМНОГО
ТЕРПЕНИЯ, ПОЖАЛУЙСТА
```

Матрицы играют в математике очень важную роль. Их применение не ограничивается растяжением и сжатием изображений. Они, например, используются при решении систем линейных уравнений.

Матрицы, с которыми мы сейчас встретились, — это всё матрицы размера 2×2: они имели по две строки и по два столбца. Но, вообще говоря, матрицы могут быть любого размера и не обязательно квадратной формы. Любой массив чисел, представленный в виде прямоугольной таблицы, называется *матрицей*. Ниже приводится пример матрицы размера 4×6, состоящей из четырех строк и шести столбцов:

$$\begin{array}{cccccc} 1 & 3 & -2 & 1 & 9 & -1 \\ 0 & 2 & 1 & -6 & 7 & 0 \\ 9 & 2 & 0 & 0.5 & 4 & 1.2 \\ 7 & 6.3 & 0 & 5 & 7.9 & 9 \end{array}$$

Матрицы — полезный прием компактного хранения информации. Другие области применения матриц описаны в следующей главе.

О теории игр

Все игры, вообще говоря, можно разбить на три категории: состязание в «везении», состязание в мастерстве и состязание в стратегии. К состязанию в «везении» относятся азартные игры: рулетка, очко и т. п. Для подобных игр большого мастерства не требуется, достаточно благосклонности фортуны. Хотя понимание законов теории вероятностей поможет вам разобраться, почему вы проиграли. К состязанию в мастерстве относятся такие игры, как гольф, футбол, хоккей и т. п. Здесь с самого начала ясно, что именно следует делать; задача в том, чтобы сделать это как можно лучше. Сюда можно отнести и головоломки наподобие кубика Рубика; в их решении вам поможет математика (см. следующую главу). Третья разновидность игр — состязание в стратегии: это крестики-нолики, шахматы, шашки, монополия, камень-ножницы-бумага и т. д. Здесь вам и вашему сопернику (соперникам) предоставляется возможность выбора из нескольких вариантов: нужно принимать решение. В шахматах, например, состязанию в стратегии отводится ведущая роль. Настоящая глава как раз и посвящена *стратегическим играм*.

Многие стратегические игры возникли как модель той или иной стороны реальной жизни, помогающая получить необходимый навык. Например, шахматы имеют в качестве источника военные действия, монополия — капиталистические предприятия. Как мы подчас наблюдаем, если два или более лиц берутся принимать решение, возникает конфликтная ситуация или простое соперничество. Такое положение характерно для экономической, политической, социальной и военной сфер. Для урегулирования подобных конфликтов несомненную пользу принесет раздел математики под названием *теория игр*. В этой науке предпринимается попытка снабдить соперников теорией, при помощи которой они смогут оптимизировать (максимизировать) свой выигрыш. Теория игр помогает изучать ситуации, взятые из реальной жизни, однако здесь кроется та опасность, что люди для этой теории — всего лишь цифры, а потери, понесенные в сражении — не более, чем один из элементов игры.

Простоты ради будем рассматривать стратегические игры только для двух участников. В качестве одного из участников может высту-

пать природа. Игроком может быть некое лицо — военачальник или глава сообщества либо даже сама армия или само сообщество. Кроме того, будем считать, что выигрыш одного игрока в точности совпадает с проигрышем другого. Иными словами, если один игрок выиграл 5 фунтов, то другой эти 5 фунтов проиграл. Такая игра называется *игрой двух лиц с нулевой суммой*. Далее приводятся четыре простых примера такого рода игр.

1. Выбрасывание карт. У Анн 2 карты — белая и черная с цифрой 1 на каждой. У Билла тоже две карты, но одна черная с цифрой 1, а другая белая с цифрой 0. Цвет и цифры обозначены только на одной стороне, так что их можно скрыть от противника. Анн и Билл одновременно открывают одну из своих карт. Если цвета совпадают, то Анн выигрывает сумму (в фунтах стерлингов), равную разности показанных цифр. Если же карты разного цвета, то эту сумму получает Билл.

2. Выбрасывание монет. У Клэр и Дерека — у каждого по пенни. Они выбрасывают монеты на стол, закрывая их от соперника. Если выясняется, что обе монеты легли одной стороной, то их забирает Клэр; если же на одной монете — орел, а на другой — решетка, то монеты достаются Дереку.

3. Выбрасывание пальцев. Эдвард и Фиона одновременно показывают друг другу один или два пальца. Если общее число показанных пальцев равно 2 или 4, то именно такую сумму (в долларах) выиграет Эдвард. Если же было показано 3 пальца, то Фиона выиграет 3 доллара.

4. Чет-нечет. Кора зажимает в кулак либо 1, либо 2, либо 3, либо 4 монеты (достоинством в 1 доллар). Инга должна угадать: «чет» или «нечет». Если она угадала, то проиграла Кора, в противном случае Инге придется заплатить ту сумму, которая оказалась в руке у Коры.

Предложите своим друзьям сразиться.

Чтобы разобраться в таких играх, прежде всего нужно построить *платежную матрицу* (матрица — это двумерный массив, или таблица). Сначала выписываются все возможные ходы каждого игрока. Например, в игре «выбрасывание карт» у Анн всего два хода: пойти либо черной, либо белой картой. Обозначим их соответственно АЧ и АБ. У Билла также два хода, которые мы обозначим БЧ и ББ. Ходы Анн запишем по вертикали, а Билла — по горизонтали:

	БЧ	ББ
АЧ		
АБ		

Теперь надо выписать, сколько выиграет Анн в зависимости от того, какой ход сделает она и какой — Билл. Если ход Анн — АЧ, а ход Билла — БЧ, то Анн выиграет 0 (1 − 1 = 0). На пересечении соответствующих строки и столбца запишем 0. Аналогично заполним остальные элементы таблицы, используя отрицательные числа для проигрышей Анн. То, что получится в результате, называется *платежной матрицей*, которая описывает ожидаемые выигрыши Анн. Она также описывает возможные проигрыши Билла; фактически элементы матрицы с противоположным знаком соответствуют выигрышам Билла.

	БЧ	ББ
АЧ	0	−1
АБ	0	1

платежная матрица А

Платежную матрицу можно представить массивом A(I,J), где I = = 1, 2 и J = 1, 2. Таким образом,

A(1,1) = 0, A(1,2) = −1, A(2,1) = 0, A(2,2) = 1.

Ниже приводятся платежные матрицы для других примеров:

Выбрасывание монет

	ДО	ДР
КО	2	−2
КР	−2	2

платежная матрица К

Выбрасывание пальцев

	Ф1	Ф2
Э1	2	−3
Э2	−3	4

платежная матрица Э

Чет-нечет

	К1	К2	К3	К4
ИЧ	−1	2	−3	4
ИН	1	−2	3	−4

платежная матрица И

Какой стратегии придерживаться? Рассмотрим первую игру: Анн и Билл выбрасывают карты. Если Анн выбрасывает черную карту, то она либо выиграет 0, либо проиграет 1. Если же она пойдет белой картой, то выиграет либо 0, либо 1. Понятно, что Анн никогда не станет ходить черной картой, потому что независимо от хода Билла ей выгоднее пойти белой картой. Итак, Анн всегда должна выбрасывать белую карту. Теперь займемся Биллом. Если он будет ходить черной картой, то никогда ничего не проиграет, но и не выиграет. Если же он пойдет

белой картой, то сможет проиграть 1 (выиграть 1 он может только в том случае, если Анн выбросит черную карту, но мы уже знаем, что она этого никогда не сделает). Таким образом, Билл всегда будет ходить черной картой. Вывод: ход Билла всегда БЧ, ход Анн — АБ. В результате ни один из игроков ничего не выиграет и не проиграет. Отклониться от этой стратегии никто из них не может — иначе он проиграет.

Это пример *строго определенной* игры. У каждого игрока в сущности только один ход, так что игра им довольно быстро должна наскучить. Угадать, кто станет победителем в строго определенной игре, — дело нехитрое, поскольку платежная матрица такой игры обладает одним интересным свойством: имеется элемент, одновременно являющийся минимальным в некоторой строке и максимальным в некотором столбце. Этот элемент называется точкой *минимакса* платежной матрицы. Например,

	БЧ	ББ
АЧ	0	−1
АБ	0	1

↑
точка минимакса

Если платежная матрица игры имеет точку минимакса, такая игра является строго определенной, и соперники должны выбирать тот ход, который соответствует строке и столбцу, содержащим точку минимакса. Значение минимакса называется *ценой* игры: тот игрок, чьи ходы выписаны по вертикали, в среднем выигрывает такую сумму. В рассмотренном выше примере цена игры равнялась нулю; в этом случае говорят, что игра *справедливая*.

Строго определенные игры — занятие довольно нудное (разве что ваш соперник не подозревает, что это именно такая игра). Большинство игр таковыми не являются, и три остальных примера — тоже не строго определенные игры; можете убедиться в этом, попытавшись найти точку минимакса.

В трех этих играх ситуация несколько более сложная. Лучшая стратегия каждого игрока здесь не столь очевидна. Однако, если участники играют продолжительное время, понятно, что ни один из игроков не будет делать все время один и тот же ход. Например, если при выбрасывании монет Клэр всегда будет выбирать орла, то Дерек вскоре это заметит и станет выбрасывать свою монету решеткой, и т. п. На самом деле лучшая стратегия и для Клэр и для Дерека — случайным образом выбирать орла и решетку (они могут, например, просто подбрасывать монету и закрывать ее при падении). В результате по прошествии некоторого времени игра станет справедливой.

Точно так же обстоит дело и с выбрасыванием пальцев. Чтобы

их линия поведения стала оптимальной, Эдвард и Фиона должны выбирать свои ходы случайным образом, но с некоторым предпочтением выбрасывания одного пальца. На самом деле можно показать, что Эдварду нужно выбрасывать 1 палец на протяжении 7/12 периода игры и 2 пальца — остальное время (т. е. 5/12 периода). Причем каждый раз число показанных пальцев должно выбираться случайным образом. Это и будет лучшей линией поведения Эдварда — его *оптимальной стратегией*. Осуществить это можно, например, при помощи стопки из 12 карт, на семи из которых — цифра 1, а на остальных 2. Или же Эдвард может воспользоваться своим компьютером для получения случайных чисел между 0 и 1. Если полученное число меньше 7/12, Эдварду следует ходить Э1, в противном случае ему надо выбросить два пальца.

Оптимальная стратегия Фионы точно такая же: ей надо выбрасывать 1 палец в 7 из 12 случаев и 2 пальца — в 5 из 12. Отклонение от этих оптимальных стратегий ни к чему хорошему не приведет. Придерживаясь же их, игроки минимизируют свой проигрыш.

Эта игра не является справедливой, потому что когда оба игрока придерживаются своей оптимальной стратегии, средние потери Эдварда составляют 1/12. Мы говорим, что *цена этой игры* есть $-1/12$.

Почему же цена игры равна $-1/12$? Чтобы ответить на этот вопрос, представим себе, что Эдвард и Фиона выбрасывают пальцы в течение долгого времени, например, 1440 раз. Предполагается, что каждый игрок ведет себя оптимальным образом, так что в среднем Эдвард выбросит 1 палец (приблизительно) 840 раз и 2 пальца — 600 раз (поскольку $1440*7/12 = 840$ и $1440*5/12 = 600$). За то время, когда Эдвард 840 раз выбросит 1 палец, мы ожидаем, что Фиона выбросит 1 палец в течение 7/12 этого периода, т. е. 490 раз, и два пальца — 350 раз. Аналогично, в течение того времени, когда Эдвард 600 раз выбросит 2 пальца, Фиона выбросит 1 палец 350 раз и 2 пальца — 250 раз (так как $600*7/12 = 350$ и $600*5/12 = 250$). Таким образом, ожидаемый выигрыш Эдварда в матче из 1440 сражений составит

$$490*2 + 350*(-3) + 350*(-3) + 250*4 = -120.$$

Итак, его ожидаемый выигрыш за одно сражение равен $-120/1440 = -1/12$.

Вы можете проделать подобные вычисления в предположении, что Эдвард придерживается стратегии, отличающейся от описанной выше. И тут же убедитесь, что ожидаемый проигрыш Эдварда будет еще больше.

Стратегия игрока — это просто список неотрицательных чисел, сумма которых равна 1. Каждому из возможных ходов игрока соответствует одно число; оно выражает пропорцию, в которой тот или

иной ход повторяется во время игры. Например, (0.33, 0.67) должно означать, что доля первого хода составляет 0.33, или 33%, а второго — 0.67, или 67%. Стратегия называется *оптимальной*, если, придерживаясь ее, игроки минимизируют свои потери.

Решить игру — это значит определить оптимальные стратегии каждого игрока и цену игры. Для решения игр двух участников с нулевой суммой разработан ряд методов. Если размер платежной матрицы невелик, можно воспользоваться графическими методами. В общем же случае решение легко получить при помощи линейного программирования (симплекс-метод). Подробное изложение этой техники выходит за рамки наших возможностей; тем не менее мы включили в книгу программу ИГРА ДВУХ ЛИЦ С НУЛЕВОЙ СУММОЙ, в которой для решения таких игр используется симплекс-метод. Заметим, что в программе не предусмотрена проверка, является ли игра строго определенной — вам следует проделать это самостоятельно, что не составит труда. Кроме того, часто у каждого игрока имеется больше одной оптимальной стратегии — программа же выявляет только одну. Иногда для определения другой оптимальной стратегии достаточно выписать ходы игроков в ином порядке.

```
10 REM              ***********************************
20 REM              *                                 *
30 REM              * ИГРА ДВУХ ЛИЦ С НУЛЕВОЙ СУММОЙ *
40 REM              *                                 *
50 REM              ***********************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 PRINT CS; : REM ОЧИСТИТЬ ЭКРАН
130 PRINT "МАТРИЦА ДЛЯ ИГРЫ ДВУХ ЛИЦ"
140 PRINT "КАКОЙ РАЗМЕР ИГРЫ?"
150 PRINT "ЧИСЛО СТРОК ";
160 INPUT M
170 LET M=INT(M+.5)
180 PRINT "    СТОЛБЦОВ ";
190 INPUT N
200 LET N=INT(N+.5)
210 IF M<2 OR N<2 THEN GOTO 120
220 LET T=M+N+1
230 DIM A(M+1,T+1)
240 DIM P(M)
250 DIM Q(N)
260 LET MIN=0
270 PRINT
280 PRINT "ВВОД ПЛАТЕЖНОЙ МАТРИЦЫ
290 PRINT
300 FOR I=1 TO M
310 FOR J=1 TO N
320 PRINT "СТРОКА";I;"СТОЛБЕЦ";J;
330 INPUT A(I,J)
340 IF A(I,J)<MIN THEN MIN=A(I,J)
```

В этой части вводится платежная матрица для решаемой игры. После программы приведен типичный дисплей. Затем определяется минимальное отрицательное значения и из него вычитается 1. Результат присваивается переменной MIN.

```
350 NEXT J
360 PRINT
370 NEXT I
380 LET MIN=MIN-1
400 REM XXXXXXXXXXX МАССИВ НА ДИСПЛЕЙ И ВЫЧИТАНИЕ MIN XXXXXXXXXXXXXX
410 PRINT CS%; : REM ОЧИСТИТЬ ЭКРАН
420 FOR I=1 TO M
430 PRINT "   ";
440 FOR J=1 TO N
450 PRINT A(I,J);
460 LET A(I,J)=A(I,J)-MIN
470 NEXT J
480 PRINT
490 NEXT I
495 PRINT
500 REM XXXXXXXXXXXXXXXX РАСШИРЕНИЕ МАССИВА XXXXXXXXXXXXXXXXXXXXXXX
510 FOR I=1 TO M+1
520 FOR J=N+1 TO M+N
530 LET A(I,J)=0
540 NEXT J
550 NEXT I
560 FOR J=1 TO N
570 LET A(M+1,J)=-1
580 NEXT J
590 FOR I=1 TO M
600 LET A(I,N+I)=1
610 LET A(I,T)=1
620 LET A(I,T+1)=0
630 NEXT I
640 LET A(M+1,T)=0
650 PRINT "ОБРАБОТКА"
700 REM XXXXXXXXXXXXXXXXX ВЫЧИСЛИТЕЛЬНЫЙ ЦИКЛ XXXXXXXXXXXXXXXXXXXX
710 PRINT "*";
720 REM XXXXXX ВЫБОР СТОЛБЦА S XXXXXX
730 LET S=1
740 FOR J=2 TO N+M
750 IF A(M+1,J)<A(M+1,S) THEN LET S=J
760 NEXT J
770 REM XXX ПРОВЕРКА НА ОКОНЧАНИЕ XXXX
780 IF A(M+1,S))-1.0E-20 THEN GOTO 1110
790 REM XXXXXX ВЫБОР СТРОКИ R XXXXXXX
800 LET R=0
810 LET R=R+1
820 IF A(R,S)<1.0E-20 THEN GOTO 810
830 FOR I=R+1 TO M
840 LET C=A(R,N+M+1)/A(R,S)
850 IF A(I,S)<1.0E-20 THEN GOTO 870
860 IF A(I,T)/A(I,S)<C THEN LET R=I
870 NEXT I
880 REM XXXXXXX РЕДУКЦИЯ XXXXXXXXXXXX
890 LET A(R,T+1)=S
900 IF S>N THEN LET A(R,T+1)=0
910 LET W=A(R,S)
920 FOR J=1 TO T
930 LET A(R,J)=A(R,J)/W
940 NEXT J
950 FOR I=1 TO M+1
960 IF I=R THEN GOTO 1020
970 LET W=A(I,S)
980 IF ABS(W)<1.0E-20 THEN GOTO 1020
```

На экран выводится платежная матрица, из каждого числа которой вычтено значение MIN. В результате все числа массива становятся положительными.

Для удобства вычислений массив A(I, J) расширяется.

Основной вычислительный цикл. Строка R и столбец S выбираются в соответствии с некоторыми правилами; массив A(I, J) при этом изменяется. Процесс повторяется до тех пор, пока не выполнится некоторое условие. После каждого прохождения цикла на экран выводится *.

```
990 FOR J=1 TO T
1000 LET A(I,J)=A(I,J)-A(R,J)*W
1010 NEXT J
1020 NEXT I
1030 GOTO 710
1100 REM XXXXXXXXXXXXXXXXXX ОПТИМАЛЬНЫЕ СТРАТЕГИИ XXXXXXXXXXXXXXXXXX
1110 LET V=1/A(M+1,N+M+1)
1120 REM XXXXXX ИГРОК ПО СТРОКАМ XXXXXX
1130 FOR I=1 TO M
1140 LET P(I)=A(M+1,N+I)*V
1150 NEXT I
1160 REM XXXXXXX ИГРОК ПО СТОЛБЦАМ XXXXX
1170 FOR J=1 TO M
1180 LET Q(A(J,T+1))=A(J,T)*V
1190 NEXT J
1200 REM XXXXXXXXXXXXXXX ВЫВОД НА ЭКРАН РЕЗУЛЬТАТА XXXXXXXXXXXXXXX
1210 PRINT "   ПОЛУЧЕНЫ"
1220 PRINT
1230 PRINT "ОПТИМАЛЬНЫЕ СТРАТЕГИИ"
1240 PRINT
1250 PRINT "ИГРОКА ПО СТРОКАМ"
1260 FOR I=1 TO M
1270 PRINT " ' ";INT(1000*P(I)+.5)/1000
1280 NEXT I
1290 PRINT
1300 PRINT "ИГРОКА ПО СТОЛБЦАМ"
1310 FOR J=1 TO N
1320 PRINT INT(1000*Q(J)+.5)/1000;
1330 NEXT J
1340 PRINT
1350 PRINT
1360 PRINT "ЦЕНА ИГРЫ ДЛЯ ИГРОКА"
1370 PRINT "ПО СТРОКАМ = ";INT(1000*(V+MIN)+.5)/1000
1400 REM XXXXXXXXXXXXXXXX ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1410 GET G$ : REM LET G$=INKEY$
1420 IF G$="" THEN GOTO 1410
1430 PRINT
1440 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
1450 GET G$ : REM LET G$=INKEY$
1460 IF G$<>"Y" AND G$<>"N" THEN GOTO 1450
1470 IF G$="Y" THEN RUN
```

Вычисляются оптимальные стратегии.

На экран выводятся оптимальные стратегии и цена игры.

Дополнительные замечания см. в приложении.

```
МАТРИЦА ДЛЯ ИГРЫ ДВУХ ЛИЦ

КАКОЙ РАЗМЕР ИГРЫ?

ЧИСЛО СТРОК? 3
   СТОЛБЦОВ? 3

ВВОД ПЛАТЕЖНОЙ МАТРИЦЫ

СТРОКА 1 СТОЛБЕЦ 1 ? .3
СТРОКА 1 СТОЛБЕЦ 2 ? -.7
СТРОКА 1 СТОЛБЕЦ 3 ? .7

СТРОКА 2 СТОЛБЕЦ 1 ? -.3
СТРОКА 2 СТОЛБЕЦ 2 ? .7
СТРОКА 2 СТОЛБЕЦ 3 ? .3

СТРОКА 3 СТОЛБЕЦ 1 ? .4
СТРОКА 3 СТОЛБЕЦ 2 ? 0
СТРОКА 3 СТОЛБЕЦ 3 ? -.5
```

```
      .3 -.7  .7
     -.3  .7 -.3
      .4  0 -.5

ОБРАБОТКА **** ПОЛУЧЕНЫ

ОПТИМАЛЬНЫЕ СТРАТЕГИИ

ИГРОКА ПО СТРОКАМ
      .381
      .449
      .169

ИГРОКА ПО СТОЛБЦАМ
 .415  .347  .237

ЦЕНА ИГРЫ ДЛЯ ИГРОКА
ПО СТРОКАМ=  .047

 ПОВТОРИТЬ? Y ИЛИ N
```

Как делать деньги? Рокси и Толбот часто играют в одну игру, платежная матрица которой выписана ниже. Рокси выбирает одну из строк, а Толбот — один из столбцов. Числа, стоящие на пересечении соответствующих строк и столбцов платежной матрицы, обозначают выигрыши Рокси у Толбота. Таким образом, если Рокси выберет первую строку, а Толбот — третий столбец, то Рокси выиграет 56.

	(1)	(2)	(3)	(4)	(5)	
(1)	−7	16	56	−7	10	
(2)	1	8	5	−5	−3	
(3)	−5	74	43	−6	36	платежная матрица игрока
(4)	−6	25	81	−5	24	по строкам (Рокси)
(5)	−5	−2	9	1	25	

Если бы вы стали играть в эту игру, за кого вы предпочли бы выступать: за Рокси или Толбота? Многие думают, что выгоднее быть игроком по строкам. На самом же деле положение игрока по столбцам значительно лучше. Если вам удастся уговорить кого-нибудь сыграть в эту игру на деньги при том условии, что вы будете играть по столбцам, то сможете слегка заработать. Дело в том, что цена этой игры отрицательна: −2. Таким образом, если вы будете придерживаться оптимальной стратегии, то выиграете в среднем 2; не исключено, что даже чуть больше.

Оптимальная стратегия игрока по столбцам состоит в выборе случайным образом и одинаково часто только двух столбцов: 1 и 4. В самом деле, вам должно быть понятно, что никогда не следует выбирать столбцы 2, 3 и 5. Например, никогда не следует выбирать столбец 2, потому что выигрыш по столбцу 1 *всегда* больше, какую бы строку ни выбрал ваш противник. Мы говорим, что столбец 1 *предпочтительнее* столбца 2. По той же причине столбец 1 предпочтительнее столбца 3 и столбец 4 предпочтительнее столбца 5. Вычеркнув эти три столбца, получим следующую редуцированную матрицу:

	(1)	(4)
(1)	−7	−7
(2)	1	−5
(3)	−5	−6
(4)	−6	−5
(5)	−5	1

Теперь уже понятно, почему эта игра чрезвычайно несправедлива для игрока по строкам.

Если вам удастся склонить кого-нибудь к участию в игре, не признавайтесь, что собираетесь ходить лишь по столбцам 1 и 4. Можете придать игре видимость справедливой, используя 5 карт с цифрой 1 и другие 5 с цифрой 4. Цифры, изображенные на картах, своему противнику, разумеется, не показывайте. Затем пусть он назовет номер строки — и можете переворачивать верхнюю карту. Вычислите по платежной матрице свой выигрыш или проигрыш. Можете ли вы проиграть?

Компьютерные стратегические игры. На основе описанных выше идей очень легко создавать занимательные игры при помощи компьютера. Простым примером тому служит программа ВЛОЖЕНИЕ КАПИТАЛА. Предполагается, что неизвестный благодетель пожертвовал вам 1000 долларов с тем условием, что вы вложите их в три предприятия: X, Y и Z. Вы обязаны вкладывать все деньги в течение 10 лет. Необходимо каждый год сообщать, какая сумма будет вложена в то или иное предприятие. У вас есть довольно мощный компьютер и свой человек в каждой фирме. Так что вы способны оценивать прибыль (убыток) в расчете на один доллар в каждой из фирм в зависимости от состояния рынка. Всего возможно 3 состояния, и компьютер вам подскажет, с какой вероятностью то или иное состояние может возникнуть. Прибыль или убыток на 1 доллар выводятся на экран дисплея в виде таблицы (ниже приведен пример):

```
   СОСТОЯНИЯ РЫНКА
   ST 1  ST 2  ST 3
   .222  .413  .365

X   .4   -.1    .3
Y  -.2    .1    .1
Z  -.1    .4   -.3
```

Здесь вероятность первого состояния ST 1 равна 0.222 (т. е. в 22.2% случаев). Если на рынке наблюдается состояние ST 1, то в предприятии X ваша прибыль с 1 доллара составит 0.40 доллара, в фирме Y вы понесете убыток 0.20 доллара на 1 доллар, а в фирме Z вы на каждом долларе потеряете 0.10 доллара. Таким образом, если вы вложили в предприятия X, Y и Z соответственно суммы 300, 300 и 400, а на рынке было состояние 1, то ваш общий доход составит

$$300*0.4 + 300*(-0.2) + 400*(-0.1) = 20$$

и теперь вы станете обладателем 1020 долларов.

В конце каждого года вам придется принимать решение, куда вкладывать деньги. Достаточно указать только, сколько вы вложите в X и Y, а остаток компьютер автоматически переведет в Z.

Вам отводится 10 лет на то, чтобы разбогатеть. Каждая игра имеет цену 0.05 или около того. Таким образом, если вы вкладываете деньги оптимальным образом, ваш капитал должен увеличиваться в среднем не менее чем на 5% ежегодно. Играя рискованно, вы можете разбогатеть (или разориться) чуть больше. Желаем удачи.

```
10 REM                  **********************
20 REM                  *                    *
30 REM                  * ВЛОЖЕНИЕ КАПИТАЛА *
40 REM                  *                    *
50 REM                  **********************
60 REM
70 REM
```

```
100 REM XXXXXXXXXX..XXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 DIM A(4,3,10)
130 FOR K=1 TO 10
140 FOR I=1 TO 4
150 FOR J=1 TO 3
160 READ A(I,J,K)
170 NEXT J
180 NEXT I
190 NEXT K
200 LET W=1000
210 LET YEAR=1
```

В память записывается 10 различных платежных матриц в виде массивов A(I, J, K). В них содержится информация о вероятностях состояний рынка.

```
300 REM XXXXXXXXXXXXXXXXXXXXX ЦИКЛ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
310 PRINT CS; : REM ОЧИСТИТЬ ЭКРАН
315 PRINT "   ВЛОЖЕНИЕ КАПИТАЛА"
320 PRINT "   ГОД";YEAR
330 PRINT
340 PRINT "КАПИТАЛ В НАЧАЛЕ ГОДА РАВЕН ";W
350 PRINT
360 PRINT "    СОСТОЯНИЯ РЫНКА"
370 PRINT "    ST 1  ST 2  ST 3"
380 LET K=INT(RND(1)*10+1)
390 FOR I=1 TO 4
400 IF I=2 THEN PRINT " X";
410 IF I=3 THEN PRINT " Y";
420 IF I=4 THEN PRINT " Z";
430 FOR J=1 TO 3
440 IF I=1 THEN PRINT TAB(-3+6*J);A(I,J,K);
450 IF I>1 THEN PRINT TAB(-2+6*J);A(I,J,K);
460 NEXT J
470 IF I=1 THEN PRINT
480 PRINT
490 NEXT I
500 PRINT
510 PRINT "СКОЛЬКО ВЫ ХОТИТЕ ВЛОЖИТЬ"
520 PRINT "В X";
530 INPUT X
540 LET X=INT(X+.5)
550 IF X<0 OR X>W THEN PRINT "ОШИБКА - ПОПРОБУЙТЕ ЕЩЕ"
560 IF X<0 OR X>W THEN GOTO 520
570 PRINT "В Y";
580 INPUT Y
590 LET Y=INT(Y+.5)
600 IF Y<0 OR Y>W-X THEN PRINT "ОШИБКА - ПОПРОБУЙТЕ ЕЩЕ"
610 IF Y<0 OR Y>W-X THEN GOTO 570
620 LET Z=W-X-Y
630 PRINT "ОСТАТОК";Z;"В Z"
```

Число K выбирается из диапазона 1—10 случайным образом. На экран выводится соответствующая платежная матрица. Компьютер запрашивает о размерах вкладов в X и Y. Состояние рынка определяется при помощи случайного числа Q в строке 650.

```
640 REM XXXX СОСТОЯНИЕ РЫНКА XXXXXXXX
650 LET Q=RND(1)
660 LET R=3
670 IF Q<A(1,1,K)+A(1,2,K)THEN LET R=2
680 IF Q<A(1,1,K) THEN LET R=1
690 PRINT "СОСТОЯНИЕ РЫНКА";R


700 REM XXXX ПРИБЫЛЬ И УБЫТОК XXXXXXXXXX
710 LET W=W+A(2,R,K)*X+A(3,R,K)*Y+A(4,R,K)*Z
720 LET W=INT(W+.5)
730 PRINT "ВАШ КАПИТАЛ ТЕПЕРЬ РАВЕН"; W
740 PRINT
750 PRINT "  ЧТОБЫ ПРОДОЛЖИТЬ НАЖМИТЕ Y"
760 GET G$ : REM LET G$=INKEY$
770 IF G$<>"Y" THEN GOTO 760
780 LET YEAR=YEAR+1
790 IF YEAR<11 THEN GOTO 310
```

Вычисляется прибыль от вложенного капитала или убыток.

```
800 REM XXXXXXXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXX
810 PRINT
820 PRINT "ВЫ НАЧАЛИ С 1000, А ТЕПЕРЬ У ВАС";W
830 PRINT
840 LET B$="* ПОТРЯСАЮЩИЙ РЕЗУЛЬТАТ *"
850 IF W<2500 THEN LET B$="         * ОТЛИЧНО *"
860 IF W<2000 THEN LET B$="         * ХОРОШО *"
870 IF W<1750 THEN LET B$="         * НЕПЛОХО *"
880 IF W<1400 THEN LET B$="       * ЖИТЬ МОЖНО *"
890 IF W<1000 THEN LET B$=" В ДРУГОЙ РАЗ ПОВЕЗЕТ"
900 PRINT B$
910 PRINT
```

B$ — мнение компьютера о том, насколько удачно вы вложили свои деньги.

```
1000 REM XXXXXXX ПОВТОРЕНИЕ XXXXXXXXXXXXXX
1010 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
1020 GET G$ : REM LET G$=INKEY$
1030 IF G$<>"Y" AND G$<>"N" THEN GOTO 1020
1040 IF G$="Y" THEN GOTO 200
1050 END : REM STOP


1100 PRINT XXXXXXXXXXXXXXXXXXXXXXX ДАННЫЕ XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1110 DATA .478,.337,.185,.3,-.7,.7,-.3,.7,-.3,.2,.1,-.5
1120 DATA .106,.53,.364,-.5,-.3,.7,.5,-.5,.7,.3,.5,-.7
1130 DATA .415,.347,.237,.3,-.7,.7,-.3,.7,-.3,.4,0.-.5
1140 DATA .274,.346,.38,.5,-.7,.4,-.3,.7,-.3,-.5-.2,.3
1150 DATA .437,.438,.125,.6,-.6,.4,-.6,.8,-.3,-.3,.5,-.3
1160 DATA .304,.435,.261,.8,-.7,.4,-.6,.4,.2,-.2,.6,-.6
1170 DATA .422,.39,.188,-.5,.8,-.3,.7,-.5.-.3,-.3,.1,.7
1180 DATA .214,.357,.429,.3,-.2,.1,-.3,.4.-.1,.1.-.2,.2
1190 DATA .222,.413,.365,.4,-.4,.3,-.2,.1,.1,-.1,.4,-.3
1200 DATA .093,.463,.444,0,-.6,.7,-.6,0,.2,.2,.8,-.8
```

Дополнительные замечания см. в приложении.

```
ВЛОЖЕНИЕ КАПИТАЛА

ГОД 5

КАПИТАЛ В НАЧАЛЕ ГОДА
РАВЕН  1854

   СОСТОЯНИЯ РЫНКА
   ST 1  ST 2  ST 3
   .222  .413  .365

X   .4   -.4    .3
Y  -.2    .1    .1
Z  -.1    .4   -.3

СКОЛЬКО ВЫ ХОТИТЕ ВЛОЖИТЬ
В X? 1854
В Y? 0
ОСТАТОК 0 В Z
СОСТОЯНИЕ РЫНКА 3
ВАШ КАПИТАЛ ТЕПЕРЬ РАВЕН
 2410
```

```
   СОСТОЯНИЯ РЫНКА
   ST 1  ST 2  ST 3
   .274  .346  .38

X   .5   -.7    .4
Y  -.3    .7   -.3
Z  -.5    .2    .3

СКОЛЬКО ВЫ ХОТИТЕ ВЛОЖИТЬ
В X? 0
В Y? 133
ОСТАТОК 0 В Z
СОСТОЯНИЕ РЫНКА 2
ВАШ КАПИТАЛ ТЕПЕРЬ РАВЕН
 226

  ЧТОБЫ ПРОДОЛЖИТЬ НАЖМИТЕ Y

ВЫ НАЧАЛИ С 1000,
А ТЕПЕРЬ У ВАС 226

В ДРУГОЙ РАЗ ПОВЕЗЕТ
```

В строках 1100—1200 программы ВЛОЖЕНИЕ КАПИТАЛА содержатся необходимые данные для массива A(I, J, K). Если в вашем компьютере команда DATA отсутствует, то массив A(I, J, K) следует строить по-другому. Ниже показано, как это делается для машины ZX 81.

```
10 REM                          ********************
20 REM                          *      ZX 81       *
30 REM                          * INVESTMENT GAME *
40 REM                          *                  *
50 REM                          ********************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXX SETTING UP FOR THE ZX 81 XXXXXXXXXXXXXXXXXXXXX
120 DIM A(4,3,10)
130 FOR K=1 TO 10
135 GOSUB 1100+K*10
140 FOR I=1 TO 4
150 FOR J=1 TO 3
160 LET S=12*I+4*J-15
165 LET A(I,J,K)=VAL(A$(S TO S+3))
170 NEXT J
180 NEXT I
190 NEXT K
200 LET W=1000
210 LET YEAR=1
```

Используйте строки 300—1050 из программы ВЛОЖЕНИЕ КАПИТАЛА.

```
1100 REM XXXXXXXXXXXXXXXXXXXXX DATA XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1110 LET A$=".478.337.1850.30-0.70.70-0.30.70-0.30.200.10-0.5"
1115 RETURN
1120 LET A$=".106.530.364-0.5-0.30.700.50-0.50.700.300.50-0.7"
1125 RETURN
1130 LET A$=".415.347.2370.30-0.70.70-0.30.70-0.30.400000-0.5"
1135 RETURN
1140 LET A$=".274.346.3800.50-0.70.40-0.30.70-0.3-0.50.200.30"
1145 RETURN
1150 LET A$=".437.438.1250.60-0.60.40-0.60.80-0.3-0.30.50-0.3"
1155 RETURN
1160 LET A$=".304.435.2610.80-0.70.40-0.60.400.20-0.20.60-0.6"
1165 RETURN
1170 LET A$=".422.390.188-0.50.80-0.30.70-0.5-0.3-0.30.100.70"
1175 RETURN
1180 LET A$=".214.357.4290.30-0.20.10-0.30.40-0.10.10-0.20.20"
1185 RETURN
1190 LET A$=".222.413.3650.40-0.40.30-0.20.100.10-0.10.40-0.3"
1195 RETURN
1200 LET A$=".093.463.4440000-0.60.70-0.600000.200.200.80-0.8"
1205 RETURN
```

Приближенные решения. Решение стратегической игры можно найти и другим способом. Представьте себе, что каждый игрок записывает все ходы, сделанные им в течение нескольких сражений. Если игроки принимают решение на основе среднего хода своего соперника, то после большого количества ходов каждый сможет выработать свою оптимальную стратегию. Чтобы получить оптимальную стратегию таким методом, нет никакой необходимости по-настоящему играть. Достаточно того, чтобы каждый из игроков по очереди говорил, как он

или она собираются пойти, причем решение они должны принимать, исходя из среднего, вычисленного по всем предыдущим ходам своего противника.

Проиллюстрируем этот метод на примере игры Эдварда и Фионы, выбрасывающих пальцы. Вот как здесь выглядит платежная матрица:

$$\begin{matrix} 2 & -3 \\ -3 & 4 \end{matrix}$$ (платежная матрица Эдварда)

Игра начинается с того, что Эдвард выбирает любую строку и записывает ее под матрицей. Предположим, он выбрал строку 1:

$$\begin{matrix} 2 & -3 \\ -3 & 4 \\ \hline 2 & -3 \end{matrix}$$

Поглядев на это, Фиона выбирает столбец, соответствующий меньшему числу строки. Это столбец 2, который она и записывает справа от матрицы:

$$\begin{matrix} 2 & -3 & | & -3 \\ -3 & 4 & | & 4 \\ \hline 2 & -3 & & \end{matrix}$$

Теперь Эдвард, взглянув на этот столбец, выбирает строку, соответствующую наибольшему числу столбца. Это строка 2. Эдвард складывает ее почленно со строкой, выбранной им раньше, и результат записывает под предыдущей строкой:

$$\begin{matrix} 2 & -3 & | & -3 \\ -3 & 4 & | & 4 \\ \hline 2 & -3 & & \\ -1 & 1 & & \end{matrix}$$

Поглядев на новую строку, Фиона выбирает столбец, соответствующий наименьшему числу в строке. Это столбец 1. Она складывает его почленно с предыдущим и результат записывает рядом:

$$\begin{matrix} 2 & -3 & | & -3 & -1 \\ -3 & 4 & | & 4 & 1 \\ \hline 2 & -3 & & & \\ -1 & 1 & & & \end{matrix}$$

Процесс этот повторяется многократно. Если окажется, что числа совпадают, то прибегают к случайному выбору.

Ниже приведен результат игры из 10 сражений. Наименьшие числа строк и наибольшие числа столбцов подчеркнуты.

2	−3	−3	−1	1	3	0	−3	−1	1	3	0	6/10
−3	4	4	1	−2	−5	−1	3	0	−3	−6	−2	4/10

2	−3
−1	1
−4	5
−2	2
0	−1
2	−4
−1	0
−4	4
−2	1
0	−2
6/10	4/10

Стратегии можно (приближенно) вычислить, поделив подчеркнутые числа каждой строки и каждого столбца на 10. Таким образом, приблизительно оптимальной стратегией Эдварда будет (0.6, 0.4), а Фионы — (0.6, 0.4). Более точный ответ получается при достаточно большом числе сражений.

Для такого поиска оптимальной стратегии великолепно подойдет компьютер, снабженный программой ПРИБЛИЗИТЕЛЬНЫЙ АНАЛИЗ ИГРЫ. Компьютер выполняет описанный процесс более 100 раз. Можете увеличить это число и получить соответственно более точный результат.

Хотя программа ИГРА ДВУХ ЛИЦ С НУЛЕВОЙ СУММОЙ дает точное решение в матричной игре, причем очень быстро, математическая сторона весьма усложнена и, несомненно, для многих программа не что иное, как черный ящик. Математический аспект программы ПРИБЛИЗИТЕЛЬНЫЙ АНАЛИЗ ИГРЫ много проще; но здесь вы теряете на том, что имеете лишь приближенное решение.

```
10 REM                ******************************
20 REM                *                            *
30 REM                * ПРИБЛИЗИТЕЛЬНЫЙ АНАЛИЗ ИГРЫ *
40 REM                *                            *
50 REM                ******************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET G=100
130 PRINT CS; : REM ОЧИСТИТЬ ЭКРАН
```

```
140 PRINT "МАТРИЦА ДЛЯ ИГРЫ ДВУХ ЛИЦ "
150 PRINT "КАКОЙ РАЗМЕР ИГРЫ?"
160 PRINT "ЧИСЛО СТРОК";
170 INPUT M
180 LET M=INT(M+.5)
190 PRINT "   СТОЛБЦОВ";
200 INPUT N
210 LET N=INT(N+.5)
220 IF M<2 OR N<2 THEN GOTO 130
230 DIM A(M,N)
240 DIM P(M)
250 DIM Q(N)
260 DIM X(M)
270 DIM Y(N)
280 PRINT
290 PRINT "ВВОД ПЛАТЕЖНОЙ МАТРИЦЫ "
300 FOR I=1 TO M
310 FOR J=1 TO N
320 PRINT "СТРОКА";I;"СТОЛБЕЦ";J;
330 INPUT A(I,J)
340 NEXT J
350 PRINT
360 NEXT I
```

Здесь вводится (INPUT) платежная матрица. Нужны два дополнительных массива. В них хранятся дополнительные строки и столбцы, участвующие в процессе.

```
400 REM XXXXXXXXXXXXXXXXX ВЫВОД НА ДИСПЛЕЙ МАССИВА XXXXXXXXXXXXXXXXXXX
410 PRINT CS%; : REM ОЧИСТИТЬ ЭКРАН
420 FOR I=1 TO M
430 PRINT "   ";
440 FOR J=1 TO N
450 PRINT A(I,J);
460 NEXT J
470 PRINT
480 NEXT I
490 PRINT
```

На дисплей выводится платежная матрица.

```
500 REM XXXXXXXXXXXXXXXXXXXX ИГРЫ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
510 PRINT "ОБРАБОТКА";
520 LET R=INT(M*RND(1)+1)
530 FOR I=0 TO 9
540 FOR U=1 TO (G+1)/10
550 FOR J=1 TO N
560 LET Y(J)=Y(J)+A(R,J)
570 NEXT J
580 GOSUB 1210
590 LET Q(C)=Q(C)+1
600 FOR I=1 TO M
610 LET X(I)=X(I)+A(I,C)
620 NEXT I
630 GOSUB 1110
640 LET P(R)=P(R)+1
650 NEXT U
660 PRINT "*";
670 NEXT T
```

В этой части проводится 100 партий игры с компьютером. Сначала компьютер случайным образом выбирает строку. Затем 100 раз повторяется описанный в тексте процесс.

```
680 PRINT "ПОЛУЧЕНЫ"
690 REM XXXXXX ЦЕНА XXXXXXX
700 LET V1=INT(Y(C)+.5)/G
710 LET V2=INT(X(R)+.5)/G
```

Если вы хотите, чтобы было сыграно больше партий, вместо значения 100 в строке 120 укажите 1000. Программа будет работать дольше, но выдаст более точный результат.

```
800 REM XXXXXXXXXXXXXXX ОКОНЧАТЕЛЬНЫЙ ВЫВОД НА ЭКРАН XXXXXXXXXXXXXXXXXX
810 PRINT
820 PRINT "ПРИБЛИЗИТЕЛЬНЫЕ ОПТИМАЛЬНЫЕ"
830 PRINT "СТРАТЕГИИ"
840 PRINT
850 PRINT "ИГРОКА ПО СТРОКАМ"
860 FOR I=1 TO M
870 PRINT "   ";P(I)/G
880 NEXT I
890 PRINT
900 PRINT "ИГРОКА ПО СТОЛБЦАМ"
910 FOR J=1 TO N
920 PRINT Q(J)/G;
930 NEXT J
940 PRINT
950 PRINT
960 PRINT "ЦЕНА ИГРЫ ДЛЯ ИГРОКА"
970 PRINT "ПО СТРОКАМ )=";V1
980 PRINT "          И (=";V2
```

На экран выводятся приблизительные оптимальные стратегии.

```
1000 REM XXXXXXXXXXXXXXXXXXX ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1010 PRINT
1020 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
1030 GET G$ : REM LET G$=INKEY$
1040 IF G$()"Y" AND G$()"N" THEN GOTO 1030
1050 IF G$="Y" THEN RUN
1060 END : REM STOP
1100 REM XXXXXXXXXX ПОДПРОГРАММА ДЛЯ MAX И MIN XXXXXXXXXXXXXXXXXXXXXXXX
```

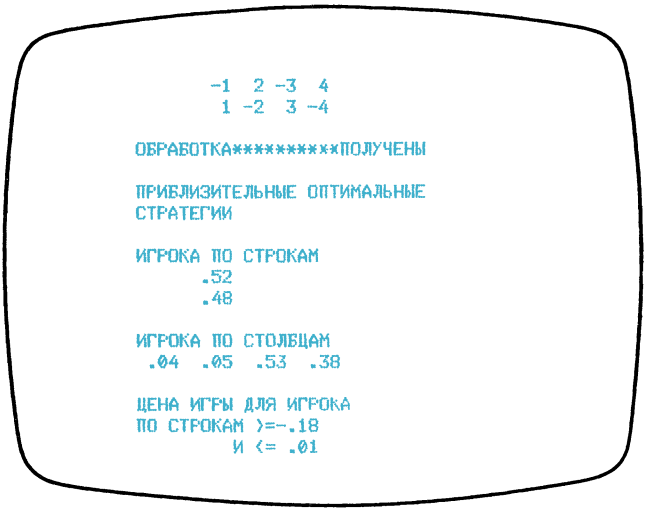
```
1200 REM XXXXXXXXXXXXX MAX XXXXXXXXXXXXXX
1210 LET R=1
1220 FOR I=2 TO M
1230 IF X(I)=X(R) AND RND(1)(0.5 THEN LET R=I
1240 IF X(I))X(R) THEN LET R=I
1250 NEXT I
1260 RETURN
```

```
1300 REM XXXXXXXXXXXXX MIN XXXXXXXXXXXXXX
1310 LET C=1
1320 FOR J=2 TO N
1330 IF Y(J)=Y(C) AND RND(1)(0.5 THEN LET C=J
1340 IF Y(J)(Y(C) THEN LET C=J
1350 NEXT J
1360 RETURN
```

Подпрограммы для определения максимума (минимума) из совокупности чисел. Если таких чисел несколько, компьютер выбирает произвольное.

```
МАТРИЦА ДЛЯ ИГРЫ ДВУХ ЛИЦ

КАКОЙ РАЗМЕР ИГРЫ?

ЧИСЛО СТРОК? 3
   СТОЛБЦОВ? 3

ВВОД ПЛАТЕЖНОЙ МАТРИЦЫ

СТРОКА 1 СТОЛБЕЦ 1 ? .3
СТРОКА 1 СТОЛБЕЦ 2 ? -.7
СТРОКА 1 СТОЛБЕЦ 3 ? .7

СТРОКА 2 СТОЛБЕЦ 1 ? -.3
СТРОКА 2 СТОЛБЕЦ 2 ? .7
СТРОКА 2 СТОЛБЕЦ 3 ? -.3

СТРОКА 3 СТОЛБЕЦ 1 ? .4
СТРОКА 3 СТОЛБЕЦ 2 ? 0
СТРОКА 3 СТОЛБЕЦ 3 ? -.5
```

```
      .3 -.7  .7
     -.3  .7 -.3
      .4  0 -.5

ОБРАБОТКА**********ПОЛУЧЕНЫ

ПРИБЛИЗИТЕЛЬНЫЕ ОПТИМАЛЬНЫЕ
СТРАТЕГИИ

ИГРОКА ПО СТРОКАМ
      .39
      .44
      .17

ИГРОКА ПО СТОЛБЦАМ
 .36   .35   .29

ЦЕНА ИГРЫ ДЛЯ ИГРОКА
ПО СТРОКАМ >= .05
        И <= .07
```



Любители пива. И наконец, игра для любителей пива. Как только двум посетителям бара приносят заказ, они решают, кто будет платить на этот раз. Каждый пишет какое-либо положительное целое число (большее нуля). Числа сравниваются и заказ в этот раз оплачивает тот, чье число больше, не менее чем на 2. Если же числа различаются только на единицу, то тот, чье число меньше, платит и в этот раз, и в следующий. Если числа оказались одинаковыми, они переигрывают. Игра эта справедливая. Спрашивается, какой должна быть оптимальная стратегия?

Если вы попытаетесь воспользоваться одной из программ ИГРА ДВУХ ЛИЦ С НУЛЕВОЙ СУММОЙ или ПРИБЛИЗИТЕЛЬНЫЙ АНАЛИЗ ИГРЫ, то столкнетесь с некоторым затруднением: а какой же размер матрицы? У каждого игрока бесконечно много вариантов!

Платежная матрица игры представлена ниже. Точки означают, что таблица не полна и ее можно продолжить.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	. . .
(1)	0	−2	1	1	1	1	1	1	1	. . .
(2)	2	0	−2	1	1	1	1	1	1	. . .
(3)	−1	2	0	−2	1	1	1	1	1	. . .
(4)	−1	−1	2	0	−2	1	1	1	1	. . .
(5)	−1	−1	−1	2	0	−2	1	1	1	. . .
(6)	−1	−1	−1	−1	2	0	−2	1	1	. . .
(7)	−1	−1	−1	−1	−1	2	0	−2	1	. . .
(8)	−1	−1	−1	−1	−1	−1	2	0	−2	. . .
(9)	−1	−1	−1	−1	−1	−1	−1	2	0	. . .
.	.	.	.	.	.	.	.	.	.	. . .

Можно, конечно, считать, что игроки никогда не называют очень больших чисел, но правильно ли это? Выполните программу ИГРА ДВУХ ЛИЦ С НУЛЕВОЙ СУММОЙ, введя ограничение, что каждый игрок может называть только небольшие числа, например, не больше 10. Попробуйте и другой вариант, скажем 9, 8, 7, 6, 5. Проделав это, вы обнаружите, что всегда получаются одинаковые оптимальные стратегии для каждого игрока. В самом деле, хотя в это и трудно поверить, лучше всего представлять себе, что каждый участник выбирает числа от 1 до 5 (возможно, после того, как уже пропущено по 2 кружки). Лучшей стратегией каждого игрока будет выбор чисел от 1 до 5 со следующими частотами:

1: 6.25%, 2: 31.25%, 3: 25.00%, 4: 31.25%, 5: 6.25%.

Когда вы в следующий раз решите выпить пива с приятелем, предложите ему оплачивать каждый заказ по описанному правилу. Разумеется, такая игра подходит и для других случаев.

Наведем порядок

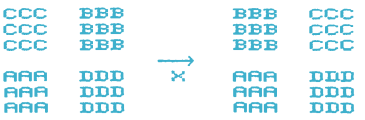
О теории групп

Если вам приходилось крутить кубик Рубика, то вы, вероятно, слышали о *группах* и о том, какую роль они играют в решении этой и других головоломок. Теория групп — одна из важнейших математических дисциплин, и ее вклад в другие области математики, в физику, химию и прочие науки достаточно ощутим. Мы лишь вкратце изложим ряд понятий, относящихся к теории групп.

Процесс переупорядочения элементов, например, вращение различных граней кубика, назовем *перестановкой*. Для иллюстрации этого понятия возьмем четыре квадрата, помеченных различным образом, и разместим их по два в порядке, представленном ниже. Для удобства квадраты были помечены буквами A, B, C и D, но буквы поменялись местами.

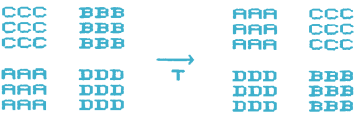
```
CCC   BBB
CCC   BBB
CCC   BBB

AAA   DDD
AAA   DDD
AAA   DDD
```

Перестановка квадратов приводит к их переупорядочению: квадраты меняют свои места. Например, процесс, при котором два верхних квадрата меняются местами, а два нижних остаются на тех же местах, будем называть перестановкой X. Какое действие оказывает эта перестановка на наши квадраты, показано ниже:



Рассмотрим другую перестановку, которая перемещает все квадраты по часовой стрелке. Квадрат в верхнем левом углу перемещается в верхний правый угол, верхний правый квадрат становится нижним правым и т. д. Такую перестановку обозначим буквой T. Ниже показа-

но ее действие на наши квадраты:



Подобные перестановки четырех квадратов весьма наглядно продемонстрирует компьютер, если вы воспользуетесь программой ЧЕТЫРЕ КВАДРАТА. При исполнении программы нажмите клавишу E, чтобы провести эксперимент. Клавиши X или T позволяют увидеть действие соответствующей перестановки на четыре квадрата.

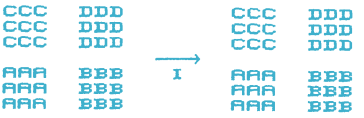
Две или более перестановок можно *комбинировать*, просто производя их одну за другой. Например, вы могли бы выполнить перестановку X, а за нею T:

Любая комбинация перестановок — тоже перестановка. Комбинация перестановки X с последующей перестановкой T записывается XT:

Заметим, что действие на наши четыре квадрата перестановки T с последующей перестановкой X отличается от действия на них перестановки XT. Иначе говоря, XT и TX — различные перестановки. Записывается это как XT ≠ TX и читается: «XT не равно TX».

Комбинация некоторой перестановки с самой собой также дает перестановку, например TT. Произведя перестановку X дважды (т. е. сделав перестановку XX), придем к исходному расположению квадратов. Точно так же, повторив 4 раза перестановку T, получим в результате исходное расположение, как если бы ничего не делалось. Попробуйте проделать это при помощи программы ЧЕТЫРЕ КВАДРАТА.

Перестановка, в результате которой ничего не меняется, называется *тождественной* перестановкой и обозначается буквой I. Может, это покажется странным называть то, что ничего не меняет, перестановкой и даже отводить для нее особый символ, но не менее странно использовать символ 0 для обозначения нуля (пустоты). Поскольку перестановки XX и TTTT производят то же действие, что и I, можно записать XX = I и TTTT = I. Заметим, что комбинация I с некой перестановкой есть сама эта перестановка. Например, XI = X, TIX = TX и т. п.



Предположим, что мы начинаем с некого расположения квадратов и производим перестановку, которая приводит к другому их расположению, не совпадающему с исходным. Повторяя эту перестановку достаточно часто, мы можем случайно вернуться к исходному расположению. Наименьшее число раз, которое необходимо повторить данную перестановку, чтобы она стала тождественной, называется ее *порядком*. Так, перестановка X имеет порядок 2, перестановка T — порядок 4. А каков порядок перестановки XT? Сколько раз нужно выполнить XT, чтобы получить исходное расположение квадратов? Попытайтесь это выяснить, и вы получите 3 — ответ, который на первый взгляд может показаться странным.

Если вы поэкспериментируете с программой ЧЕТЫРЕ КВАДРАТА, то довольно быстро обнаружите, что имеется изрядное количество различных перестановок из четырех квадратов. Сколько разных перестановок вы можете найти? Будьте начеку: очень простая по существу перестановка внешне может выглядеть чрезвычайно сложной. Взгляните, например, на следующее:

XTXTTTXTTTXTT.

То, что эта комбинация *выглядит* сложной, еще не означает, что она

таковой и является. И, действительно, эта перестановка не что иное, как просто X — можете сами в этом убедиться, призвав на помощь ЧЕТЫРЕ КВАДРАТА.

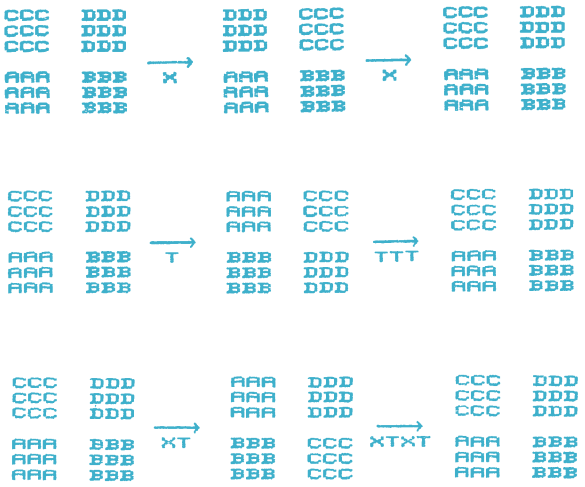
Несмотря на то, что различных выражений для перестановок в виде комбинаций X и T великое множество, фактически имеется 24 *различные* перестановки. Все они перечислены здесь:

I	X	XT	XTT	XTX	XTTX
T	TX	TXT	TXTT	TXTX	TXTTX
TT	TTX	TTXT	TTXTT	TTXTX	TTXTTX
TTT	TTTX	TTTXT	TTTXTT	TTTXTX	TTTXTTX

Разумеется, есть и другие способы представления этих перестановок. Например, вместо TTTXTTX можно написать XTTXT. Проверьте это самостоятельно при помощи программы ЧЕТЫРЕ КВАДРАТА.

Такой набор из двадцати четырех перестановок назовем *группой перестановок из четырех элементов* — это великолепный образчик *группы*.

Группа обладает тем важным свойством, что каждая перестановка в группе может быть сопоставлена другой (называемой *обратной*), которая нейтрализует действие первой перестановки. Например, обратной для перестановки X служит сама X, а обратной перестановкой для T будет такая, которая перемещает четыре квадрата в направлении против часовой стрелки. (Обратной к T будет также TTT.) Обратная к данной перестановке обычно обозначается той же буквой, но с верхним индексом -1: X^{-1}, T^{-1}, и т. д.



Получить обратную для комбинации перестановок очень просто. Например, чтобы обратить действие перестановки XTT, мы просто обратим действие каждой перестановки и применим их в обратном порядке:

$$(XTT)^{-1} = T^{-1}T^{-1}X^{-1}.$$

Как бы ни была сложна комбинация перестановок, ее всегда можно обратить по этому правилу.

Мы уже видели, что различный порядок, в котором записаны две перестановки, приводит к разным результатам. Например, перестановки XT и TX различаются. Будем говорить, что X и T не перестановочны, или что они *не коммутируют* друг с другом. Записываем это свойство как XT ≠ TX и читаем: «XT не равно TX». Если порядок, в котором мы комбинируем две перестановки, не имеет значения, говорим, что перестановки коммутируют друг с другом. Будем говорить, что группа перестановок из четырех элементов *некоммутативна*, так как перестановки здесь не коммутируют друг с другом.

Программа ЧЕТЫРЕ КВАДРАТА предлагает вам также и простую *головоломку*. Для этого при исполнении программы надо нажать P. Компьютер перемешает квадраты случайным образом. Ваша задача получить на экране четыре квадрата в правильном порядке. «Правильный порядок» означает, что квадрат A находится в левом верхнем углу, квадрат B — в верхнем правом, C — в нижнем правом и D — в нижнем левом углу. Разрешается использовать только перестановки X и T, но, разумеется, в любом порядке и сколько угодно раз. Прекрасная зарядка для ума. Потренируйтесь и постарайтесь добиться результата за 6 попыток (или меньше).

В программе ЧЕТЫРЕ КВАДРАТА используется цвет. Если на вашем компьютере такая возможность имеется, то используйте соответствующие коды, в противном случае удалите из программы все команды, где встречаются RED$, BLUE$, и т. п. Так, например, для компьютера ZX 81 нужно удалить все команды с упоминанием кодов цвета и, кроме того, заменить строки 170 и 180 следующими:

```
170 DIM A$(4,3)
180 DIM C$(4,3)
```

На некоторых марках микрокомпьютеров (например, семейства Sinclair) имена текстовых переменных должны состоять только из *одной* буквы, за которой следует знак $. Если такой компьютер дает изображение в цвете (это может быть ZX Spectrum), то вам придется придумать обозначения для RED$, GREEN$, BLUE$ и YELLOW$. Не сле-

дует пользоваться начальными заглавными буквами R$, G$, B$ и Y$, так как они уже заняты в программе для других целей. Предложите еще какие-нибудь буквы. Имеется и иной способ. Например, для машины ZX Spectrum можно заменить строки 130—220 на следующие:

```
170 DIM A$(4,5)
180 DIM C$(4,5)
190 LET A$(1)=CHR$(16)+CHR$(2)+"AAA"
200 LET A$(2)=CHR$(16)+CHR$(4)+"BBB"
210 LET A$(3)=CHR$(16)+CHR$(1)+"CCC"
220 LET A$(4)=CHR$(16)+CHR$(6)+"DDD"
```

```
10 REM                    ********************
20 REM                    *                  *
30 REM                    * ЧЕТЫРЕ КВАДРАТА *
40 REM                    *                  *
50 REM                    ********************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS$=CHR$(147) : REM КОД ДЛЯ ОЧИСТКИ ЭРАНА
120 LET HM$=CHR$(19) : REM КОД ДЛЯ ИСХОДНОГО ПОЛОЖЕНИЯ
130 LET RED$=CHR$(28)
140 LET GREEN$=CHR$(30)
150 LET BLUE$=CHR$(31)
160 LET YELLOW$=CHR$(158)
170 DIM A$(4)
180 DIM C$(4)
190 LET A$(1)=RED$+"AAA"
200 LET A$(2)=GREEN$+"BBB"
210 LET A$(3)=BLUE$+"CCC"
220 LET A$(4)=YELLOW$+"DDD"
230 FOR I=1 TO 4
240 LET C$(I)=A$(I)
250 NEXT I
300 REM XXXXXXXXXXX ЭКСПЕРИМЕНТ ИЛИ ГОЛОВОЛОМКА XXXXXXXXXXXXXXXXXXXXXXXX
310 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
320 PRINT "     ЧЕТЫРЕ КВАДРАТА"
330 PRINT
340 LET T=0
350 PRINT "ДЛЯ ЭКСПЕРИМЕНТА НАЖМИТЕ E"
360 PRINT "ДЛЯ ГОЛОВОЛОМКИ НАЖМИТЕ P"
370 GET G$ : REM LET G$=INKEY$
380 IF G$<>"E" AND G$<>"P" THEN GOTO 370
390 IF G$="E" THEN LET T=1
400 REM XXXXXXXXXXXXXXX ПЕРВОНАЧАЛЬНЫЙ ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXXX
410 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
420 PRINT "     ЧЕТЫРЕ КВАДРАТА"
430 IF T=0 THEN GOSUB 1210
440 GOSUB 1010
450 PRINT
460 PRINT BLUE$
470 PRINT " X ОБМЕНИВАЕТ ДВА ВЕРХНИХ"
480 PRINT
```

Если на вашем компьютере есть цветное изображение, укажите соответствующие коды; в противном случае удалите RED$, BLUE$ и т. д.

В массиве A$(I) содержится информация, выводимая на экран. Массив C$(I) используется для проверки правильного расположения квадратов.

Значение T указывает, что́ это: эксперимент (T = 1) или головоломка (T = 0).

```
490 PRINT " T ВРАЩАЕТ ПО ЧАСОВОЙ"
500 PRINT
510 PRINT " ДЛЯ ОКОНЧАНИЯ НАЖМИТЕ *"
600 REM XXXXXXXXXXXXX ВАШИ ДЕЙСТВИЯ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
610 GET G$ : REM LET G$= INKEY$
620 IF G$="" THEN GOTO 610
630 IF G$="X" THEN GOSUB 1310
640 IF G$="T" THEN GOSUB 1410
650 IF G$="*" THEN GOTO 820
660 GOSUB 1010
670 IF T=1 THEN GOTO 610
700 REM XXXXXXXXXXX ПРОВЕРКА РАСПОЛОЖЕНИЯ КВАДРАТОВ XXXXXXXXXXXXXXXXXXX
710 LET K=1
720 IF A$(K)()C$(K) THEN GOTO 610
730 LET K=K+1
740 IF K<5 THEN GOTO 720
800 REM XXXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXXXXX
810 GOSUB 1610 : REM ДЛЯ ЗВУКОВЫХ ЭФФЕКТОВ
820 FOR I=1 TO 5
830 PRINT
840 NEXT I
850 PRINT BLUE$
860 PRINT "  ПОВТОРИТЬ? Y ИЛИ N"
870 GET G$ : REM LET G$=INKEY$
880 IF G$()"Y" AND G$()"N" THEN GOTO 870
890 IF G$="Y" THEN GOTO 410
900 END : REM STOP
1000 REM XXXXXXXXXXXXXXXXXXXX ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1010 PRINT HM$. : REM PRINT AT 0,0
1020 PRINT
1030 PRINT
1040 FOR J=1 TO 3
1050 PRINT TAB(7);A$(1);" ";A$(2)
1060 NEXT J
1070 PRINT
1080 PRINT
1090 FOR J=1 TO 3
1100 PRINT TAB(7);A$(4);" ";A$(3)
1110 NEXT J
1120 RETURN
1200 REM XXXXXXXXXXXXXXXXXX ПОДПРОГРАММА ПЕРЕМЕШИВАНИЯ XXXXXXXXXXXXXXXXX
1210 FOR K=1 TO 25
1220 ON INT(RND(1)*3+1) GOSUB 1310,1410,1510
1225 REM GOSUB 1310+INT(RND*3)*100
1230 NEXT K
1240 RETURN
1300 REM XXXXXXXXXXXXX ПЕРЕСТАНОВКА X - ОБМЕН ДВУХ ВЕРХНИХ XXXXXXXXXX
1310 LET B$=A$(1)
1320 LET A$(1)=A$(2)
1330 LET A$(2)=B$
1340 RETURN
1400 REM XXXXXXXXXXX ПЕРЕСТАНОВКА T - ВРАЩЕНИЕ ПО ЧАСОВОЙ СТРЕЛКЕ XXXXXXX
1410 LET B$=A$(4)
1420 FOR I=4 TO 2 STEP -1
1430 LET A$(I)=A$(I-1)
1440 NEXT I
1450 LET A$(1)=B$
1460 RETURN
1500 REM XXXXXXXXXXXXXXX ДОПОЛНИТЕЛЬНОЕ ПЕРЕМЕШИВАНИЕ XXXXXXXXXXXXXXXXX
1510 LET B$=A$(3)
1520 FOR I=3 TO 2 STEP -1
```

Вы должны нажать либо X, либо E, либо *.

Если решается головоломка, то здесь проверяется, правильно ли расположены квадраты.

На дисплей выводятся 4 квадрата.

Для головоломки делается 25 случайных перестановок.

Подпрограмма, реализующая перестановку X: два верхних квадрата меняются местами.

Подпрограмма, реализующая перестановку T: все квадраты перемещаются на одну позицию по часовой стрелке.

Дополнительные перестановки для перемешивания квадратов.

```
1530 LET A$(I)=A$(I-1)
1540 NEXT I
1550 LET A$(1)=B$
1560 RETURN
```

Фактически это перестановка TTXTTT, которая перемещает три квадрата по часовой стрелке. (Можете указать, какие именно.)

```
1600 REM XXXXXXXXXXXXXXXXXX ЗВУКОВЫЕ ЭФФЕКТЫ XXXXXXXXXXXXXXXXXXXXXXXXXX
1610 POKE 36878,15
1620 FOR L=200 TO 250
1630 POKE 36876,L
1640 FOR I=1 TO 30:NEXT I
1650 NEXT L
1660 POKE 36876,0
1670 POKE 36878,0
1680 RETURN
```

Если желаете, дайте свои звуковые эффекты.

Дополнительные замечания см. в приложении.

```
ЧЕТЫРЕ КВАДРАТА

ДЛЯ ЭКСПЕРИМЕНТА НАЖМИТЕ E

ДЛЯ ГОЛОВОЛОМКИ НАЖМИТЕ P
```

```
ЧЕТЫРЕ КВАДРАТА

   CCC  AAA
   CCC  AAA
   CCC  AAA

   DDD  BBB
   DDD  BBB
   DDD  BBB

X ОБМЕНИВАЕТ ДВА ВЕРХНИХ

T ВРАЩАЕТ ПО ЧАСОВОЙ

ДЛЯ ОКОНЧАНИЯ НАЖМИТЕ *
```

Транспозиции. Перестановка, которая меняет местами только два квадрата, называется *транспозицией*. Перестановка X, например, меняющая местами два верхних квадрата, — транспозиция. Перестановка T не является транспозицией, поскольку она перемещает все четыре квадрата. Однако T представляет собой комбинацию транспозиций. Рассмотрим следующую последовательность операций:

поменять между собой два верхних квадрата,
затем поменять между собой два левых квадрата,
затем поменять между собой два нижних квадрата.

Такая последовательность действий приводит к тому же результату, что и перестановка T. Таким образом, T не что иное, как комбинация транспозиций. На самом деле любая перестановка в любой группе есть комбинация транспозиций, но — не будем углубляться в детали.

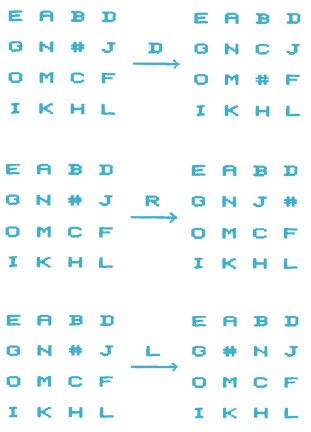
Программа ГОЛОВОЛОМКА 16 КВАДРАТОВ позволяет получить на компьютере интересную игру, основанную на понятии транспозиции. Фактически это известная головоломка «15», придуманная еще в прошлом веке Сэмом Лойдом. Пятнадцать квадратов помечены буквами от A до O, а на шестнадцатом нанесен символ # или (пустышка). Цель игры — расположить квадраты в правильном порядке, как показано ниже:

```
A B C D
E F G H
I J K L
M N O #
```

При нажатии клавиши U, D, R или L можно менять расположение пустышки. Когда нажато U, этот квадрат меняется местами с тем, что находится непосредственно над ним.

```
E A B D          E A # D
G N # J   U      G N B J
O M C F   ---->  O M C F
I K H L          I K H L
```

Точно так же при нажатии D, R или L пустышка меняется местами с квадратом, находящимся непосредственно под ним, справа или слева соответственно. Иначе говоря, нажатие клавиши U, D, R или L осуществляет транспозицию.




```
10 REM                    ****************************
20 REM                    *                          *
30 REM                    * ГОЛОВОЛОМКА 16 КВАДРАТОВ *
40 REM                    *                          *
50 REM                    ****************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ДЛЯ ОЧИСТКИ ЭКРАНА
120 LET HM=CHR$(19) : REM КОД ДЛЯ ИСХОДНОГО ПОЛОЖЕНИЯ
130 LET S=ASC("A") : REM LET S=CODE("A")
140 LET P=4
150 LET Q=4
160 DIM A$(P,Q)
170 FOR I=1 TO P
180 FOR J=1 TO Q
190 LET T=S+Q*(I-1)+(J-1)
200 LET A$(I,J)=CHR$(T)
210 NEXT J
220 NEXT I
230 LET A$(P,Q)="#" : REM LET A$(P,Q)="▒"
240 LET X=P
250 LET Y=Q
260 LET M=P
270 LET N=Q
280 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
290 PRINT "ГОЛОВОЛОМКА 16 КВАДРАТОВ "
300 PRINT
310 PRINT "        ЖДИТЕ "
400 REM XXXXXXXXXXXXXXXXXXXXXX ПЕРЕМЕШИВАНИЕ XXXXXXXXXXXXXXXXXXXXXXXXX
410 FOR K=1 TO 100
420 LET U=RND(1)
430 IF U<0.25 AND M<P THEN LET M=M+1
440 IF U>=0.25 AND U<0.5 AND M>1 THEN LET M=M-1
450 IF U>=0.5 AND U<0.75 AND N<Q THEN LET N=N+1
460 IF U>=0.75 AND N>1 THEN LET N=N-1
470 GOSUB 1210
480 NEXT K
```

«Размер» головоломки можно изменить, воспользовавшись другими значениями P и Q в строках 140 и 150.

Делается 100 случайных перестановок. Если хотите облегчить головоломку, уменьшите это число. Если есть необходимость наблюдать за перемешиванием, вставьте дополнитель-

```
490 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
500 GOSUB 1010
600 REM XXXXXXXXXXXXXXXXXXXXXX ВАШИ ДЕЙСТВИЯ XXXXXXXXXXXXXXXXXXXXXXXXX
610 GET G$ : REM LET G$=INKEY$
620 IF G$="*" THEN GOTO 950
630 IF G$<>"D" AND G$<>"U" AND G$<>"R" AND G$<>"L" THEN GOTO 610
640 IF G$="D" AND M<P THEN LET M=M+1
650 IF G$="U" AND M>1 THEN LET M=M-1
660 IF G$="R" AND N<Q THEN LET N=N+1
670 IF G$="L" AND N>1 THEN LET N=N-1
700 REM XXXXXXXXXXXXXX ОБМЕН И ПРОВЕРКА НА ОКОНЧАНИЕ XXXXXXXXXXXXXXXXXXX
710 REM
720 GOSUB 1210
730 GOSUB 1010
740 LET I=1
750 LET J=1
760 IF I=P AND J=Q THEN GOTO 910
770 LET T=S+Q*(I-1)+(J-1)
780 IF A$(I,J)<>CHR$(T) THEN GOTO 610
790 LET J=J+1
800 IF J>Q THEN LET I=I+1
810 IF J>Q THEN LET J=1
820 IF I<P+1 THEN GOTO 760
900 REM XXXXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXX
910 GOSUB 1010
920 PRINT "      ХОРОШО"
930 PRINT
940 GOSUB 1310
950 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
960 GET G$ : REM LET G$=INKEY$
970 IF G$<>"Y" AND G$<>"N" THEN GOTO 960
980 IF G$="Y" THEN GOTO 280
990 END : REM STOP
1000 REM XXXXXXXXXXXXXXXXXXXX ВЫВОД НА ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXX
1010 PRINT HM$ : REM PRINT AT 0,0
1020 PRINT "ГОЛОВОЛОМКА 16 КВАДРАТОВ"
1030 PRINT " U=ВВЕРХ       R=ВПРАВО"
1040 PRINT
1050 PRINT " D=ВНИЗ         L=ВЛЕВО"
1060 PRINT
1070 PRINT
1080 FOR I=1 TO P
1090 PRINT "      ";
1100 FOR J=1 TO Q
1110 PRINT " ";A$(I,J);
1120 NEXT J
1130 PRINT
1140 PRINT
1150 NEXT I
1160 PRINT
1170 PRINT "  ДЛЯ ОКОНЧАНИЯ НАЖМИТЕ *"
1180 PRINT
1190 RETURN
1200 REM XXXXXXXXXXXXXXXXXXXXXXXXX ОБМЕН XXXXXXXXXXXXXXXXXXXXXXXXX
1210 LET B$=A$(M,N)
1220 LET A$(M,N)=A$(X,Y)
1230 LET A$(X,Y)=B$
1240 LET X=M
1250 LET Y=N
1260 RETURN
```

ную строке 465 с командой GOSUB 1010.

Делается проверка, правильно ли расположены квадраты.

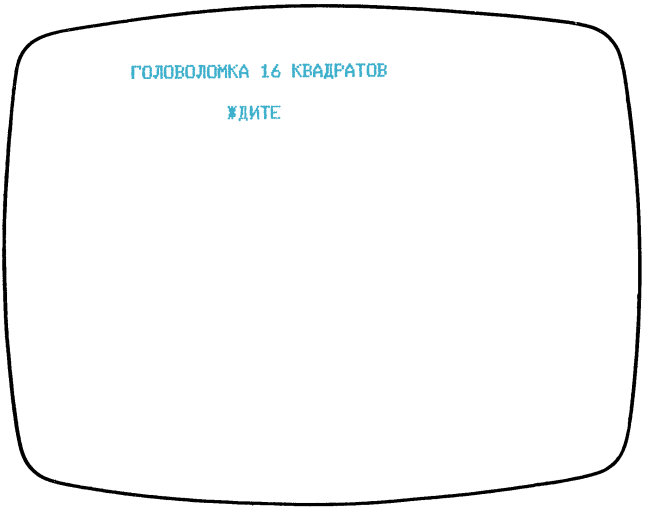
Вывод на дисплей.

Подпрограмма, выполняющая транспозицию.

```
1300 REM XXXXXXXXXXXXXXXXXX ЗВУКОВЫЕ ЭФФЕКТЫ XXXXXXXXXXXXXXXXXXXXX
1310 POKE 36878,15
1320 FOR L=200 TO 250
1330 POKE 36876,L
1340 FOR I=1 TO 30:NEXT I
1350 NEXT L
1360 POKE 36876,0:POKE 36878,0
1370 RETURN
```

Дайте свои звуковые эффекты.

Дополнительные замечания см. в приложении.

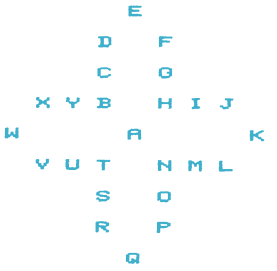


Циклы. Перестановка T «циклически» переставляет четыре квадрата по часовой стрелке. Будем называть T *циклом*, или *4-циклом*, поскольку в данном случае речь идет о четырех элементах. Транспозиция — это частный случай цикла; она циклически переставляет два квадрата, т. е. это 2-цикл. Не всякая перестановка является циклом. Например, вот перестановка XTTX, которая меняет местами два верхних квадрата и два нижних:

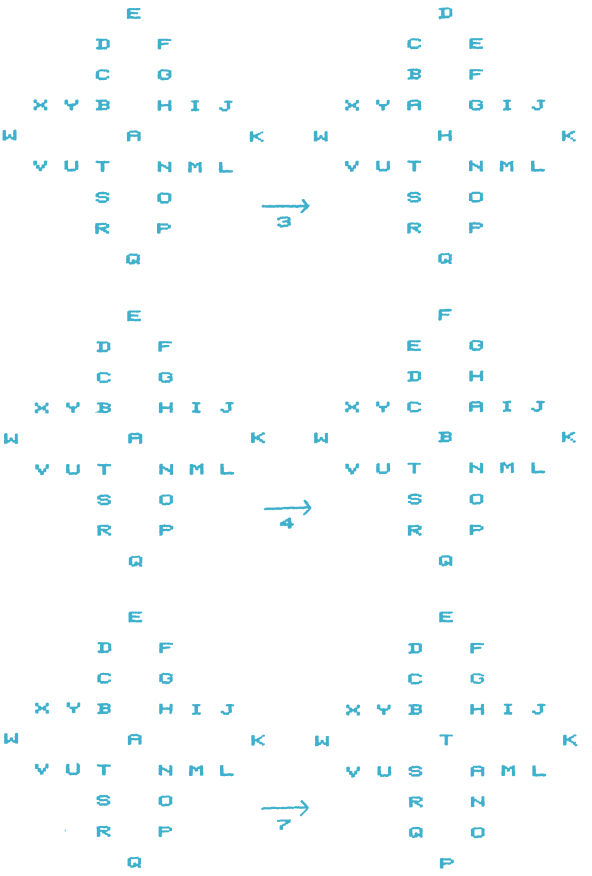
Она состоит из двух (непересекающихся) 2-циклов и, следовательно, циклом быть не может.

Еще одним примером цикла служит перестановка XT; это 3-цикл, потому что она циклически переставляет три квадрата.

Понятие цикла играет в теории групп очень важную роль как в теоретическом, так и в практическом отношении, но в дальнейшие подробности мы здесь вдаваться не будем. Следующая программа ГОЛОВОЛОМКА 25 КВАДРАТОВ основана на понятии цикла. Она представляет собой очень интересную и чрезвычайно сложную головоломку, в которой уже приходится иметь дело с перестановками из 25 квадратов. Эти квадраты расположены в виде конфигурации, представленной ниже, и помечены 25 буквами английского алфавита от A до Y:



Компьютер перемешивает эти квадраты, а вы должны снова переупорядочить их в правильном порядке (как изображено выше). Допускаются только несколько перестановок и комбинации из них. Они перемещают элементы каждого лепестка из 8 квадратов либо по часовой, либо против часовой стрелки. Каждая перестановка представляет собой 8-цикл. Эти циклы можно вызвать нажатием клавиш от 1 до 8. Если, например, нажать 3, то восемь верхних квадратов переместятся по часовой стрелке (см. рисунок ниже). Результаты других перестановок можно увидеть, исполнив программу.



```
10 REM                  ****************************
20 REM                  *                          *
30 REM                  * ГОЛОВОЛОМКА 25 КВАДРАТОВ *
40 REM                  *                          *
50 REM                  ****************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET HM=CHR$(19) : REM КОД ИСХОДНОГО ПОЛОЖЕНИЯ
130 LET I$=CHR$(18)
140 LET O$=CHR$(146)
150 DIM A$(25)
160 LET S=ASC("A")-1 : REM LET S=CODE("A")-1
170 FOR I=1 TO 25
180 LET A$(I)=I$+CHR$(S+I)+O$
190 REM FOR ZX81 USE LET A$(I)=CHR$(128+S+I)
200 NEXT I
```

При помощи I$ включается обратное (или инверсное) изображение, а при помощи O$ оно отключается; см. приложение.

В массиве A$(I) содержится 25 букв латинского алфавита от A до Y.

```
300 REM XXXXXXXXXXXXXXXXX ПЕРВОНАЧАЛЬНЫЙ ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXX
310 PRINT CS$; : REM ОЧИСТИТЬ ЭКРАН
320 PRINT "     25 КВАДРАТОВ"
330 GOSUB 810
340 PRINT
350 PRINT "C=ПО ЧАСОВОЙ  A=ПРОТИВ"
360 PRINT "ДЛЯ ОКОНЧАНИЯ НАЖМИТЕ *"
400 REM XXXXXXXXXXXXXXXXXX ПЕРЕМЕШИВАНИЕ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
410 FOR J=1 TO 25
420 ON INT(8*RND(1)+1)GOSUB 1110,1210,1310,1410,1510,1610,1710,1810
430 REM GOSUB 1110+INT(RND*8)*100
440 NEXT J
450 GOSUB 810
```

Если хотите увидеть, как происходит перемешивание, поменяйте местами строки 440 и 450.

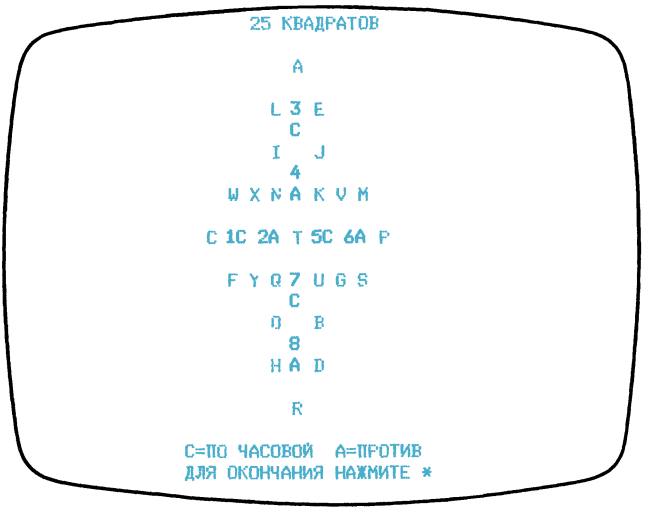
```
500 REM XXXXXXXXXXXXXXXXX ВАШИ ДЕЙСТВИЯ XXXXXXXXXXXXXXXXXXXXXXXXXXX
510 GET G$ : REM LET G$=INKEY$
520 IF G$="*" THEN GOTO 670
530 IF G$("1" OR G$)"B" THEN GOTO 510
540 ON VAL(G$) GOSUB 1110,1210,1310,1410,1510,1610,1710,1810
550 REM GOSUB 1010+100*VAL(G$)
560 GOSUB 810
600 REM XXXXXXXXX ПРОВЕРКА НА ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXX
610 LET J=1
620 IF A$(J)()I$+CHR$(S+J)+O$ THEN GOTO 510
630 REM IF A$(J)()CHR$(128+S+J) THEN GOTO 510
640 LET J=J+1
650 IF J(26 THEN GOTO 620
660 GOSUB 1910
670 PRINT
680 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
690 GET G$ : REM LET G$=INKEY$
700 IF G$()"Y" AND G$()"N" THEN GOTO 690
710 IF G$="Y" THEN GOTO 310
720 END : REM STOP
800 REM XXXXXXXXXXXXXXXXXXX ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
810 PRINT HM$ : REM PRINT AT 0,0
820 PRINT
830 PRINT TAB(10);A$(5)
840 PRINT
850 PRINT TAB(8);A$(4);" 3 ";A$(6)
```

```
860 PRINT TAB(10);"C"
870 PRINT TAB(8);A%(3);"   ";A%(7)
880 PRINT TAB(10);"4"
890 PRINT TAB(4);A%(24);" ";A%(25);" ";A%(2);" A ";A%(8);" ";
900 PRINT A%(9);" ";A%(10)
910 PRINT
920 PRINT TAB(2);A%(23);" 1C 2A ";A%(1);" 5C 6A ";A%(11)
930 PRINT
940 PRINT TAB(4);A%(22);" ";A%(21);" A%(20);" 7 ";A%(14);
950 PRINT " ";A%(13);" ";A%(12)
960 PRINT TAB(10);"C"
970 PRINT TAB(8);A%(19);"   ";A%(15)
980 PRINT TAB(10);"8"
990 PRINT TAB(8);A%(18);" A ";A%(16)
1000 PRINT
1010 PRINT TAB(10);A%(17)
1020 RETURN
1100 REM XXXXXXXXXX ПЕРЕСТАНОВКА 1 - ЛЕВЫЙ ПО ЧАСОВОЙ XXXXXXXXXXXXXXXXX
1110 LET B%=A%(25)
1120 FOR I=25 TO 21 STEP -1
1130 LET A%(I)=A%(I-1)
1140 NEXT I
1150 LET A%(20)=A%(1)
1160 LET A%(1)=A%(2)
1170 LET A%(2)=B%
1180 RETURN
1200 REM XXXXXXXXXXXXX ПЕРЕСТАНОВКА 2 - ЛЕВЫЙ ПРОТИВ  ЧАСОВОЙ XXXXXXXXX
1210 LET B%=A%(20)
1220 FOR I=20 TO 24
1230 LET A%(I)=A%(I+1)
1240 NEXT I
1250 LET A%(25)=A%(2)
1260 LET A%(2)=A%(1)
1270 LET A%(1)=B%
1280 RETURN
1300 REM XXXXXXXXXXXX ПЕРЕСТАНОВКА 3 - ВЕРХНИЙ ПО ЧАСОВОЙ XXXXXXXXXXXXX
1310 LET B%=A%(8)
1320 FOR I=8 TO 2 STEP -1
1330 LET A%(I)=A%(I-1)
1340 NEXT I
1350 LET A%(1)=B%
1360 RETURN
1400 REM XXXXXXXXXXXX ПЕРЕСТАНОВКА 4 - ВЕРХНИЙ ПРОТИВ ЧАСОВОЙ XXXXXXXXX
1410 LET B%=A%(1)
1420 FOR I=1 TO 7
1430 LET A%(I)=A%(I+1)
1440 NEXT I
1450 LET A%(8)=B%
1460 RETURN
1500 REM XXXXXXXXXXX ПЕРЕСТАНОВКА 5 - ПРАВЫЙ ПО ЧАСОВОЙ XXXXXXXXXXXXXXXX
1510 LET B%=A%(14)
1520 FOR I=14 TO 9 STEP -1
1530 LET A%(I)=A%(I-1)
1540 NEXT I
1550 LET A%(8)=A%(1)
1560 LET A%(1)=B%
1570 RETURN
1600 REM XXXXXXXXXXX ПЕРЕСТАНОВКА 6 - ПРАВЫЙ ПРОТИВ ЧАСОВОЙ XXXXXXXXXXXX
1610 LET B%=A%(8)
```

```
1620 FOR I=8 TO 13
1630 LET A$(I)=A$(I+1)
1640 NEXT I
1650 LET A$(14)=A$(1)
1660 LET A$(1)=B$
1670 RETURN
1700 REM XXXXXXXXXXXXX ПЕРЕСТАНОВКА 7 - НИЖНИЙ ПО ЧАСОВОЙ XXXXXXXXXXXXX
1710 LET B$=A$(20)
1720 FOR I=20 TO 15 STEP -1
1730 LET A$(I)=A$(I-1)
1740 NEXT I
1750 LET A$(14)=A$(1)
1760 LET A$(1)=B$
1770 RETURN
1800 REM XXXXXXXXXXXX ПЕРЕСТАНОВКА 8 - НИЖНИЙ ПРОТИВ ЧАСОВОЙ XXXXXXXXXX
1810 LET B$=A$(14)
1820 FOR I=14 TO 19
1830 LET A$(I)=A$(I+1)
1840 NEXT I
1850 LET A$(20)=A$(1)
1860 LET A$(1)=B$
1870 RETURN
1900 REM XXXXXXXXXXXXXXXXXXXXX ЗВУКОВЫЕ ЭФФЕКТЫ XXXXXXXXXXXXXXXXXXXXXXX
1910 POKE 36878,15
1920 FOR L=200 TO 250
1930 POKE 36876,L
1940 FOR I=1 TO 30
1950 NEXT I
1960 NEXT L
1970 POKE 36876,0
1980 POKE 36878,0
1990 RETURN
```

Если такая возможность есть, дайте свои звуковые эффекты.

Общие замечания можно найти в приложении.



Более подробно с различными понятиями теории групп можно ознакомиться по книге J. Ewing, C. Kosniowski. Puzzle it Out: Cubes, Groups and Puzzles. (См. также Курош А. Г. Теория групп. — М.: Наука, 1967 (*Перев.*).)

О теории массового обслуживания

Сколько раз, придя в магазин или на почту и увидев там несколько очередей, вы становились в одну из них и обнаруживали, что именно эта очередь совершенно не сдвигается с места из-за какого-то недотепы, стоящего впереди? Очереди нас раздражают: мы, несомненно, тратим страшно много времени на ожидание обслуживания.

Один из способов избежать нескольких очередей состоит в том, чтобы иметь только одну очередь, из которой клиенты поступают на обслуживание по принципу «первым пришел — первым обслужен». Эта система сейчас принята во многих банковских и почтовых отделениях.

Предположим, что мы имеем дело именно с такой ситуацией, скажем, в банке с одним или более клерков, обслуживающих очередь клиентов. Эта очередь единая для всех клиентов, и обслуживаются они по принципу «первым пришел — первым обслужен». Как правило, клиенты приходят не через строго определенные интервалы. Если в среднем за час придут 10 клиентов, или 1 клиент за 6 минут, то это, разумеется, не означает, что каждые 6 минут обязательно появляется по одному клиенту. Разность между моментами прихода двух последовательных клиентов (между моментами поступления) называется *интервалом между поступлениями*. Это — случайная величина, не зависящая от длины очереди.

Для моделирования моментов поступления клиентов можно использовать персональный компьютер. Метод, при помощи которого генерируются эти случайные величины, называется методом *Монте-Карло*. Продемонстрируем его на следующем простом примере. Если известно, что каждые 6 минут в среднем приходит один клиент, то в течение любого заданного одноминутного интервала клиент может появиться приблизительно с вероятностью 1/6. Таким образом, каждую минуту наш микрокомпьютер выбирает случайное число между 0 и 1 и проверяет, меньше ли оно 1/6. Если да, то клиент пришел, в противном случае — клиента нет. Все, разумеется, происходит значительно быстрее и слова «каждую минуту» не следует понимать буквально.

Вот как выглядят несколько первых минут нашего процесса:

Время (минуты)	Случайное число RND	Меньше 1/6?	Момент поступления клиента
1	0.21273	Нет	—
2	0.10601	Да	2
3	0.31993	Нет	—
4	0.65851	Нет	—
5	0.66965	Нет	—
6	0.97381	Нет	—
7	0.59476	Нет	—
8	0.50985	Нет	—
9	0.42101	Нет	—
10	0.52048	Нет	—
11	0.09276	Да	11
.	.	.	.
.	.	.	.
.	.	.	.

Программа ПОСТУПЛЕНИЕ КЛИЕНТОВ I моделирует моменты поступления клиентов в течение 60 мин. Предполагается, что в среднем каждые 6 мин приходит новый клиент, иначе говоря, скорость (CAR), с которой появляются клиенты, равна 1/6 клиента в минуту.

```
10 REM                    ***************************
20 REM                    *                         *
30 REM                    * ПОСТУПЛЕНИЕ КЛИЕНТОВ I *
40 REM                    *                         *
50 REM                    ***************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXX
110 LET CAR=1/6 : REM СКОРОСТЬ ПОСТУПЛЕНИЯ
120 REM СМ. ПРЕДОСТЕРЕЖЕНИЕ В ТЕКСТЕ
130 LET T=60 : REM ОБЩИЙ ВРЕМЕННОЙ ИНТЕРВАЛ
140 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
150 PRINT " ПОСТУПЛЕНИЕ КЛИЕНТОВ I"
200 REM XXXXXXXXXXXXXXXXXXX СЧЕТ И ПЕЧАТЬ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
210 PRINT
220 PRINT "  СКОРОСТЬ ПОСТУПЛЕНИЯ"
230 PRINT CAR;"В МИНУТУ"
240 PRINT
250 PRINT "  МОМЕНТ ПОСТУПЛЕНИЯ "
260 PRINT " В МИНУТАХ ОТ НАЧАЛА "
270 PRINT
280 LET K=0
290 FOR I=1 TO T
300 LET X=RND(1)
310 IF X<CAR THEN PRINT "        ";I
320 IF X<CAR THEN LET K=K+1
330 NEXT I
```

```
340 PRINT
350 PRINT "ОБЩЕЕ ЧИСЛО КЛИЕНТОВ ";K
360 PRINT
370 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
380 GET G$ : REM LET G$=INKEY$
390 IF G$<>"Y" AND G$<>"N" THEN GOTO 380
400 IF G$="Y" THEN GOTO 210
```

Общие замечания можно найти в приложении.

```
ПОСТУПЛЕНИЕ КЛИЕНТОВ I

 СКОРОСТЬ ПОСТУПЛЕНИЯ
.166666667 В МИНУТУ

 МОМЕНТ ПОСТУПЛЕНИЯ
В МИНУТАХ ОТ НАЧАЛА

        7
        12
        14
        25
        30
        33
        35
        36
        37

ОБЩЕЕ ЧИСЛО КЛИЕНТОВ 10

 ПОВТОРИТЬ?  Y ИЛИ N
```

Имея дело с такой программой, как ПОСТУПЛЕНИЕ КЛИЕНТОВ I, следует быть очень внимательным. Как быть, если в среднем поступает 1 клиент в минуту? Просто заменить значение CAR в строке 110 числом 1 нельзя. Если только, конечно, вы не хотите рассмотреть ситуацию, когда каждую минуту приходит ровно по одному клиенту, которая малоправдоподобна (именно этим отчасти объясняется употребление слова «приблизительно» при описании метода Монте-Карло). Чтобы выйти из этого затруднения, следует уменьшить временной интервал и рассмотреть, например, 1/10 мин. Теперь вероятность того, что за одну десятую минуты поступит один клиент, равна приблизительно 1/10. Вообще говоря, нужно брать как можно меньший интервал. Но чем меньше интервал, тем больше вычислений будет проделано. В программе ПОСТУПЛЕНИЕ КЛИЕНТОВ II эти соображения учтены.

Временной интервал делится на 10 до тех пор, пока вероятность поступления клиента за этот интервал не станет меньше или равна 1/10.

```
10 REM                    ***************************
20 REM                    *                         *
30 REM                    * ПОСТУПЛЕНИЕ КЛИЕНТОВ II *
40 REM                    *                         *
50 REM                    ***************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXXX
110 REM CAR=1/6 : REM СКОРОСТЬ ПОСТУПЛЕНИЯ
120 LET R=1
130 LET R=R/10
140 LET S= CAR*R
150 IF S)0.1 THEN GOTO 130
160 LET T=60 : REM ОБЩИЙ ВРЕМЕННОЙ ИНТЕРВАЛ
170 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
180 PRINT " ПОСТУПЛЕНИЕ КЛИЕНТОВ II"
200 REM XXXXXXXXXXXXXXXXXXXXX СЧЕТ И ПЕЧАТЬ XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
210 PRINT
220 PRINT "  СКОРОСТЬ ПОСТУПЛЕНИЯ"
230 PRINT CAR;"В МИНУТУ"
240 PRINT
250 PRINT "   МОМЕНТ ПОСТУПЛЕНИЯ"
260 PRINT "  В МИНУТАХ ОТ НАЧАЛА"
270 PRINT
280 LET K=0
290 FOR I=1 TO T STEP R
300 LET X=RND(1)
310 IF X<S THEN PRINT "          ";INT(I*10)/10
320 IF X<S THEN LET K=K+1
330 NEXT I
340 PRINT
350 PRINT "ОБЩЕЕ ЧИСЛО КЛИЕНТОВ";K
360 PRINT
370 PRINT "  ПОВТОРИТЬ?  Y ИЛИ N"
380 GET G$ : REM LET G$=INKEY$
390 IF G$<>"Y" AND G$<>"N" THEN GOTO 380
400 IF G$="Y" THEN  GOTO 210
```

Временной интервал делится на 10 до тех пор, пока вероятность поступления клиента за этот интервал не станет меньше или равна 1/10.

Общие замечания можно найти в приложении.

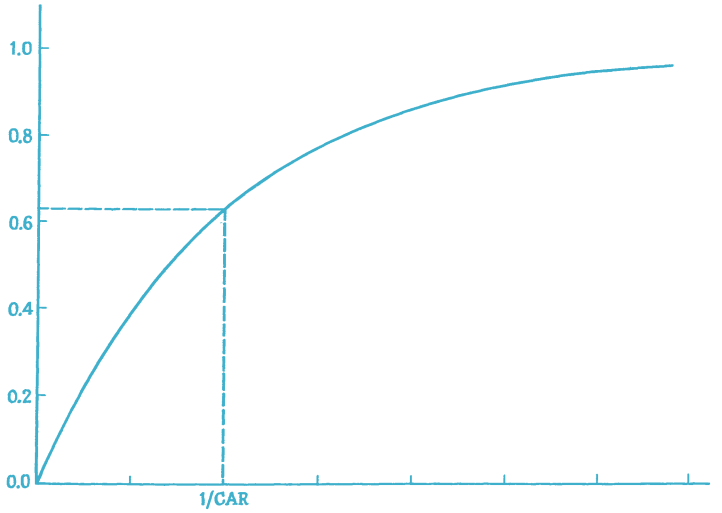
```
ПОСТУПЛЕНИЕ КЛИЕНТОВ II

 СКОРОСТЬ ПОСТУПЛЕНИЯ
.166666667 В МИНУТУ

  МОМЕНТ ПОСТУПЛЕНИЯ
 В МИНУТАХ ОТ НАЧАЛА

          2.8
          15.5
          20.6
          26.3
          27.7
          34.5
          36.1
          45.9
          55.7

ОБЩЕЕ ЧИСЛО КЛИЕНТОВ  9

 ПОВТОРИТЬ?  Y ИЛИ N
```

Существует и другой подход, при котором скорость вычислений увеличивается. Для определения интервалов между поступлениями клиентов мы используем случайные числа. Предположим, что клиенты появляются со скоростью CAR. Это число не может быть целым! Оказывается, вероятность того, что интервал между поступлениями меньше или равен Y, задается формулой 1 − EXP(−CAR*Y).

Например, имеется вероятность 0.632 того, что клиент придет в течение интервала 1/CAR после поступления предыдущего клиента (т. е. в 63.2% случаев). Подразумевается, что за интервал 1/CAR приходит один клиент в среднем, а не регулярно. Ниже представлен график 1 − EXP(−CAR*Y) для разных значений Y.

Из графика видно, что вероятность того, что интервал между поступлениями нулевой, равна нулю. Вероятность же того, что интервал между поступлениями меньше некоего очень большого числа, равна приблизительно единице.

Выписанное выше выражение называется *вероятностной функцией распределения* или *функцией накопленных вероятностей*. Точного обоснования этой формуле здесь дано не будет, но на опыте можно убедиться, что она очень хорошо согласуется с наблюдениями. Это основывается на так называемом *пуассоновском* распределении для моментов поступления клиентов. В двух словах это означает, что клиенты появляются случайным образом, но с фиксированной вероятностью поступления в неком малом интервале. Должно быть, вы раньше слышали о Пуассоне. Французский математик и физик Симеон Дени Пуас-



сон (1781—1840) известен своими открытиями в механике и электростатике.

Ваш компьютер может моделировать моменты поступления клиентов при помощи упомянутой выше формулы. Прежде всего обратите внимание, что 1 − EXP(−CAR*Y) — число, заключенное между 0 и 1. Момент поступления произвольного клиента можно получить, выбрав случайное число RND (между 0 и 1), вычислив 1 — RND и затем сравнив результат с 1 − EXP(−CAR*Y). Простые алгебраические преобразования приводят к следующим уравнениям:

$$\begin{aligned} 1 - \mathrm{EXP}(-\mathrm{CAR}*\mathrm{Y}) &= 1 - \mathrm{RND}, \\ \mathrm{EXP}(-\mathrm{CAR}*\mathrm{Y}) &= \mathrm{RND}, \\ -\mathrm{CAR}*\mathrm{Y} &= \mathrm{LN}(\mathrm{RND}), \\ \mathrm{Y} &= -\mathrm{LN}(\mathrm{RND})/\mathrm{CAR}. \end{aligned}$$

Таким образом, интервал между поступлениями Y вычисляется через случайное число RND по следующей формуле:

$$\mathrm{Y} = -\mathrm{LN}(\mathrm{RND})/\mathrm{CAR}.$$

Предположим, например, что CAR = 1/6. Ниже показано, как можно получить несколько первых моментов поступления клиентов:

Случайное число RND	Интервал между поступлениями −LN(RND)/CAR	Момент поступления клиента
0.81785	1.21	1.21
0.18753	10.04	11.25
0.54975	3.59	14.84
0.38481	5.73	20.57
0.05226	17.71	38.28
0.23964	8.57	46.85
0.69152	2.21	49.06
0.56127	3.47	52.53
0.79813	1.35	53.88
0.81238	1.25	55.13
.	.	.
.	.	.
.	.	.

В программе ПОСТУПЛЕНИЕ КЛИЕНТОВ III по только что упомянутой формуле моделируется поступление клиентов за период времени, равный 60 мин. Скорость поступления равна 1/6. При желании вы можете изменять длину периода и скорость поступления.

```
10 REM                  ****************************
20 REM                  *                          *
30 REM                  * ПОСТУПЛЕНИЕ КЛИЕНТОВ III *
40 REM                  *                          *
50 REM                  ****************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CAR=1/6 : REM СКОРОСТЬ ПОСТУПЛЕНИЯ
130 LET T=60 : REM ОБЩИЙ ВРЕМЕННОЙ ИНТЕРВАЛ
140 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
150 PRINT "ПОСТУПЛЕНИЕ КЛИЕНТОВ III"
200 REM XXXXXXXXXXXXXXXXXXXXXX СЧЕТ И ПЕЧАТЬ XXXXXXXXXXXXXXXXXXXXXXXXXXXX
210 PRINT
220 PRINT "  СКОРОСТЬ ПОСТУПЛЕНИЯ"
230 PRINT CAR;"В МИНУТУ"
240 PRINT
250 PRINT "   МОМЕНТ ПОСТУПЛЕНИЯ"
260 PRINT "  В МИНУТАХ ОТ НАЧАЛА"
270 PRINT
280 LET K=0
290 LET Z=0
300 LET X=RND(1)
310 LET Y=LOG(X)/CAR : REM ИНТЕРВАЛ МЕЖДУ ПОСТУПЛЕНИЯМИ
320 LET Y=INT(100*Y+0.5)/100
330 LET Z=Z+Y
340 IF Z>T THEN GOTO 380
350 PRINT "        ";Z
360 LET K=K+1
370 GOTO 300
380 PRINT
390 PRINT "ОБЩЕЕ ЧИСЛО КЛИЕНТОВ ";K
400 PRINT
410 PRINT " ПОВТОРИТЬ?  Y ИЛИ N"
420 GET G$ : REM LET G$=INKEY$
430 IF G$<>"Y" AND G$<>"N" THEN GOTO 420
440 IF G$="Y" THEN GOTO 210
```

Общие замечания можно найти в приложении.

```
ПОСТУПЛЕНИЕ КЛИЕНТОВ III

 СКОРОСТЬ ПОСТУПЛЕНИЯ
.166666667 В МИНУТУ

  МОМЕНТ ПОСТУПЛЕНИЯ
 В МИНУТАХ ОТ НАЧАЛА

        18.95
        19.63
        29.43
        33.78
        34.42
        40.3
        57
        57.94
        58.52
        58.72

ОБЩЕЕ ЧИСЛО КЛИЕНТОВ  10

 ПОВТОРИТЬ?  Y ИЛИ N
```

Заметим, что для программы ПОСТУПЛЕНИЕ КЛИЕНТОВ I нужно 60 случайных чисел, тогда как для ПОСТУПЛЕНИЯ КЛИЕНТОВ II — около 600. А вот программа ПОСТУПЛЕНИЕ КЛИЕНТОВ III обходится всего 10 случайными числами.

Клиенты не появляются регулярно через заданный интервал, равно как и клерки не обслуживают их за определенный отрезок времени. Время обслуживания для каждого клиента свое. Эти времена задаются формулой, аналогичной выражению для интервалов между поступлениями клиентов. В самом деле, если CSR есть скорость обслуживания клиента одним клерком (среднее число клиентов, которое может обслужить один клерк в единицу времени), то вероятность того, что клиент будет обслужен за время Y, есть 1 − EXP(−CSR*Y). Специалисты по теории массового обслуживания говорят в таких случаях, что время обслуживания имеет *показательное* (*экспоненциальное*) *распределение*. Смоделировать время обслуживания клиентов можно точно так же, как и моменты между их поступлениями, заменив в соответствующих программах CAR на CSR.

Можно объединить оба процесса. Промоделируем такую ситуацию, когда имеется одна очередь и один или несколько клерков. Подобная модель поможет вам прекрасно разобраться в любой аналогичной задаче из реальной жизни при минимальных затратах и без ненужного риска, возникающего при работе с настоящими клиентами и клерками.

Программа ОБСЛУЖИВАНИЕ В БАНКЕ моделирует одну неделю работы банка, операции в котором производят один или несколько клерков. Рабочая неделя состоит из 6 дней по 360 мин (6 часов) в каждом. Банк работает с 10 до 16 часов. Имеется одна общая очередь клиентов, которые обслуживаются по правилу «первым пришел — первым обслужен». Клиенты появляются со скоростью 24 клиента в час, а на обслуживание каждого у клерка уходит в среднем 10 мин. При желании эти скорости можно изменить, если, например, вы хотите смоделировать другое деловое учреждение.

Прежде чем приступить к исполнению программы, необходимо задать количество клерков на каждый день. Ваш банк невелик и в нем не должна образовываться слишком длинная очередь. Нужно задать также необходимую максимальную длину очереди (наибольшее количество клиентов, стоящих в очереди, исключая тех, что находятся на обслуживании). Затем вы будете получать поминутный (фактически значительно быстрее) отчет о работе банка. В конце каждого дня подводится итог за день и повторяются итоги всех предыдущих дней. Эта информация содержит число клерков, максимальную длину очереди, число потерянных клиентов из-за того, что очередь была переполнена, и среднее время из 360 мин, которое клерк был занят обслуживанием.

Последняя цифра дает информацию о загруженности клерка работой.

Для удобства предполагается, что у клерков нет перерывов на обед и прочие нужды. Кроме того, по прошествии 360 мин все клиенты в очереди за исключением тех, кого уже начали обслуживать, считаются потерянными. Их число включается в общее число потерянных клиентов. Заметим, что возможна такая ситуация, когда клерк оказывается в среднем занят более 360 мин. Это происходит тогда, когда он был непрерывно занят и продолжал обслуживать какого-нибудь клиента непосредственно после закрытия банка.

Поэкспериментируйте с программой, чтобы выяснить, насколько мал ваш банк и сколько вам нужно клерков.

```
10 REM                  ************************
20 REM                  *                      *
30 REM                  * ОБСЛУЖИВАНИЕ В БАНКЕ *
40 REM                  *                      *
50 REM                  ************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS$=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET HM$=CHR$(19) : REM КОД ИСХОДНОГО ПОЛОЖЕНИЯ
130 LET CAR=0.4 : REM СКОРОСТЬ ПОСТУПЛЕНИЯ
140 LET CSR=0.1 : REM СКОРОСТЬ ОБСЛУЖИВАНИЯ
150 DIM A(10)
160 DIM S(6)
170 DIM T(6)
180 DIM L(6)
190 DIM M(6)
```

Значения CAR и CSR можно изменить

В массиве A(I) хранятся моменты окончания обслуживания клиентов каждым клерком. Количество клерков на каждый из 6 дней заносится в массив S(I). Массив T(I) — суммарные времена занятости клерков на каждый из 6 дней, а L(I) — количество потерянных клиентов на каждый день. Наконец, в массиве M(I) хранится максимальное количество мест ожидания в очереди.

```
200 LET DAY=1
210 LET B$=CHR$(31) : REM ГОЛУБОЙ
220 LET R$=CHR$(28) : REM КРАСНЫЙ
230 LET M$=CHR$(156) : REM ФИОЛЕТОВЫЙ
240 LET G$=CHR$(30) : REM ЗЕЛЕНЫЙ
250 LET C$=CHR$(159) : REM БИРЮЗОВЫЙ
260 LET Y$=CHR$(158) : REM ЖЕЛТЫЙ
```

Если ваш дисплей дает цветное изображение, то воспользуйтесь соответствующими кодами; в противном случае строки с B$, R$ и т. д. исключите.

```
300 REM XXXXXXXXXXXXXXXXXXX НАЧАЛО ДНЯ XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
310 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
320 PRINT R$+" ОБСЛУЖИВАНИЕ В БАНКЕ"
330 PRINT
340 PRINT B$+"          ДЕНЬ ";DAY
350 PRINT
360 PRINT "СКОРОСТЬ ПОСТУПЛЕНИЯ = 24/ЧАС"
370 PRINT "СКОРОСТЬ ОБСЛУЖИВАНИЯ =  6/ЧАС"
380 PRINT M$+" СКОЛЬКО СЕГОДНЯ"
390 PRINT M$+" НУЖНО КЛЕРКОВ? 1-10"
400 PRINT B$
410 INPUT AS
420 LET AS=INT(AS+0.5)
430 IF AS<1 OR AS>10 THEN GOTO 410
440 LET S(DAY)=AS
450 PRINT
460 PRINT M$+" МАКСИМАЛЬНАЯ ДОПУСТИМАЯ"
470 PRINT M$+" ДЛИНА ОЧЕРЕДИ"
480 PRINT B$
490 INPUT MAX
500 LET MAX=INT(MAX+0.5)
510 LET M(DAY)=MAX
520 FOR I=1 TO AS
530 LET A(I)=0
540 NEXT I
600 REM XXXXXXXXXXXXXXXXXXXXX СОБЫТИЯ ЗА ДЕНЬ XXXXXXXXXXXXXXXXXXXXXXXX
610 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
620 PRINT R$+"  ОБСЛУЖИВАНИЕ В БАНКЕ"
630 PRINT
640 PRINT "ДЕНЬ";DAY;" ОТКРЫТО 1000-1600"
650 PRINT
660 PRINT "КЛЕРКИ: ";AS
670 PRINT
680 LET LC=0 : REM ПОТЕРЯ КЛИЕНТА
690 LET TT=0 : REM ОБЩЕЕ ВРЕМЯ ОБСЛУЖИВАНИЯ
700 LET Q=0 : REM ДЛИНА ОЧЕРЕДИ
710 LET T=0 : REM ВРЕМЯ В МИНУТАХ
720 GOSUB 1510
800 REM XXXXXXXXXXXXXXXX МОДЕЛИРОВАНИЕ С ШАГОМ В 1 МИН XXXXXXXXXXXXXXXXX
810 FOR T=1 TO 360
820 IF T>TA THEN GOSUB 1510
830 LET TS=0 : REM ЧИСЛО ОБСЛУЖИВАЮЩИХ КЛЕРКОВ
840 FOR J=1 TO AS
850 IF Q>0 AND A(J)<=T THEN GOSUB 1710
860 IF A(J)>T THEN LET TS=TS+1
870 NEXT J
880 LET P=7
890 GOSUB 1810
900 PRINT G$+"ВРЕМЯ: ";1000+T+40*INT(T/60)
910 PRINT
920 PRINT B$+"ЖДУЩИХ КЛИЕНТОВ";STR$(Q);" "
930 PRINT
940 PRINT M$+"ОБСЛУЖИВАЮЩИХ КЛЕРКОВ";STR$(TS);" "
950 PRINT
960 PRINT Y$+"ПОТЕРЯННЫХ КЛИЕНТОВ";LC
970 PRINT
980 PRINT C$+"ОБЩЕЕ ВРЕМЯ"
990 PRINT C$+"ОБСЛУЖИВАНИЯ ";TT
1000 NEXT T
```

Здесь вводятся необходимое количество клерков и максимальное число мест ожидания в очереди.

Здесь каждую минуту вычисляются состояния процесса поступления клиентов и процесса обслуживания их клерками.

```
1100 REM XXXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ИТОГ ЗА ДЕНЬ XXXXXXXXXXXXXXXXXXXX
1110 LET T(DAY)=INT(TT/AS+.5)
1120 LET L(DAY)=LC+Q
1130 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
1140 PRINT R$+"       БАНК - ИТОГ"
1150 PRINT
1160 PRINT B$+"СКОРОСТЬ ПОСТУПЛЕНИЯ = 24/ЧАС"
1170 PRINT B$+"СКОРОСТЬ ОБСЛУЖИВАНИЯ =  6/ЧАС"
1180 PRINT R$+"КЛЕРКИ  МАКС  ПОТЕРИ  ВРЕМЯ "
1190 PRINT R$+"          ОЧ            ОБСЛУЖ"
1200 PRINT
1210 FOR I=1 TO DAY
1220 PRINT TAB(3-LEN(STR$(S(I))));S(I);
1230 PRINT TAB(8-LEN(STR$(M(I))));M(I);
1240 PRINT TAB(13-LEN(STR$(L(I))));L(I);
1250 PRINT TAB(20-LEN(STR$(T(I))));T(I)
1260 NEXT I
1270 PRINT
1280 PRINT G$+" ЧТОБЫ ПРОДОЛЖИТЬ, НАЖМИТЕ Y"
1290 GET H$ : REM LET H$=INKEY$
1300 IF H$<>"Y" THEN GOTO 1290
1310 LET DAY=DAY+1
1320 IF DAY<7 THEN GOTO 310
1400 REM XXXXXXXXXXXXXXXXXX ЕЩЕ ОДНА НЕДЕЛЯ? XXXXXXXXXXXXXXXXXXXXXX
1410 PRINT
1420 PRINT G$+" ЕЩЕ ОДНА НЕДЕЛЯ? Y ИЛИ N"
1430 GET H$ : REM LET H$=INKEY$
1440 IF H$<>"Y" AND H$<>"N" THEN GOTO 1430
1450 IF H$="Y" THEN GOTO 200
1460 END : REM STOP
1500 REM XXXXXXXXXX МОМЕНТ ПОСТУПЛЕНИЯ СЛЕД. КЛИЕНТА XXXXXXXXXXXXXXXX
1510 LET Q=Q+1
1520 LET TA=T-LOG(RND(1))/CAR
1530 IF Q<=MAX THEN RETURN
1540 REM XXXXXX ПОТЕРЯ КЛИЕНТА XXXXXXXX
1550 LET Q=MAX
1560 LET P=18
1570 GOSUB 1810
1580 PRINT Y$+"    ДЛИННАЯ ОЧЕРЕДЬ"
1590 PRINT Y$+"    КЛИЕНТ ПОТЕРЯН"+B$
1600 FOR I=1 TO 200
1610 NEXT I
1620 LET P=18
1630 GOSUB 1810
1640 PRINT "
1650 PRINT "
1660 LET LC=LC+1
1670 RETURN
1700 REM XXXXXXXXXXXXXXXXX ВРЕМЯ ОБСЛУЖИВАНИЯ XXXXXXXXXXXXXXXXXXXX
1710 LET Q=Q-1
1720 LET SA=LOG(RND(1))/CSR
1730 LET A(J)=T-SA
1740 LET TT=TT-INT(SA)
1750 RETURN
1800 REM XXXXXXXXXXXXXXXXXX ВЫВОД P СТРОК XXXXXXXXXXXXXXXXXXXXXXXX
1805 REM PRINT AT P,0
1810 PRINT HM$;
1820 FOR I=1 TO P
1830 PRINT
1840 NEXT I
1850 RETURN
```

На дисплей выводится итог каждого дня.

Дополнительные замечания см. в приложении.

Вычисляется момент поступления следующего клиента.

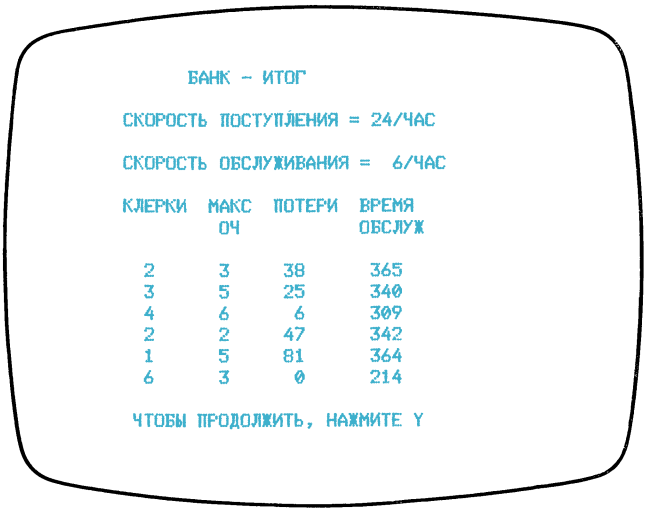
Если очередь достигла своего максимума, выдается сообщение об этом.

Вычисляется длительность обслуживания.

Здесь определяется количество выпечатываемых строк. Если такая возможность есть, воспользуйтесь PRINT AT P,0.

```
  ОБСЛУЖИВАНИЕ В БАНКЕ

         ДЕНЬ  1

СКОРОСТЬ ПОСТУПЛЕНИЯ = 24/ЧАС

СКОРОСТЬ ОБСЛУЖИВАНИЯ =  6/ЧАС

 СКОЛЬКО СЕГОДНЯ
 НУЖНО КЛЕРКОВ? 1-10

? 2

 МАКСИМАЛЬНАЯ ДОПУСТИМАЯ
 ДЛИНА ОЧЕРЕДИ?

? 3
```

```
  ОБСЛУЖИВАНИЕ В БАНКЕ

ДЕНЬ 1  ОТКРЫТО 1000-1600

КЛЕРКИ:  2

ВРЕМЯ:  1220

ЖДУЩИХ КЛИЕНТОВ 3

ОБСЛУЖИВАЮЩИХ КЛЕРКОВ 2

ПОТЕРЯННЫХ КЛИЕНТОВ 14

ОБЩЕЕ ВРЕМЯ ОБСЛУЖИВАНИЯ 252

     ДЛИННАЯ ОЧЕРЕДЬ
     КЛИЕНТ ПОТЕРЯН
```



Существуют формулы, которые могут описать то, что происходит в очереди, но они в значительной мере лишают изучение очередей занимательности. Программа АНАЛИЗ ОЧЕРЕДИ выполнит за вас необходимые вычисления и сообщит о средней длине очереди.

Когда вы приступите к выполнению программы, вам надо будет ввести (INPUT) скорость поступления клиентов (CAR), скорость обслуживания (CSR) и количество клерков (AS). Если клерков меньше, чем CAR/CSR, то очередь довольно быстро становится очень длинной и неуправляемой. Печатается средняя длина очереди EQ.

Вы можете предусмотреть в этой программе и другие вычисления. Например, среднее время ожидания клиента в очереди EQ/CAR. Или среднее время, проведенное клиентом в очереди и на обслуживании: EQ/CAR + 1/CSR. Наконец, среднее число клиентов, т. е. величину EQ + CAR/CSR.

```
10 REM                          ******************
20 REM                          *                  *
30 REM                          * АНАЛИЗ ОЧЕРЕДИ  *
40 REM                          *                  *
50 REM                          ******************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXX
110 DIM F(10) : REM ФАКТОРИАЛЫ
120 LET F(0)=1
130 FOR I=1 TO 10
140 LET F(I)=I*F(I-1)
150 NEXT I
```

Вычисляются факториалы, которые понадобятся в этой программе.

```
200 REM XXXXXXXXXXXXXXXXXXXXXXXXXX ВВОД ДАННЫХ XXXXXXXXXXXXXXXXXXXXXXXXXX
210 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
220 PRINT "   АНАЛИЗ ОЧЕРЕДИ"
230 PRINT
240 PRINT "КЛЕРКОВ НЕ БОЛЕЕ 10"
250 PRINT
260 PRINT "СКОРОСТЬ ПОСТУПЛЕНИЯ"
270 RRINT "КЛИЕНТОВ";
280 INPUT CAR
290 PRINT
300 PRINT "СКОРОСТЬ ОБСЛУЖИВАНИЯ"
310 PRINT "КЛИЕНТОВ";
320 INPUT CSR
330 PRINT
340 PRINT "ЧИСЛО"
350 PRINT "КЛЕРКОВ";
360 INPUT AS
370 LET AS=INT(AS+0.5)
380 IF AS<1 OR AS>10 THEN GOTO 360
390 PRINT
400 REM XXXXXXXXXXXXXXXXXXXXXXXXX СЧЕТ XXXXXXXXXXXXXXXXXXXXXXXXXX
410 IF CSR<=0 THEN GOTO 650
420 LET X=CAR/CSR
430 IF AS<=X THEN GOTO 650
440 LET P1=0
450 FOR K=0 TO AS-1
460 LET P1=P1+X1K/F(K)
470 NEXT K
480 LET P1=P1+X1AS*AS/(F(AS)*(AS-X))
490 LET E=(AS-X)*(AS-X)*F(AS-1)*P1
500 LET EQ=X1(AS+1)/E
510 LET EQ=INT(EQ+0.5)
600 REM XXXXXXXXXXXXXXXXXXXXXXXXX ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXXX
610 PRINT "СРЕДНЕЕ ЧИСЛО"
620 PRINT "КЛИЕНТОВ В ОЧЕРЕДИ"
630 PRINT "РАВНО: ";EQ
640 GOTO 670
650 PRINT "ОЧЕРЕДЬ БУДЕТ"
660 PRINT "ОЧЕНЬ ДЛИННОЙ"
670 PRINT
680 PRINT "ПОВТОРИТЬ? Y ИЛИ N"
690 GET G$ : REM LET G$=INKEY$
700 IF G$<>"Y" AND G$<>"N" THEN GOTO 690
710 IF G$="Y" THEN GOTO 210
```

В этом фрагменте вводятся необходимые данные.

Вычисляется средняя длина очереди.

Дополнительные замечания см. в приложении.

Если в программу ОБСЛУЖИВАНИЕ В БАНКЕ ввести дополнительные ограничения, получится интересная игра под названием МОДНЫЙ МАГАЗИН. Это касается очередей в дорогих магазинах модной одежды.

Предположим, что вы владелец салона модной одежды, который открыт 6 дней в неделю по 6 часов ежедневно с 10 до 16 (в программе это обозначается 1000 и 1600). Продавцов вы можете нанимать поденно. Платить им приходится очень много — 200 долларов в день, а обслуживают они клиентов со скоростью 9 человек в час. Минута обслуживания обходится покупателю примерно в 1.5 доллара.

```
   АНАЛИЗ ОЧЕРЕДИ

КЛЕРКОВ НЕ БОЛЕЕ 10

СКОРОСТЬ ПОСТУПЛЕНИЯ
КЛИЕНТОВ? 24

СКОРОСТЬ ОБСЛУЖИВАНИЯ
КЛИЕНТОВ? 6

ЧИСЛО
КЛЕРКОВ? 3

ОЧЕРЕДЬ БУДЕТ
ОЧЕНЬ ДЛИННОЙ

ПОВТОРИТЬ? Y ИЛИ N
```

```
   АНАЛИЗ ОЧЕРЕДИ

КЛЕРКОВ НЕ БОЛЕЕ 10

СКОРОСТЬ ПОСТУПЛЕНИЯ
КЛИЕНТОВ? 24

СКОРОСТЬ ОБСЛУЖИВАНИЯ
КЛИЕНТОВ? 6

ЧИСЛО
КЛЕРКОВ? 5

СРЕДНЕЕ ЧИСЛО
КЛИЕНТОВ В ОЧЕРЕДИ
РАВНО:  2

ПОВТОРИТЬ? Y ИЛИ N
```

Сначала предполагается, что клиенты появляются со средней скоростью 6 человек в час. Прибегнув, однако, к рекламе (это обойдется вам в 100 долларов), вы можете увеличить скорость их появления на 6 человек в час. Если же вы дадите и скидку 10% для всех покупателей, скорость возрастает еще на 6 человек в час. Так, если, например, в пер-

вый же день вы воспользовались рекламой и сделали 10-процентную скидку, то вправе ожидать, что каждый час будет появляться приблизительно по 18 новых покупателей. (Скорость обслуживания клиентов всегда остается 9 человек в час.)

Клиент не любит ждать. Он встанет в очередь, только если она совсем короткая. Если же очередь очень длинная, покупатель уйдет чрезвычайно раздраженным. Настолько раздраженным, что молва об этом отпугнет других потенциальных покупателей. В результате скорость поступления клиентов упадет на 10%.

Цель игры — получить за неделю максимальную прибыль.

```
10 REM                    ******************
20 REM                    *                  *
30 REM                    * МОДНЫЙ МАГАЗИН *
40 REM                    *                  *
50 REM                    ******************
60 REM
70 REM
100 REM %%%%%%%%%%%%%%%%% НАЧАЛЬНАЯ УСТАНОВКА %%%%%%%%%%%%%%%%%%%%%%%%%%%
110 LET CS$=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET HM$=CHR$(19) : REM КОД ИСХОДНОГО ПОЛОЖЕНИЯ
125 LET B$=CHR$(31) : REM ГОЛУБОЙ
127 LET R$=CHR$(28) :REM КРАСНЫЙ
129 LET M$=CHR$(156) : REM ФИОЛЕТОВЫЙ
131 LET G$=CHR$(30) : REM ЗЕЛЕНЫЙ
133 LET C$=CHR$(159) : REM БИРЮЗОВЫЙ
135 LET Y$=CHR$(156) : REM ЖЕЛТЫЙ
```

Если ваш компьютер дает цветное изображение, воспользуйтесь соответствующими кодами; в противном случае исключите B$, R$ и т. д.

```
140 DIM A(10)
150 DIM P(6)
160 LET CAR=0.1 : REM СКОРОСТЬ ПОСТУПЛЕНИЯ
170 LET CSR=0.15 : REM СКОРОСТЬ ОБСЛУЖИВАНИЯ
180 LET DAY=1
190 LET PR=0 : REM ПРИБЫЛЬ
200 REM %%%%%%%%%%%%%%%%%% НАЧАЛО ДНЯ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
210 PRINT CS$; : REM ОЧИСТИТЬ ЭКРАН
220 PRINT R$+"      МОДНЫЙ МАГАЗИН"
230 PRINT B$+"           ДЕНЬ ";DAY
240 PRINT
250 PRINT B$+"СКОРОСТЬ ПОСТУПЛЕНИЯ=";INT(600*CAR)/10
260 PRINT
270 PRINT M$+"100 ДОЛЛАРОВ НА РЕКЛАМУ?"
280 PRINT M$+"Y ИЛИ N ";
290 INPUT A$
300 LET AD=0
310 IF A$="Y" THEN LET CAR=CAR+0.1
320 IF A$="Y" THEN LET AD=1
330 PRINT
340 PRINT B$+"НОВАЯ СКОРОСТЬ ПОСТУПЛЕНИЯ=";INT(600*CAR)/10
350 PRINT
```

У вас спрашивают, нужна ли вам реклама и (или) будете ли вы давать 10-процентную скидку. Вас также спрашивают, сколько вам нужно продавцов.

```
360 LET DIS=1
370 PRINT M$+"СКИДКА 10%? Y ИЛИ N"
380 INPUT A$
390 IF A$="Y" THEN LET DIS=0.9
400 IF A$="Y" THEN LET CAR=CAR+0.1
410 PRINT
420 PRINT B$+"НОВАЯ СКОРОСТЬ ПОСТУПЛЕНИЯ=";INT(600*CAR)/10
430 PRINT
440 PRINT B$+"СКОРОСТЬ ОБСЛУЖИВАНИЯ= 9/ЧАС"
450 PRINT
460 PRINT M$+"ЧИСЛО ПРОДАВЦОВ"
470 PRINT M$+"ПО 200 ДОЛЛАРОВ? 1-10";
480 INPUT AS
490 LET AS=INT(AS+0.5)
500 IF AS<1 OR AS>10 THEN GOTO 480
510 LET MAX=2+INT(AS/4)
520 PRINT
530 PRINT B$+"МАКСИМАЛЬНАЯ ДЛИНА ОЧЕРЕДИ";MAX;"
540 FOR I=1 TO AS
550 FOR A(I)=0
560 NEXT I
570 PRINT G$+" ЧТОБЫ ПРОДОЛЖИТЬ,НАЖМИТЕ Y"
580 GET H$ : REM LET H$=INKEY$
590 IF H$<>"Y" THEN GOTO 580
600 REM XXXXXXXXXXXXXXXXXXXXX СОБЫТИЯ ЗА ДЕНЬ XXXXXXXXXXXXXXXXXXXXXXXXXX
610 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
620 PRINT R$+"     МОДНЫЙ МАГАЗИН"
630 PRINT
640 PRINT R$+"ДЕНЬ";DAY;" ОТКРЫТО 1000-1600"
650 PRINT
660 PRINT R$+"ПРОДАВЦОВ: ";AS
670 PRINT
680 LET LC=0 : REM ПОТЕРЯННЫЕ КЛИЕНТЫ
690 LET TT=0 : REM ОБЩАЯ ВЫРУЧКА
700 LET Q=0 : REM ДЛИНА ОЧЕРЕДИ
710 LET T=0 : REM ВРЕМЯ В МИНУТАХ
720 GOSUB 1710
800 REM XXXXXXXXXXXXXXXX МОДЕЛИРОВАНИЕ С ШАГОМ В 1 МИНУТУ XXXXXXXXXXXXXXX
810 FOR T=1 TO 360
820 IF T>TA THEN GOSUB 1710
830 LET TS=0 : REM ЧИСЛО ПРОДАВЦОВ
840 FOR J=1 TO AS
850 IF Q>0 AND A(J)<=T THEN GOSUB 1910
860 IF A(J)>T THEN LET TS=TS+1
870 NEXT J
880 LET P=7
890 GOSUB 2010
900 PRINT G$+"ВРЕМЯ: ";1000+T+40*INT(T/60)
910 PRINT
920 PRINT B$+"СКОРОСТЬ ПОСТУПЛЕНИЯ";STR$(INT(600*CAR)/10);"  "
930 PRINT
940 PRINT M$+"ЖДУЩИХ КЛИЕНТОВ";Q
950 PRINT
960 PRINT C$+"ОБСЛУЖИВАЮЩИХ ПРОДАВЦОВ";STR$(TS);" "
970 PRINT
980 PRINT Y$+"ПОТЕРЯННЫХ КЛИЕНТОВ";LC
990 PRINT
1000 PRINT R$+"ВЫРУЧКА";TT
1010 NEXT T
```

Все вычисления выполняются с интервалом в 1 мин.

```
1100 REM XXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ИТОГ ЗА ДЕНЬ XXXXXXXXXXXXXXXXXXXXXX
1110 LET P(DAY)=INT(TT-200*AS-100*AD+0.5)
1120 LET PR=PR+P(DAY)
1130 LET LC=LC+Q
1140 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
1150 PRINT R$+"     МОДНЫЙ МАГАЗИН"
1160 PRINT
1170 PRINT R$+"ИТОГ : ДЕНЬ";DAY
1180 PRINT
1190 PRINT B$+"СКОРОСТЬ ПОСТУПЛЕНИЯ=";INT(600*CAR)/10
1200 PRINT
1210 PRINT B$+"СКОРОСТЬ ОБСЛУЖИВАНИЯ= 9/ЧАС"
1220 PRINT
1230 PRINT B$+"ПРОДАВЦОВ";AS
1240 PRINT
1250 PRINT Y$+"ПОТЕРЯННЫХ КЛИЕНТОВ";LC
1260 PRINT
1270 PRINT R$+"ВЫРУЧКА";TT
1280 PRINT
1290 PRINT B$+"ПРИБЫЛЬ ";P(DAY)
1300 PRINT
1310 PRINT G$+"ЧТОБЫ ПРОДОЛЖИТЬ, НАЖМИТЕ Y"
1320 GET H$ : REM LET H$=INKEY$
1330 IF H$<>"Y" THEN GOTO 1320
1340 LET DAY=DAY+1
1350 IF DAY<7 THEN GOTO 210
```

В конце дня подводится итог.

```
1400 REM XXXXXXXXXXXXXXXXXXX КОНЕЦ НЕДЕЛИ XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1410 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
1420 PRINT R$+"     МОДНЫЙ МАГАЗИН"
1430 PRINT
1440 PRINT B$+"     ИТОГ ЗА НЕДЕЛЮ"
1450 PRINT
1460 PRINT R$+"     ДЕНЬ    ПРИБЫЛЬ"
1470 FOR I=1 TO 6
1480 PRINT "    ";I;TAB(13-LEN(STR$(P(I))));P(I)
1490 NEXT I
1500 PRINT
1510 PRINT R$+"   ИТОГО :";TAB(13-LEN(STR$(PR)));PR
```

Подводится итог за неделю.

```
1600 REM XXXXXXXXXXXXXXXXXXX ЕЩЕ ОДНА НЕДЕЛЯ? XXXXXXXXXX
1610 PRINT
1620 PRINT G$+" ЕЩЕ ОДНА НЕДЕЛЯ? Y ИЛИ N"+B$
1630 GET H$ : REM LET H$=INKEY$
1640 IF H$<>"Y" AND H$<>"N" THEN GOTO 1630
1650 IF H$="Y" THEN GOTO 160
1660 END : REM STOP
1700 REM XXXXXXXXXX МОМЕНТ ПОСТУПЛЕНИЯ СЛЕДУЮЩЕГО КЛИЕНТА XXXXXXXXXXXX
1710 LET Q=Q+1
1720 LET TA=T-LOG(RND(1))/CAR
1730 IF Q<=MAX THEN RETURN
1740 REM XXXXXX ПОТЕРЯ КЛИЕНТА XXXXXX
1750 LET Q=MAX
1760 LET P=19
1770 GOSUB 2010
```

Здесь вычисляется момент поступления каждого клиента.

```
1780 PRINT Y%+"  ДЛИННАЯ ОЧЕРЕДЬ"
1790 PRINT Y%+"  КЛИЕНТ ПОТЕРЯН"
1800 FOR I=1 TO 200
1810 NEXT I
1820 LET P=19
1830 GOSUB 2010
1840 PRINT "
1850 PRINT "
1860 LET LC=LC+1
1870 LET CAR=0.1+(CAR-0.1)*0.9
1880 RETURN
```

Если очередь слишком длинная, сообщение об этом выводится на экран.

```
1900 REM XXXXXXXXXXXXXXXXXXX ВРЕМЯ ОБСЛУЖИВАНИЯ XXXXXXXXXXXXXXXXXXXXXXXX
1910 LET Q=Q-1
1920 LET SA=LOG(RND(1))/CSR
1930 LET A(J)=T-SA
1940 LET TT=TT-INT(SA*(1+RND(1))*DIS)
1950 RETURN
```

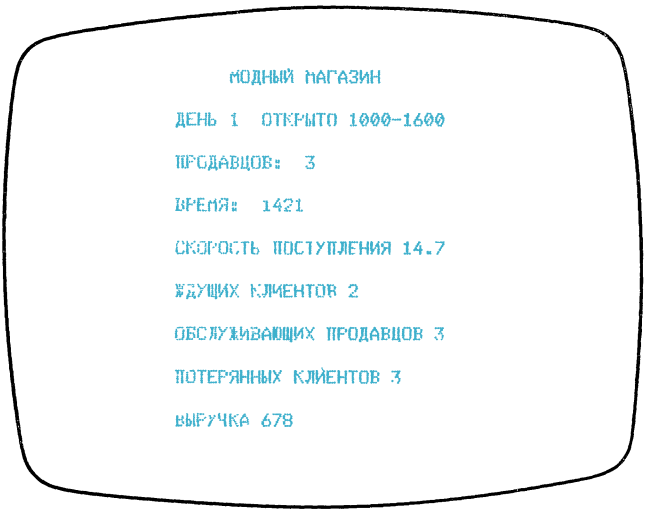
Здесь вычисляется длительность обслуживания каждого клиента, а также полученная от него сумма.

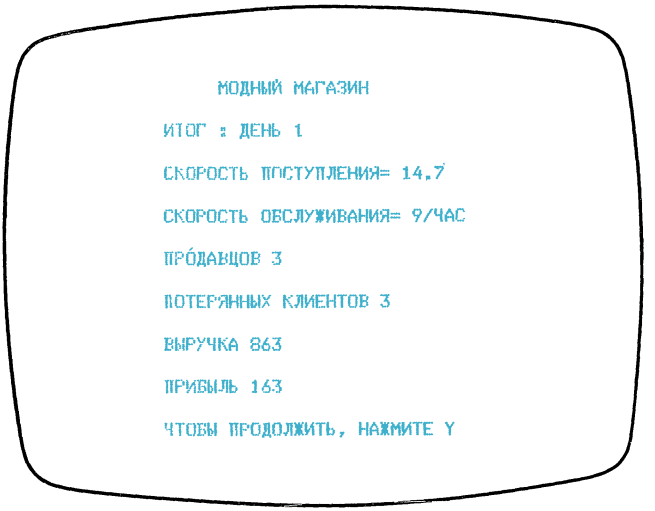
```
2000 REM XXXXXXXXXXXXXXXXXXX ВЫВОД P СТРОК XXXXXXXXXXXXXXXXXXXXXXXXXXXX
2005 REM PRINT AT P,0
2010 PRINT HM%;
2020 FOR I=1 TO P
2030 PRINT
2040 NEXT I
2050 RETURN
```

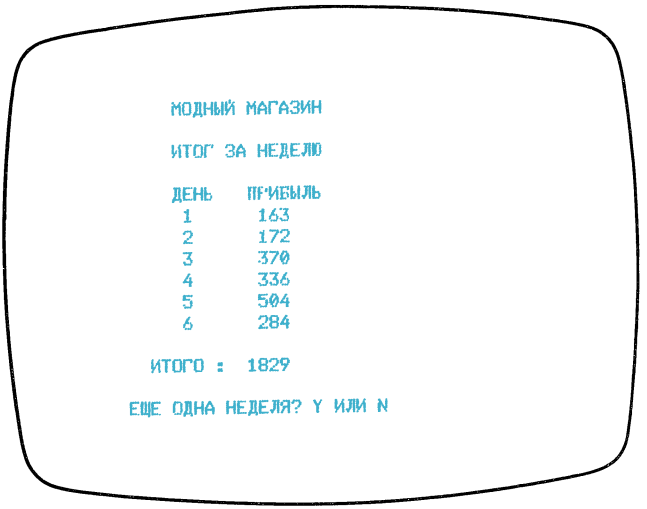
Если можно, воспользуйтесь PRINT AT P,0.

Дополнительные замечания см. в приложении.

```
          МОДНЫЙ МАГАЗИН
             ДЕНЬ  1

 СКОРОСТЬ ПОСТУПЛЕНИЯ= 6

 100 ДОЛЛАРОВ НА РЕКЛАМУ?
 Y ИЛИ  N ? Y

 НОВАЯ СКОРОСТЬ ПОСТУПЛЕНИЯ= 12

 СКИДКА 10%? Y ИЛИ N
 ? Y

 НОВАЯ СКОРОСТЬ ПОСТУПЛЕНИЯ= 18

 СКОРОСТЬ ОБСЛУЖИВАНИЯ= 9/ЧАС

 ЧИСЛО ПРОДАВЦОВ
 ПО 200 ДОЛЛАРОВ? 1-10? 3

 МАКСИМАЛЬНАЯ ДЛИНА ОЧЕРЕДИ 2

 ЧТОБЫ ПРОДОЛЖИТЬ, НАЖМИТЕ Y
```


МОДНЫЙ МАГАЗИН
ДЕНЬ 1 ОТКРЫТО 1000-1600
ПРОДАВЦОВ: 3
ВРЕМЯ: 1421
СКОРОСТЬ ПОСТУПЛЕНИЯ 14.7
ЖДУЩИХ КЛИЕНТОВ 2
ОБСЛУЖИВАЮЩИХ ПРОДАВЦОВ 3
ПОТЕРЯННЫХ КЛИЕНТОВ 3
ВЫРУЧКА 678


МОДНЫЙ МАГАЗИН
ИТОГ : ДЕНЬ 1
СКОРОСТЬ ПОСТУПЛЕНИЯ= 14.7
СКОРОСТЬ ОБСЛУЖИВАНИЯ= 9/ЧАС
ПРОДАВЦОВ 3
ПОТЕРЯННЫХ КЛИЕНТОВ 3
ВЫРУЧКА 863
ПРИБЫЛЬ 163
ЧТОБЫ ПРОДОЛЖИТЬ, НАЖМИТЕ Y


МОДНЫЙ МАГАЗИН
ИТОГ ЗА НЕДЕЛЮ
ДЕНЬ ПРИБЫЛЬ
1 163
2 172
3 370
4 336
5 504
6 284
ИТОГО : 1829
ЕЩЕ ОДНА НЕДЕЛЯ? Y ИЛИ N

Прелестные картинки

О функциях двух переменных

Площадь прямоугольника, имеющего 4 см в длину и 6 см в ширину, равна $4*6 = 24$ см^2. В общем случае площадь Z прямоугольника с длинами сторон X и Y выражается формулой $Z = X*Y$. В таком случае говорят, что Z есть функция от X и Y, или Z есть *функция двух переменных* X и Y.

Можно привести немало примеров зависимости одной величины от двух других. Вот, скажем, формула для объема V цилиндра, выраженного через его высоту HT и радиус R:

$$V = PI*R*R*HT.$$

Значение V зависит от значений, принимаемых R и HT (PI — величина постоянная). Таким образом, V есть функция двух переменных R и HT.

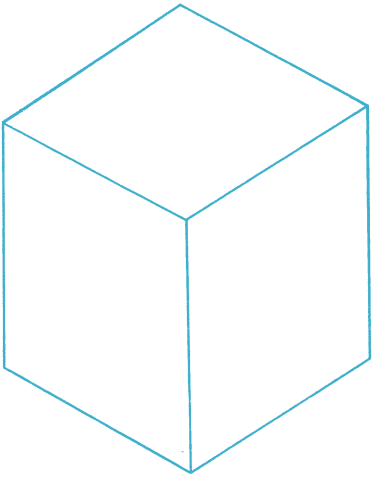
Приведем еще несколько примеров функций двух переменных:

$$Z = COS(X*Y),$$
$$Z = COS(X*EXP(-Y/5)),$$
$$Z = SIN(X)*COS(Y),$$
$$Z = COS(X)*COS(Y),$$
$$Z = X - X*X*X/12 - Y*Y/2 + 1/4$$
$$Z = 5 - (X + 2)*(X + 1)*(X - 1)*(X - 2) - Y*Y.$$

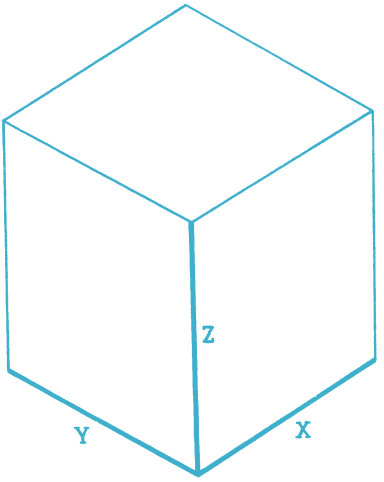
Как же «выглядят» такие функции? Во второй главе вы познакомились с тем, как чертить графики функций одной переменной. Напомним, как это делается. Если Y — функция переменной X, то для каждого значения X мы вычисляем соответствующее значение Y и строим точку с координатами (X, Y).

Попробуем применить аналогичную процедуру для функций двух переменных. Если Z — функция X и Y, то для каждого значения X и Y можно вычислить соответствующее значение Z. Если бы в нашем распоряжении было три измерения, то мы могли бы построить точку с высотой Z над точкой с координатами (X, Y). В результате получилась бы поверхность, похожая на остров.

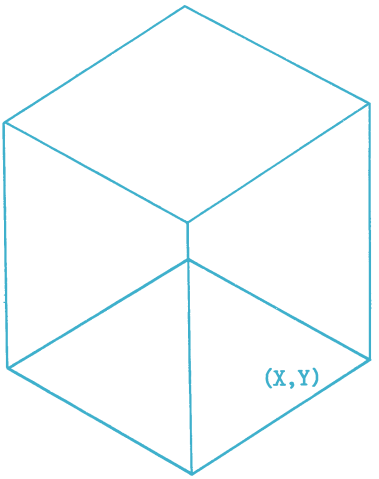
Экран нашего компьютера и лист бумаги не имеют трех измерений. Поэтому нам надо изобразить эти три измерения при помощи возможностей двумерного пространства; это очень напоминает фотографию или чертеж куба.



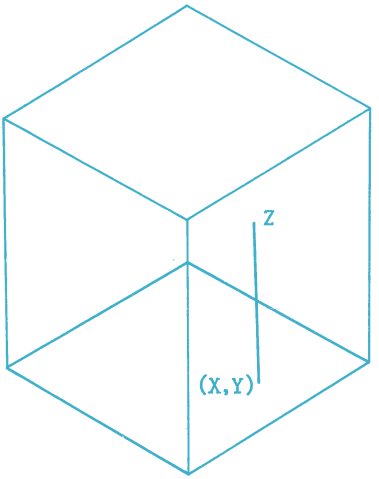
Будем предполагать, что этот куб является частью трехмерного пространства, в котором нам предстоит строить графики. Самую нижнюю вершину назовем началом координат. Правое нижнее ребро будет осью X, а левое нижнее — осью Y. Исходящее из этой вершины вертикальное ребро будет соответствовать третьему измерению.



Таким образом, основание куба представляет собой кусок плоскости X,Y, а точки на ней задаются своими декартовыми координатами (X,Y).



Чтобы изобразить график, мы находим точку (X,Y) в основании нашего куба, затем вычисляем соответствующее значение Z и отмечаем точку над точкой (X,Y) на расстоянии Z.



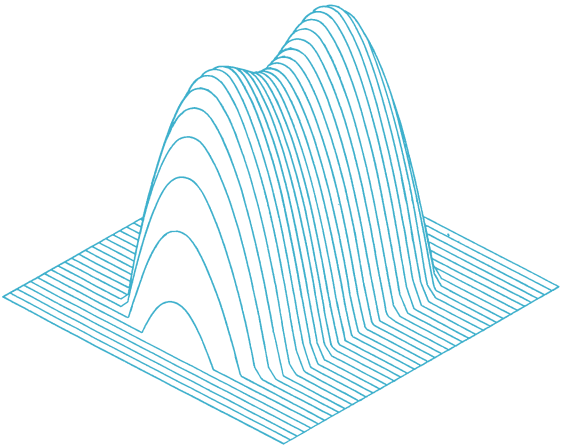
Чтобы произвести на нашем экране необходимые построения, нужно знать соответствие между координатами экрана и координатами куба. Если (A, B) обозначает координаты экрана для точки, находящейся в основании куба, то значение X и Y можно получить по следующей формуле (здесь через S обозначено SQR(2)/2):

$$X = S*A + S*B,$$

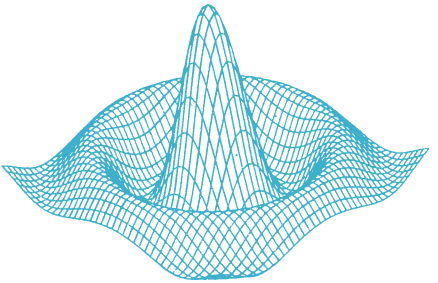
$$Y = -S*A + S*B.$$

Чтобы построить график, нужно начать с точек в основании куба. Предположим, что такая точка на нашем экране имеет координаты (A, B). По выписанным выше формулам вычисляем соответствующие координаты (X,Y) в основании куба. Используя значения X и Y, находим значение Z. Теперь мы можем построить на нашем экране точку с координатами (A, B + Z). Однако для получения более глубокой перспективы мы уменьшаем вертикальный отрезок, умножая его длину на S. Итак, на экране появляется точка с координатами (A, S*(B + Z)).

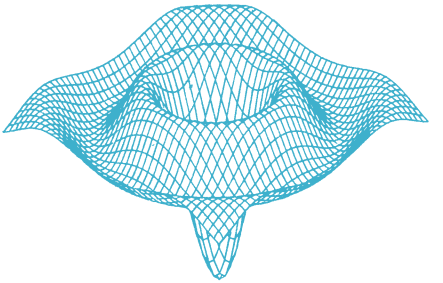
Если бы мы строили такие точки для всех точек в основании куба, то получалась бы ужасная мазня. Чтобы избежать этого, мы строим только избранные точки, как правило, те, которые соответствуют точкам на прямых, параллельных оси X и (или) оси Y. Поэтому поверхность выглядит так, как будто ее разрезали на полоски. Ниже приводится подобный пример.



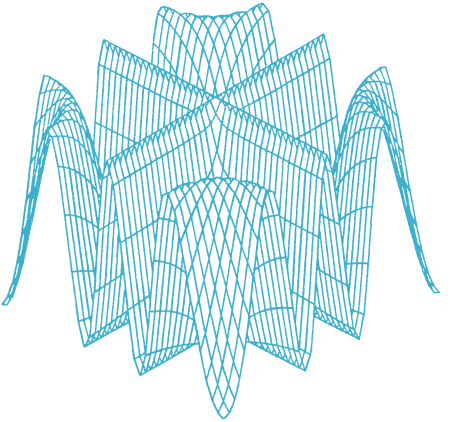
Далее на рисунках представлено еще несколько примеров.



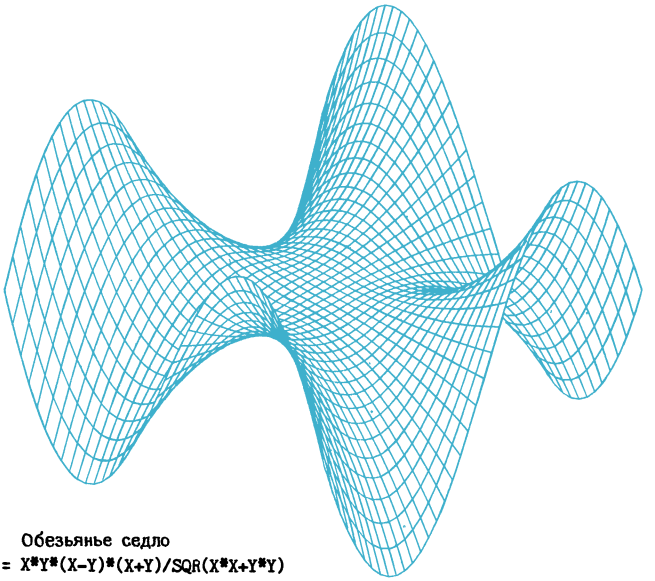
```
Z = SIN(R)/R
      R = SQR(X*X+Y*Y)
```



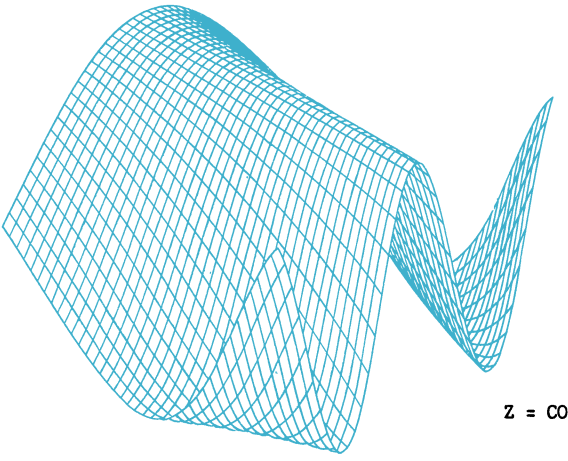
```
Z = -SIN(R)/R
      R = SQR(X*X+Y*Y)
```



```
Z = COS(X*Y)
```



Обезьянье седло
Z = X*Y*(X-Y)*(X+Y)/SQR(X*X+Y*Y)



Z = COS(X*EXP(-Y/5))

Программа ПОВЕРХНОСТИ позволит вам чертить графики функций двух переменных. Качество полученных графиков в большой степени зависит, разумеется, от графических устройств вашего компьютера. В программе предусмотрено построение 5 функций. Если пожелаете, можно добавить еще и другие.

В программе использован алгоритм «скрытой точки»; это означает, что точки графика, скрытые другими точками, не изображаются.

```
 10 REM                        ***************
 20 REM                        *             *
 30 REM                        * ПОВЕРХНОСТИ *
 40 REM                        *             *
 50 REM                        ***************
 60 REM
 70 REM
100 REM XXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET SX=176 : REM РАЗМЕР ЭКРАНА ПО ГОР
130 LET SY=160 : REM РАЗМЕР ЭКРАНА ПО ВЕРТ
140 LET RATIO=0.6 : REM СДЕЛАТЬ СТРОКИ ПО ГОР И ВЕРТ ОДНОЙ ДЛИНЫ
150 LET HY=SY/2
160 LET HX=SX/2
170 LET S=SQR(2)/2
180 LET AA=HX*S
185 GOSUB 1310 : REM ДОПОЛНИТЕЛЬНАЯ УСТАНОВКА ДЛЯ VIC 20
190 PRINT CS : REM ОЧИСТИТЬ ЭКРАН
200 PRINT "      ПОВЕРХНОСТИ"
210 PRINT
220 PRINT "1. Z = COS(R)"
230 PRINT
240 PRINT "2. Z = EXP(-R*R)"
250 PRINT
260 PRINT "3. Z = SIN(R)/R"
270 PRINT
280 PRINT " ГДЕ R=SQR(X*X+Y*Y)"
290 PRINT
300 PRINT "4. Z = COS(X)*COS(Y)"
310 PRINT
320 PRINT "5. ОБЕЗЬЯНЬЕ СЕДЛО"
330 PRINT
340 REM МЕСТО ДЛЯ ДРУГИХ ФУНКЦИЙ
400 PRINT "УКАЖИТЕ НОМЕР"
410 PRINT "УРАВНЕНИЯ";
420 INPUT N
430 PRINT CS : REM ОЧИСТИТЬ ЭКРАН
440 GOSUB 1010 : REM ПОДГОТОВИТЬ ЭКРАН, ЕСЛИ НУЖНО

500 REM XXXXXXXXXXXXXXXXXXXX ПОСТРОЕНИЕ ГРАФИКОВ XXXXXXXXXXXXXXXXXXXXXX
510 FOR A=-AA TO AA+5*S
520 LET MAX=-HY
530 LET BB=AA+A-10*S*INT((A+ABS(A))/(10*S))
540 FOR B=-BB TO BB+S*4 STEP 10*S
550 LET X=S*(A+B)
560 LET Y=S*(B-A)
570 LET Z=B
580 LET R=SQR(X*X+Y*Y)
590 IF N=1 THEN LET Z=10*COS(R/5)+B
600 IF N=2 THEN LET Z=75*EXP(-R*R/600)+B
610 IF N=3 AND R<>0 THEN LET Z=125*SIN(R/5)/R+B
620 IF N=4 THEN LET Z=10*COS(X/10)*COS(Y/10)+B
```

Возьмите подходящие для вашего компьютера значения SX, SY и RATIO.

При желании можно добавить другие функции.

Обратите внимание, что может произойти, если вы укажете число, отличающееся от 1—5.

Это основная программа построения графиков.

```
630 IF N=5 AND R()0 THEN LET Z=X*Y*(X-Y)*(X+Y)/(1000*R)+B
640 REM МЕСТО ДЛЯ ДРУГИХ ФУНКЦИЙ
700 IF Z<MAX THEN GOTO 760
710 LET MAX=Z
720 LET U=HX+A

730 LET V=HY+Z*S/RATIO
740 IF V<0 OR V>SY THEN GOTO 760
750 GOSUB 1110 : REM PLOT U,V
760 NEXT B
770 NEXT A
```

Использования MAX позволяет предотвратить построение «скрытых точек».

Если на вашем компьютере есть команда PLOT, то GOSUB можно пренебречь.

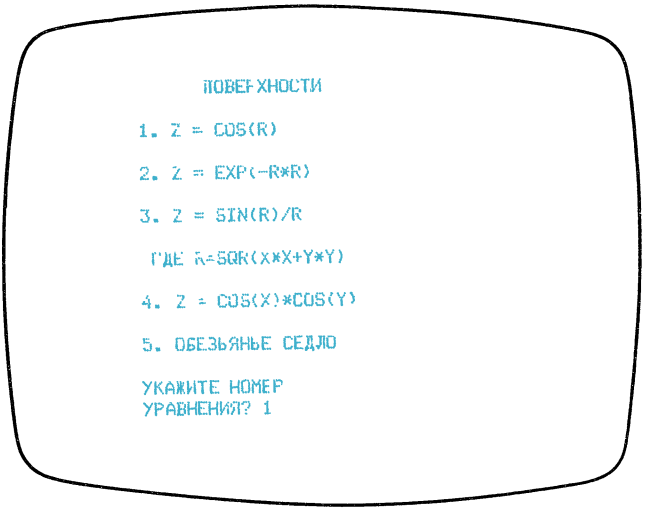
```
800 REM XXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXX
810 GET G$ : REM LET G$=INKEY$
820 IF G$="" THEN GOTO 810
830 GOSUB 1210 : REM ВОССТАНОВИТЬ ЭКРАН, ЕСЛИ НУЖНО
840 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
850 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
860 GET G$: REM LET G$=INKEY$
870 IF G$()"Y" AND G$()"N" THEN GOTO 860
880 IF G$="Y" THEN GOTO 190
890 END : REM STOP

1000 PEM XXXXXXXXXXXX ПОДГОТОВКА ЭКРАНА ДЛЯ VIC 20 XXXXXXXXXXXXXXXXXX
1010 POKE 36869,R8+12:POKE 36867,(PEEK(36867) AND 128) OR 21
1020 FOR I=RR TO SS:POKE I,0:NEXT I
1030 FOR I=0 TO 219:POKE PP+I,I-32*Q:POKE QQ+I,2:NEXT I
1040 RETURN

1100 REM XXXXXXXXXX ПОСТРОЕНИЕ ПОСРЕДСТВОМ POKE ДЛЯ VIC 20 XXXXXXXXXXXX
1110 V=SY-Y
1120 J=INT(U/8)
1130 L=INT(U-8*J)
1140 I=INT(V/16)
1150 K=INT(V-16*I)
1160 W=RR+I*352+J*16+K
1170 POKE W,PEEK(W) OR 2↑(7-L)
1180 RETURN
```

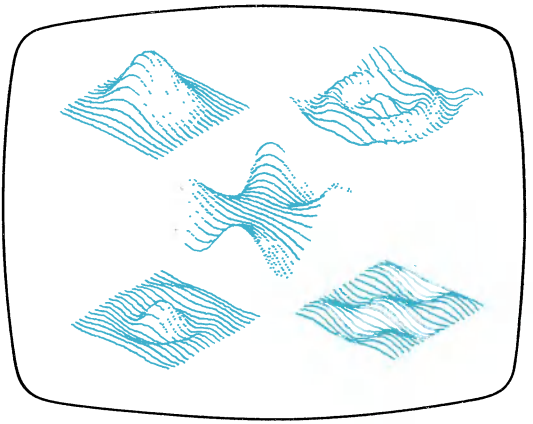
Если приходится строить при помощи POKE, то нужно указать соответствующие детали.

```
1200 REM XXXXXXXXXXXX ВОССТАНОВЛЕНИЕ ЭКРАНА ДЛЯ VIC 20 XXXXXXXXXXXXXX
1210 POKE 36869,R8:POKE 36867,R6:POKE 198,0
1220 RETURN

1300 REM XXXXXXXXXXXXXXXX УСТАНОВКА ДЛЯ VIC 20 XXXXXXXXXXXXXXXXXXXX
1310 Q=PEEK(44))=18:PP=7680+Q*3584:QQ=38400+Q*512
1320 IFQ=-1ANDPEEK(44)<32THENPRINT"ПРОГРАММА ПРЕРВАНА - СМ ПРИЛОЖЕНИЕ":END
1330 RR=4096-Q*512:SS=RR+3583:R8=PEEK(36869):R6=PEEK(36867)
1340 RETURN
```

Общие замечания по программированию см. в приложении.

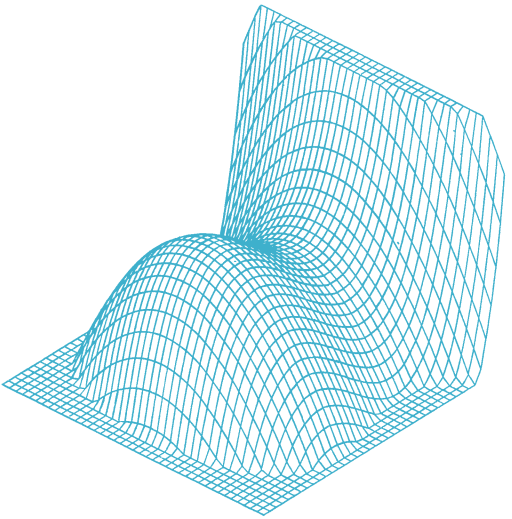
Имеется другой способ для изображения функций двух переменных. Его идея аналогична той, что используется составителями атласов физических или рельефных карт. На карты наносятся изолинии и соответствующие цвета для указания высоты выше (или ниже) уровня

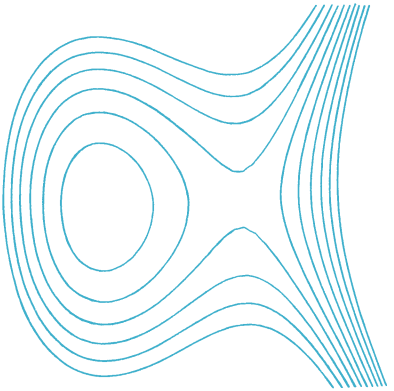




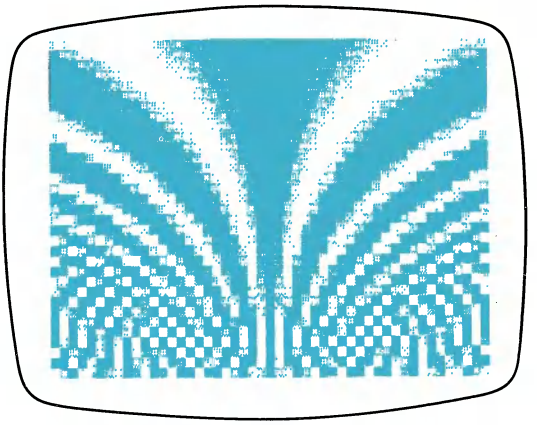
моря. Изолинии представляют собой кривые, соединяющие точки одинаковой высоты выше (или ниже) уровня моря. Взглянув на такую карту, вы получите полное представление об интересующем вас регионе: возвышенность это или равнина.

Точно так же можно изобразить «карту изолиний» функции двух переменных. Такая карта позволяет представить, как выглядит функция. Далее в качестве примера приводится график одной функции, а затем — карта изолиний для нее.





Для изображения карт изолиний на экране дисплея можно использовать различные цвета и (или) символы. Получаются очень интересные и красивые картинки. Ниже приводится пример функции Z = = COS(X*EXP(−Y/5)).



При составлении карт изолиний предполагается, что на экране заданы декартовы координаты X,Y. Для каждой точки (X,Y) экрана нужно вычислить значение Z. Если Z — большое число, то в точку (X,Y) ставится темная «клякса», а если маленькое, то светлая. Промежуточные значения окрашиваются более или менее темным тоном в зависимости от имеющихся на машине возможностей.

Программа ИЗОЛИНИИ выдает рельефные карты функций двух переменных.

```
10 REM                    ************
20 REM                    *          *
30 REM                    * ИЗОЛИНИИ *
40 REM                    *          *
50 REM                    ************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXXX УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET HM=CHR$(19) : REM КОД ИСХОДНОГО ПОЛОЖЕНИЯ
130 LET PX=22 : REM КОЛИЧЕСТВО ТОЧЕК ПО ГОР
140 LET PY=23 : REM КОЛИЧЕСТВО ТОЧЕК ПО ВЕРТ
```

Значения PX и PY соответствуют размерам экрана. Задайте размеры экрана вашего компьютера.

```
150 LET AX=(PX-3)/2
160 LET AY=(PY-3)/2
170 DIM C$(8)
180 LET C$(1)=CHR$(5) : REM БЕЛЫЙ
190 LET C$(2)=CHR$(158) : REM ЖЕЛТЫЙ
200 LET C$(3)=CHR$(159) : REM БИРЮЗОВЫЙ
210 LET C$(4)=CHR$(30) : REM ФИОЛЕТОВЫЙ
```

При желании порядок цветов можно изменить.

```
220 LET C$(5)=CHR$(156) : REM ЗЕЛЕНЫЙ
230 LET C$(6)=CHR$(31) : REM ГОЛУБОЙ
240 LET C$(7)=CHR$(28) : REM КРАСНЫЙ
250 LET C$(8)=CHR$(144) : REM ЧЕРНЫЙ
260 LET P$=CHR$(18)+" "CHR$(146) : REM
270 REM LET P$="■"
```

Используйте те коды, которыми в вашей машине обозначены соответствующие цвета. Для компьютера ZX Spectrum нужно строку 170 заменить на DIM C$(8,2). Если ваш дисплей не дает цветного изображения, можно воспользоваться какими-нибудь символами. Ниже приведены два возможных варианта:

```
180 LET C$(1)=" "
190 LET C$(2)=" "
200 LET C$(3)="▒"
210 LET C$(4)="▒"
220 LET C$(5)="▓"
230 LET C$(6)="▓"
240 LET C$(7)="■"
250 LET C$(8)="■"
260 LET P$=""
```

```
180 LET C$(1)=" "
190 LET C$(2)=":"
200 LET C$(3)="."
210 LET C$(4)="="
220 LET C$(5)="+"
230 LET C$(6)="%"
240 LET C$(7)="X"
250 LET C$(8)="*"
260 LET P$=""
```

Символом P$ обозначен инверсный квадрат, т. е. полностью закрашенный квадрат. Если такой возможности нет, используйте что-нибудь еще.

```
300 REM XXXXXXX ВЫБОР ФУНКЦИИ XXXXXXX
310 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
320 PRINT "КАКУЮ ФУНКЦИЮ ЖЕЛАЕТЕ? УКАЖИТЕ НОМЕР"
330 PRINT
340 PRINT "1. Z=COS(X*X)"
350 PRINT
360 PRINT "2. Z=COS(X*EXP(-Y/5))"
370 PRINT
380 PRINT "3. Z="
390 PRINT " COS(X*Y/SQR(X*X+Y*Y))"
400 PRINT "4.Z=COS(Y/(ABS(X)+.5))"
410 PRINT "5. Z=COS((X+Y)/"
420 PRINT "    (LOG(X*X+Y↑4+.5)))"
430 PRINT "6. Z=SIN((X+Y)/"
440 PRINT "    LOG(ABS(X*Y)+1.1))"
450 PRINT "7. Z=SIN(X)*COS(Y)"
460 PRINT
470 INPUT F
480 PRINT
490 PRINT "МАСШТАБ - ЧЕМ БОЛЬШЕ ЧИСЛО,ТЕМ БОЛЬШЕ"
500 PRINT "МАСШТАБ ";
510 INPUT S
600 REM XXXXXXXXXXXXXXXX ПОСТРОЕНИЕ ГРАФИКОВ XXXXXXXXXXXXXXXXXXXXXXXXX
610 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
620 FOR J=-AY TO AY
630 PRINT " ";
```

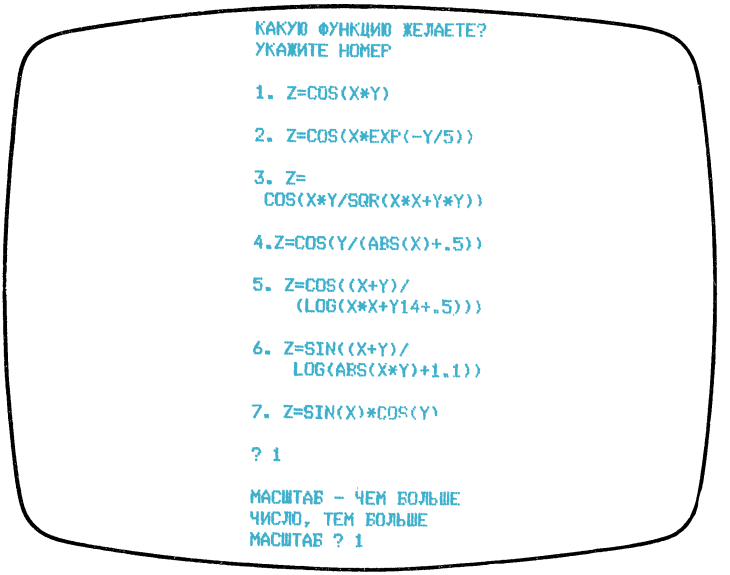
Используется семь различных функций.

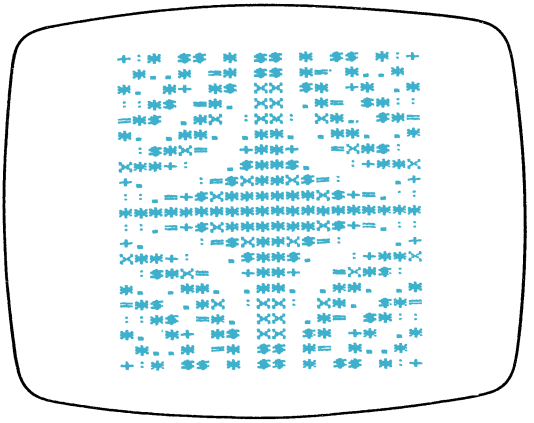
Значение S определяет области изменения значений X и Y, внутри которых производится построение. Попробуйте значения 10, 1, 0.1 и т. д..

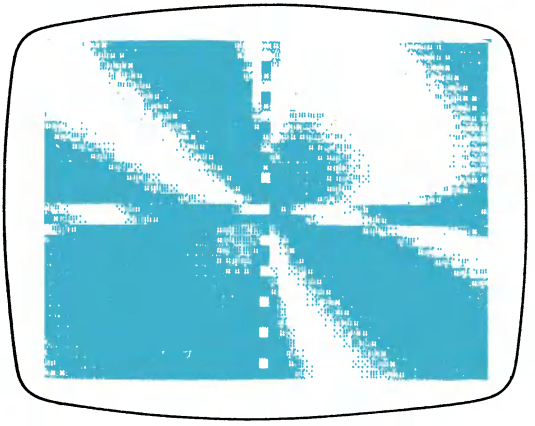
Основная программа построения графиков.

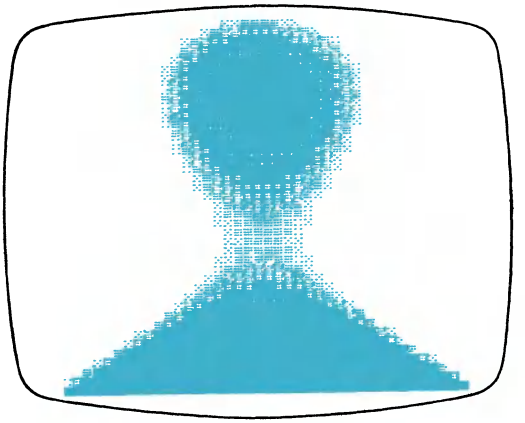
```
640 FOR I=-AX TO AX
650 LET X=I*S/2
660 LET Y=J*S/2
670 LET Z=1
680 IF F=1 THEN LET Z=COS(X*Y)
690 IF F=2 THEN LET Z=COS(X*EXP(-Y/5))
700 IF F=3 AND X*X+Y*Y()0 THEN LET Z=COS(X*Y/SQR(X*X+Y*Y))
710 IF F=4 THEN LET Z=COS(Y/(ABS(X)+.5))
720 IF F=5 THEN LET Z=COS((X+Y)/(LOG(X*X+Y↑4+.5)))
730 IF F=6 THEN LET Z=SIN((X+Y)/LOG(ABS(X*Y)+1.1))
740 IF F=7 THEN LET Z=SIN(X)*COS(Y)
750 LET W=INT(4*Z+5)
760 IF W(1 THEN LET W=1
770 IF W)8 THEN LET W=8
780 PRINT C$(W)+P$;
790 NEXT I
800 PRINT " ";
810 NEXT J
820 PRINT HM$;C$(6); : REM PRINT AT 0,0
900 REM XXXXXXXXXXXXXXXX ОКОНЧАНИЕ И ПОВТОРЕНИЕ XXXXXXXXXXXXXXXXXXXX
910 GET G$ : REM LET G$=INKEY$
920 IF G$=" " THEN GOTO 910
930 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
940 GET G$ : REM LET G$=INKEY$
950 IF G$()"Y" AND G$()"N" THEN GOTO 940
960 IF G$="Y" THEN GOTO 310
970 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
```

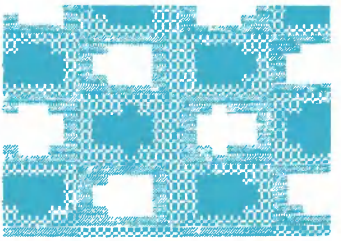
Прежде чем продолжить свою работу, программа должна подождать, пока не нажмут одну из клавиш.

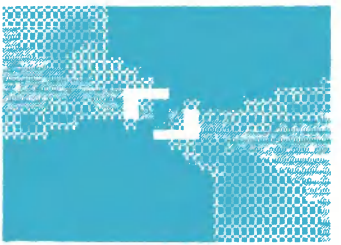












О дифференциальных уравнениях

Все мы, наверное, бросали камень в колодец, чтобы определить его глубину. Узнать (хотя бы приблизительно) расстояние S, которое пролетит камень за T секунд, нам помогало уравнение $S = G*T^2/2$. Здесь G — константа (ускорение свободного падения), равная 32, если мы хотим получить ответ в футах, или 9.8, если ответ будет выражен в метрах. Так, например, после двух секунд свободного падения камень пролетит $32*2*2/2 = 64$ фута.

В уравнение, описывающее то или иное физическое явление, часто входит скорость изменения (производная) по одной переменной. В только что разобранном примере это была скорость изменения расстояния, прямо пропорциональная изменению времени. Это соотношение записывается следующим образом:

$$\frac{dS}{dT} = G*T,$$

или же

$$S' = G*T,$$

где выражение в левой части обоих соотношений означает производную функции S по T (в данном случае это скорость падения камня).

Если вы немного знакомы с математическим анализом (если нет — не огорчайтесь), вам будет понятно, как связаны между собой следующие уравнения:

$$S = G*T^2/2,$$
$$S' = G*T, \quad (S = 0, \text{ если } T = 0).$$

Второе соотношение называется *дифференциальным уравнением*, а условия в скобках — начальными условиями. Первое соотношение — это *решение* данного дифференциального уравнения. Оно так называется потому, что производная от $G*T^2/2$ есть в точности $G*T$. Приведенный ниже пример тесно связан с предыдущим (здесь V обозначает скорость):

$$V = G*T,$$
$$V' = G \quad (V = 0, \text{ если } T = 0).$$

Второе соотношение — это дифференциальное уравнение, а первое — его решение. Данное дифференциальное уравнение утверждает, что производная от скорости — величина постоянная, т. е. что ускорение предмета имеет постоянное значение.

В курсе теории дифференциальных уравнений обычно рассматриваются такие дифференциальные уравнения, решения которых можно выразить явными математическими формулами, т. е. представить в явном виде. Однако уравнения, встречающиеся на практике, обычно слишком сложны, чтобы их можно было так решать. Взгляните, например, на следующее уравнение:

$$Y' = 1 - Y + K*Y^3*SIN(T).$$

Не существует формулы, которая выражала бы Y через T при $K \neq 0$, хотя для $K = 0$ такая формула имеется. Из этого вовсе не следует, что уравнение не имеет решения, просто не существует решения в явном виде. Однако, используя компьютер, можно найти приближенное решение. Мы получим *численное решение*, а зачастую ничего другого и не требуется.

Вернемся снова к примеру с падающим камнем. Пусть V_N означает скорость камня по прошествии N секунд. Из дифференциального уравнения видно, что производная скорости есть G. Таким образом, каждую секунду мы ожидаем изменения скорости на G. Поэтому записываем такое уравнение:

$$V_{N+1} = V_N + G.$$

При заданном значении V_0 можно многократно применить это уравнение для определения V_N при любых значениях N.

А что можно сказать о расстоянии S? Расстояние, пройденное за одну секунду, равно средней скорости. Что происходит в интервале между (N − 1)-й и N-й секундами? Поскольку в нашем случае ускорение — величина постоянная, средняя скорость равна скорости в средней точке этого интервала, т. е. в момент (N − 1/2) с. Так как скорость каждую секунду изменяется на величину G, то за 1/2 с она возрастет на G/2. Таким образом, средняя скорость равна $V_{N-1} + G/2$, или же $V_N - G/2$. Обозначим символом S_N расстояние, которое пролетит падающий предмет за Nc. Это расстояние равно расстоянию, пройденному за (N − 1) с, плюс расстояние, пройденное за следующую секунду, что можно записать как

$$S_N = S_{N-1} + V_N - G/2.$$

Используя это уравнение, вы получите на вашем микрокомпьютере циклическую итеративную процедуру для вычисления скорости и расстояния, которое пролетает падающий камень.

В качестве иллюстрации изложенных выше соображений приводится программа ПАДАЮЩИЙ КАМЕНЬ, в которой вычисляются скорость и пройденное падающим камнем расстояние за 10 с.

```
10 REM                          ********************
20 REM                          *                  *
30 REM                          * ПАДАЮЩИЙ КАМЕНЬ *
40 REM                          *                  *
50 REM                          ********************
60 REM
70 REM
100 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
110 PRINT "  ПАДАЮЩИЙ КАМЕНЬ"
120 PRINT
130 LET V=0 : REM НАЧАЛЬНАЯ СКОРОСТЬ
140 LET S=0 : REM НАЧАЛЬНОЕ РАССТОЯНИЕ
150 PRINT "ВРЕМЯ  СКОР. РАССТ."
160 PRINT
170 FOR T=0 TO 10
180 PRINT TAB(3-LEN(STR$(T)));T;
190 PRINT TAB(9-LEN(STR$(V)));V;
200 PRINT TAB(16-LEN(STR$(S)));S
210 LET V=V+32
220 LET S=S+V-16
230 NEXT T
```

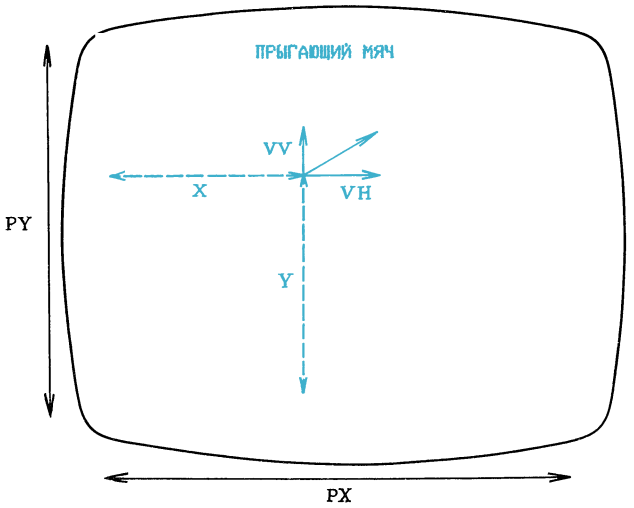
Общие замечания по программированию см. в приложении.

Что произойдет, если в программе ПАДАЮЩИЙ КАМЕНЬ строки 210 и 220 поменять местами?

```
  ПАДАЮЩИЙ КАМЕНЬ

ВРЕМЯ  СКОР. РАССТ.

 0     0       0
 1    32      16
 2    64      64
 3    96     144
 4   128     256
 5   160     400
 6   192     576
 7   224     784
 8   256    1024
 9   288    1296
10   320    1600
```

Такой метод решения дифференциальных уравнений называется методом Эйлера — Коши, хотя, строго говоря, мы использовали некую его модификацию для решения задачи определения расстояния. Вообще говоря, метод Эйлера — Коши дает приближенное решение дифференциального уравнения, тогда как в программе ПАДАЮЩИЙ КАМЕНЬ этот метод дает точное решение. Леонард Эйлер (1707—1783 г.) был швейцарским математиком, внесшим значительный вклад во многие разделы математики; долгое время он работал в России. Он изучал почти все математические дисциплины и опубликовал больше 1000 статей. Барон Огюстен Луи Коши (1789—1857 г.) был французским математиком и, наверное, самым плодовитым математиком: иногда он публиковал по статье в день. Коши развил строгий аппарат математического анализа, используя так называемый подход «дельта-эпсилон».

В программе ПРЫГАЮЩИЙ МЯЧ только что изложенные идеи получают дальнейшее развитие. Мяч выбрасывается из левого верхнего угла экрана и падает на основание, отскакивая от сторон и от основания. Каждый раз, когда мяч отскакивает от стороны, он теряет 80% своей скорости, и 70% — когда отскакивает от основания. Так что рано или поздно мяч остановится. Задается (INPUT) начальная (горизонтальная) скорость VH. Вертикальная скорость VV определяется так же, как в программе ПАДАЮЩИЙ КАМЕНЬ. В программе предусмотрена возможность вывода на экран траектории мяча. Обратите внимание на ее вид.



На рисунке наглядно поясняется смысл используемых в программе переменных.

```
 10 REM                      ******************
 20 REM                      *                  *
 30 REM                      * ПРЫГАЮЩИЙ МЯЧ   *
 40 REM                      *                  *
 50 REM                      ******************
 60 REM
 70 REM
100 REM XXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
110 LET P=8164+(PEEK(44))=18)*3584 : REM НАЧАЛО POKE
120 POKE 36879,170 : REM ЦВЕТНОЕ ИЗОБРАЖЕНИЕ
130 LET CS$=CHR$(147) : REM КОД ЦВЕТНОГО ИЗОБРАЖЕНИЯ
140 LET HM$=CHR$(19) : REM КОД ИСХОДНОГО ПОЛОЖЕНИЯ
150 LET PX=22 : REM КОЛИЧЕСТВО ТОЧЕК ПО ГОР
160 LET PY=23 : REM КОЛИЧЕСТВО ТОЧЕК ПО ВЕРТ
170 LET PX=PX-1
180 LET PY=PY-1
190 LET H=0.1
```

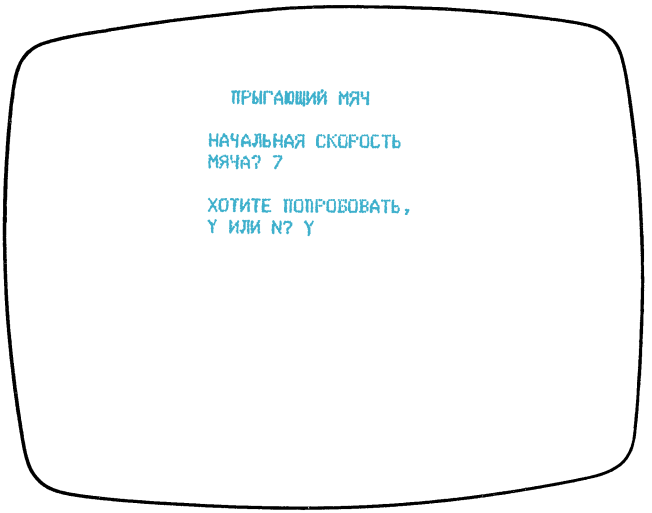
PX и PY — количества точек изображения по горизонтали и вертикали при использовании команд PRINT, PLOT или POKE. Лучше иметь низкую разрешающую способность, так как иначе работа программы замедляется.

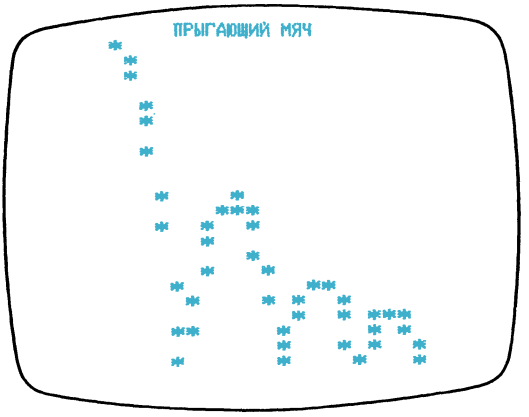
Множитель H помогает регулировать скорость изображения. Чем больше значение H, тем быстрее меняется изображение.

```
200 REM XXXXXXXXXXXXX НАЧАЛЬНОЕ ПОЛОЖЕНИЕ МЯЧА XXXXXXXXXXXXXXXXXXXX
210 LET X=0
220 LET Y=PY
230 LET U=0
240 LET V=PY
250 LET VV=0
260 LET N=0

300 REM XXXXXXXXXXXXXXXXXX ВВОД ДАННЫХ XXXXXXXXXXXXXXXXXXXXXXXXXXX
310 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
320 PRINT "  ПРЫГАЮЩИЙ МЯЧ"
330 PRINT
340 PRINT "НАЧАЛЬНАЯ СКОРОСТЬ"
350 PRINT "МЯЧА";
360 INPUT VH
370 PRINT
380 LET T=0
390 PRINT "ХОТИТЕ ПОПРОВОВАТЬ,"
400 PRINT "Y ИЛИ N ";
410 INPUT D$
420 IF D$="Y" THEN LET T=1
430 PRINT CS$; : REM ОЧИСТИТЬ ЭКРАН
440 PRINT "  ПРЫГАЮЩИЙ МЯЧ"
```

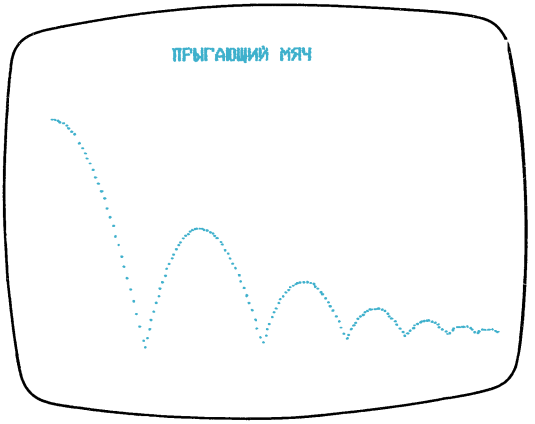
```
500 REM XXXXXXXXXXXXXXXX ВЫЧИСЛИТЕЛЬНЫЙ ЦИКЛ XXXXXXXXXXXXXXXXXXX
510 LET X=X+VH*H
520 LET VV=VV+32*H
530 LET Y=Y-VV*H+16*H*H

540 IF Y<=0 THEN GOSUB 710
550 IF X>=PX THEN GOSUB 810
560 IF X<=0 THEN GOSUB 910
570 IF T=0 THEN GOSUB 1110 : REM UNPLOT U,V
580 GOSUB 1310 : REM SOUND OFF
590 LET U=INT(X)
600 LET V=INT(Y)
610 GOSUB 1010 : REM PLOT U,V
620 IF N<16 THEN GOTO 510
630 PRINT HM$; : REM PRINT AT 0,0
640 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
650 GET G$ : REM LET G$=INKEY$
660 IF G$<>"Y" AND G$<>"N" THEN GOTO 650
670 IF G$="Y" THEN GOTO 210
680 END : REM STOP

700 REM XXXXXXXXXXXXXXXX ОТСКОК ОТ ОСНОВАНИЯ XXXXXXXXXXXXXXXXXX
710 LET Y=0
720 LET VV=-0.7*VV
730 LET N=N+1
740 GOSUB 1210 : REM SOUND ON
750 RETURN

800 REM XXXXXXXXXXXXXX ОТСКОК ОТ ПРАВОЙ СТОРОНЫ XXXXXXXXXXXXXXXX
810 LET X=PX
820 LET VH=-0.8*VH
830 GOSUB 1210 : REM SOUND ON
840 RETURN

900 REM XXXXXXXXXXXXXX ОТСКОК ОТ ЛЕВОЙ СТОРОНЫ XXXXXXXXXXXXXXXXX
910 LET X=0
920 LET VH=-0.8*VH
930 GOSUB 1210 : REM SOUND ON
940 RETURN

1000 REM XXXXXXXXXXX ПОСТРОЕНИЕ ПОСРЕДСТВОМ POKE XXXXXXXXXXXXXXX
1010 POKE P+U-22*V,42
1020 RETURN

1100 REM XXXXXXX ОТМЕНА ПОСТРОЕНИЯ ПОСРЕДСТВОМ POKE XXXXXXXXXXXXX
1110 POKE P+U-22*V,32
1120 RETURN

1200 REM XXXXXXXXXXXXXXX ВКЛЮЧЕНИЕ ЗВУКА XXXXXXXXXXXXXXXXXXXXX
1210 POKE 36878,15
1215 POKE 36876,215
1220 RETURN

1300 REM XXXXXXXXXXXXXXX ВЫКЛЮЧЕНИЕ ЗВУКА XXXXXXXXXXXXXXXXXXXXX
1310 POKE 36876,0
1315 POKE 36878,0
1320 RETURN
```

Если есть такая возможность, дайте свои звуковые эффекты.





Низкая разрешающая способность



Высокая разрешающая способность

В реальной жизни всякий падающий или движущийся предмет встречает сопротивление воздуха. Производимый им эффект приблизительно пропорционален скорости движущегося предмета. Таким образом, с учетом сопротивления воздуха, наше уравнение движения примет вид

$$V' = G - K*V,$$

где K — некоторая постоянная. Скорость в момент времени N есть скорость в момент N − 1 плюс ожидаемое изменение скорости (ее производная V′) в момент N − 1. Таким образом, метод Эйлера — Коши в нашем случае дает формулу

$$V_N = V_{N-1} + G - K*V_{N-1}.$$

Для игровых целей это достаточно хорошее приближение.

Аналогичные уравнения можно использовать и для движущегося объекта (ракеты или автомобиля), приводимого в действие некой внутренней силой (двигателем). Для этого заменим G на TH — ускорение, или тягу объекта. Величину ускорения обычно можно регулировать при помощи некоторого устройства (акселератора или педали). Уравнение примет вид

$$V_N = V_{N-1} + TH - K*V_{N-1}.$$

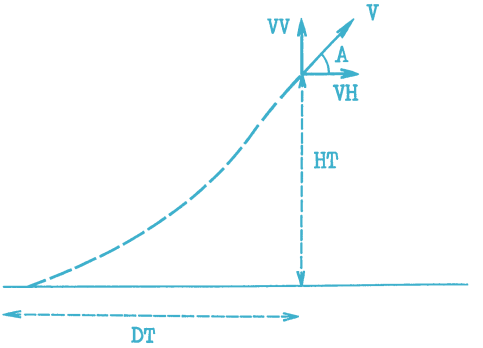
Заметим, что как только скорость достигает уровня TH/K, она больше не меняется. В частности, объект, движущийся со скоростью, превышающей TH/K, будет замедляться. Таким образом, меняя значение TH, можно менять скорость.

Изложенные идеи использованы в простой игре ПОЛЕТ, в которой моделируются взлет и посадка реактивного самолета. Вы должны регулировать угол взлета (или приземления), который вы будете задавать самолету. В вашем распоряжении тормозное устройство, позволяющее менять значение TH и, следовательно, скорости самолета. Прежде чем вы начнете приземляться, нужно набрать высоту 500 м или выше. Если достигнете уровня земли с вертикальной скоростью, превышающей −10, то автоматически будет включен тормоз и вы благополучно приземлитесь. В противном случае вы разобьетесь. Вы разобьетесь и в том случае, если горизонтальная скорость станет отрицательной.

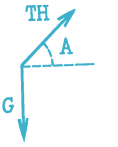
Скорость в игре ПОЛЕТ задается в единицах м/с. Умножив ее на 2.2, вы получите мили/час.

В программе ПОЛЕТ используются несколько более сложные уравнения, чем выписанные выше. Кроме того, учтено сопротивление среды, пропорциональное квадрату скорости.

Если вы никогда не изучали процесс приземления самолета, то с удивлением обнаружите, что ПОЛЕТ вас этому обучит.



ПОЛЕТ: расстояния и скорости



ПОЛЕТ: направления сил

```
10 REM                          **********
20 REM                          *        *
30 REM                          * ПОЛЕТ *
40 REM                          *        *
50 REM                          **********
60 REM
70 REM
100 REM %%%%%%%%%%%%%%% НАЧАЛЬНАЯ УСТАНОВКА %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110 LET CS=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET HM=CHR$(19) : REM КОД ИСХОДНОГО ПОЛОЖЕНИЯ
130 LET B=CHR$(31) : REM ГОЛУБОЙ
140 LET R=CHR$(28) : REM КРАСНЫЙ
150 LET M=CHR$(156) : REM ФИОЛЕТОВЫЙ
160 LET G=CHR$(30) : REM ЗЕЛЁНЫЙ
170 LET C=CHR$(159) : REM БИРЮЗОВЫЙ
180 LET Y=CHR$(158) : REM ЖЕЛТЫЙ
```

Если ваш дисплей дает цветное изображение, используйте соответствующие коды, в противном случае исключите B$, R$ и т. д.

```
190 LET VV=0 : REM ВЕРТ. СКОРОСТЬ В М/С
200 LET VH=0.01 : REM ГОРИЗ. СКОРОСТЬ В М/С
210 LET HT=0.01 : REM ВЫСОТА В М
220 LET D1=0 : REM РАССТОЯНИЕ В КМ
230 LET A=0 : REM УГОЛ ВЗЛЕТА В ГРАД
240 LET TH=0 : REM ТЯГА
250 LET L=0 : REM МИНИМАЛЬНАЯ ВЫСОТА НЕ НАБРАНА
260 LET SW=0 : REM ДВИГАТЕЛЬ НЕ ВКЛЮЧЕН
300 REM XXXXXXXXXXXXX ПЕРВОНАЧАЛЬНЫЙ ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXXXXX
310 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
320 PRINT
330 PRINT R$+"ЕСЛИ ГОТОВЫ ВКЛЮЧИТЬ"
340 PRINT R$+"ДВИГАТЕЛЬ, НАЖМИТЕ Y"
350 GOTO 400
360 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
370 PRINT
380 LET SW=1
390 PRINT B$+"      ВЗЛЕТАЕТ"
400 PRINT HM$ : REM PRINT AT 0,0
410 FOR I=1 TO 17
420 PRINT
430 NEXT I
440 PRINT B$+"ПОКАЗАНИЯ ПРИБОРОВ"
450 PRINT G$+"ТЯГА    Z:-  C:+"
460 PRINT B$+"УГОЛ    B:-  M:+"
470 GOSUB
500 REM XXXXXXXXXXXXXXX ПОКАЗАНИЯ ПРИБОРОВ XXXXXXXXXXXXXXXXXXXXXXXXXXXX
510 GET H$ : REM LET H$=INKEY$
520 IF H$="Z" THEN LET TH=TH-1
530 IF H$="C" THEN LET TH=TH+1
540 IF TH>15 THEN LET TH=15
550 IF H$="B" THEN LET A=A-1
560 IF H$="M" THEN LET A=A+1
570 IF H$="Y" THEN GOTO 360
580 IF SW=0 THEN GOTO 470
600 REM XXXXXXXXXXXXXXXXXX ВЫЧИСЛЕНИЯ XXXXXXXXXXXXXXXXXXXXXXXXXXXX
610 LET XX=COS(PI*A/180)*TH-(1+VH/200)*VH/200
620 LET YY=-9.8+SIN(PI*A/180)*TH-(1+VV/100)*VV/100+(6-VH/100)*VH/100
630 LET VH=INT((VH+XX)*100)/100
640 LET VV=INT((VV+YY)*100)/100
650 LET HT=INT((HT+VV-YY/2)*100)/100
660 IF L=0 AND HT>499 THEN GOSUB 1410
670 IF HT<0 THEN LET HT=0
680 IF HT=0 AND VV>-10 THEN LET VV=0
690 IF HT=0 AND VV=0 AND L=1 THEN GOTO 910
700 LET DT=INT((1000*DT+VH-XX/2))/1000
710 GOSUB 1210
720 IF HT=0 AND VV<=-10 THEN GOTO 810
730 IF VH<=0 THEN GOTO 810
740 GOTO 510
800 REM XXXXXXXXXXXXXXXXXX КРУШЕНИЕ XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
810 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
820 PRINT
830 PRINT R$+"      КРУШЕНИЕ"
840 GOSUB 1210
```

Перед взлетом можете отрегулировать величину тяги и угол взлета. Чтобы взлететь, нажмите Y.

Вы можете захотеть удалить строку 680, а в строку 690 внести изменение: VV > −10. Обратите внимание на то, как это отразится на программе.

```
850 REM ЗДЕСЬ ЗВУКОВЫЕ ЭФФЕКТЫ
851 POKE 36877,220
852 FOR I=15 TO 0 STEP -1
853 POKE 36878,I
854 FOR J=1 TO 200 NEXT J
855 NEXT I
856 POKE 36877,0
860 GOTO 1110
900 REM XXXXXXXXXXXXXXXXXX ПРИЗЕМЛЕНИЕ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
910 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
920 LET TH=0
930 LET A=0
940 LET VV=0
950 LET VH=0
960 PRINT
970 PRINT G$+"    ПРИЗЕМЛЕНИЕ"
980 PRINT R$+"   ПРОШЛО УСПЕШНО"
990 GOSUB 1210
1000 FOR J=1 TO 3
1010 GOSUB 1510 : REM ЗВУКОВЫЕ ЭФФЕКТЫ
1020 NEXT J
1100 REM XXXXXXXXXXXXXXXXX ПОВТОРИТЬ? XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1110 GET H$ : REM LET H$=INKEY$
1120 IF H$()"" THEN GOTO 1110
1130 PRINT
1140 PRINT B$="ПОВТОРИТЬ? Y ИЛИ N"
1150 GET H$ : REM LET H$=INKEY$
1160 IF H$()"Y" AND H$()"N" THEN GOTO 1150
1170 IF H$="Y" THEN GOTO 190
1180 END : REM STOP
1200 REM XXXXXXXXXXXXXXXXXX ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1210 PRINT HM$; : REM PRINT AT 0,0
1220 PRINT R$+"     ПОЛЕТ"
1230 PRINT
1240 PRINT
1250 PRINT
1260 PRINT
1270 PRINT M$+"ВЕРТ.СКОР.=";STR$(VV);"   "
1280 PRINT
1290 PRINT R$+"ВЫСОТА=";STR$(HT);"    "
1300 PRINT
1310 PRINT G$+"ТЯГА=";STR$(TH);" "
1320 PRINT
1330 PRINT B$+"УГОЛ=";STR$(A);" "
1340 PRINT
1350 PRINT C$+"ГОРИЗ.СКОР.=";STR$(VH);"   "
1360 PRINT
1370 PRINT Y$="РАССТОЯНИЕ=";STR$(DT);"  "
1380 RETURN
1400 REM XXXXXXXXXXXXXX ВЫСОТА 500 НАБРАНА? XXXXXXXXXXXXXXXXXXXXXXXX
1410 LET L=1
1420 PRINT HM$ : REM PRINT AT 0,0
1430 PRINT
1440 PRINT R$+"  МОЖНО ПРИЗЕМЛЯТЬСЯ"
1450 GOSUB 1510 : REM ЗВУКОВЫЕ ЭФФЕКТЫ
1460 RETURN
1500 REM XXXXXXXXXXXXXXX ЗВУКОВЫЕ ЭФФЕКТЫ XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1510 POKE 36878,15
1520 POKE 36876,200
1530 FOR I=1 TO 100
```

Если есть такая возможность, дайте свои звуковые эффекты.

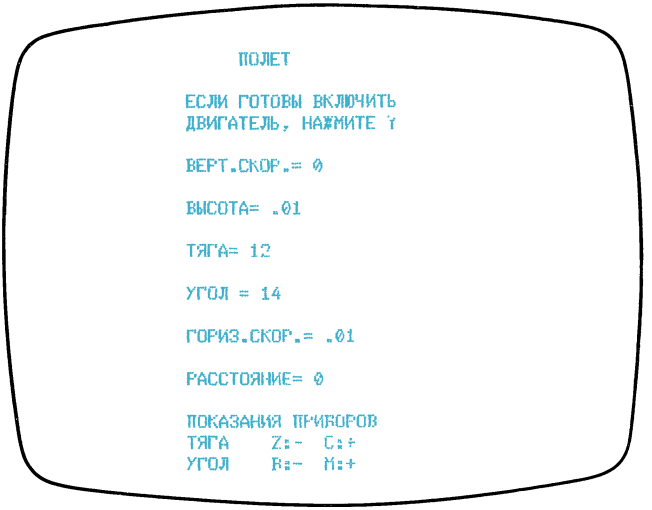
Если все сделано правильно, автоматически включается тормоз и вы благополучно приземляетесь.

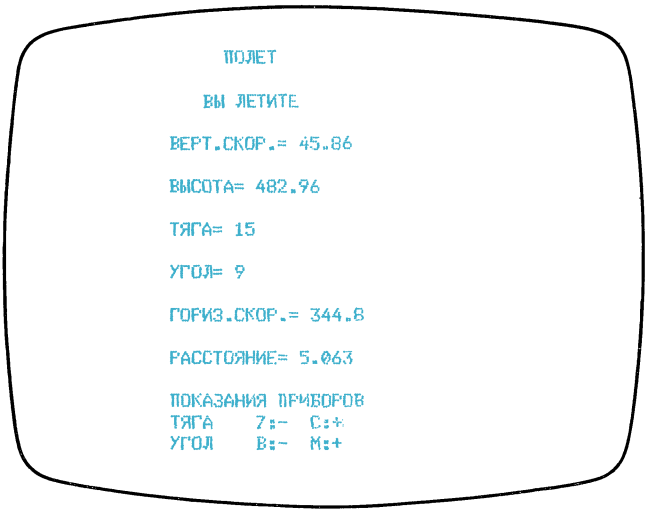
Прежде чем начать приземлятся, вы должны набрать высоту 500 м.

Если есть такая возможность, дайте свои звуковые эффекты.

```
1540 NEXT I
1550 POKE 36876,0
1560 POKE 36878,0
1570 FOR I=1 TO 100
1580 NEXT I
1590 RETURN
```

Дополнительные замечания см. в приложении.





```
     ПОЛЕТ

МОЖНО ПРИЗЕМЛЯТЬСЯ

ВЕРТ.СКОР.=-9.21

ВЫСОТА= 253.3

ТЯГА= 5

УГОЛ= 13

ГОРИЗ.СКОР.= 346.93

РАССТОЯНИЕ= 79.586

ПОКАЗАНИЯ ПРИБОРОВ
ТЯГА    Z:-  C:+
УГОЛ    B:-  M:+
```

```
     ПОЛЕТ

   ПРИЗЕМЛЕНИЕ
  ПРОШЛО УСПЕШНО

 ВЕРТ.СКОР.= 0

 ВЫСОТА= 0

 ТЯГА= 0

 УГОЛ= 0

 ГОРИЗ.СКОР.= 0

 РАССТОЯНИЕ= 89.334

 ПОКАЗАНИЯ ПРИБОРОВ
 ТЯГА    Z:-  C:+
 УГОЛ    B:-  M:+
```

Программу ПОЛЕТ можно усовершенствовать. Для этого существует целый ряд приемов. Наигравшись вдоволь, добавьте к программе что-нибудь, интересное именно вам. Можно, скажем, принять во внимание горючее. Ниже приводятся необходимые для этого команды:

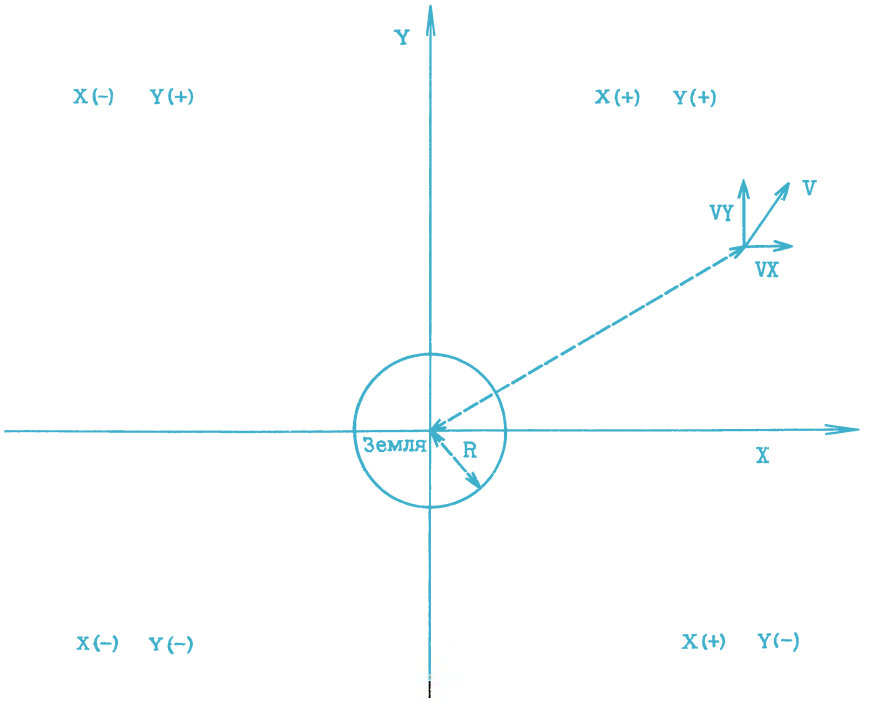
```
270 LET FUEL=3000
280 REM 3000 ЛЕГКОЕ, 2000 СРЕДНЕЕ, 1000 ТЯЖЕЛОЕ
732 LET FUEL=FUEL-ABS(TH)
734 IF FUEL<0 THEN LET FUEL=0
736 IF FUEL=0 THEN LET TH=0
738 IF FUEL=0 THEN GOTO 610
1372 PRINT
1374 PRINT "ГОРЮЧЕЕ=";STR$(FUEL);"   "
```

Если мы запускаем ракету вертикально к поверхности Земли, то следует принимать во внимание уже другие факторы. Например, ускорение свободного падения G при удалении от Земли меняется. На самом деле приближенное значение этой величины можно получить по формуле $G0/(1 + HT/R)^2$, где G0 — значение ускорения свободного падения у поверхности земного шара, R — его радиус и HT — расстояние до земной поверхности. Удобно воспользоваться такими обозначениями: G1 = R*R*G0 и DT = R + HT. Тогда G = G1/(DT*DT). Для малых значений HT значение DT есть приблизительно R и G почти совпадает с G0.

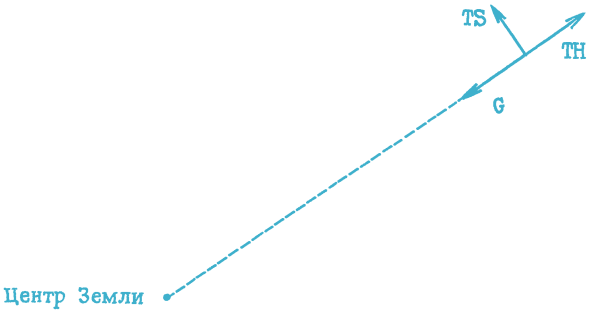
Другой немаловажный фактор в случае ракеты — это то, что при удалении от поверхности Земли воздух становится более разреженным. Следовательно, сопротивление воздуха меняется. В приводимой ниже программе это уточнение, однако, не делается.

В программе ОРБИТА с поверхности Земли запускается ракета или летающее блюдце. В вашем распоряжении имеется два рычага управления, под названием вертикальная и горизонтальная тяга. Вертикальная тяга — это сила, направленная от центра (или к центру) Земли. Под ее воздействием ракета или поднимается, или опускается. Горизонтальная тяга — сила, направленная перпендикулярно вертикальной тяге. Под ее воздействием ракета смещается в ту или иную сторону.

При помощи программы ОРБИТА и вертикальной и горизонтальной тяг выведите свою ракету на околоземную орбиту. В вашем распоряжении ограниченный (хотя и довольно большой) запас горючего, так что пользуйтесь рычагами управления аккуратно. Если горючее иссякло, а вес вашей ракеты и ее скорость не те, какие требуются, то вы либо унесетесь в космическое пространство, либо рухнете на землю. Последнее — расплата за чрезмерную экономию горючего. Кроме того, не забирайтесь слишком высоко, иначе ваша околоземная орбита окажется очень протяженной.



ОРБИТА: расстояния и скорости



ОРБИТА: направление сил

```
10 REM                          **********
20 REM                          *        *
30 REM                          * ОРБИТА *
40 REM                          *        *
50 REM                          **********
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS$=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET HM$=CHR$(19) : REM КОД ИСХОДНОГО ПОЛОЖЕНИЯ
130 LET B$=CHR$(31) : REM ГОЛУБОЙ
132 LET R$=CHR$(28) : REM КРАСНЫЙ
134 LET M$=CHR$(156) : REM ФИОЛЕТОВЫЙ
136 LET G$=CHR$(30) : REM ЗЕЛЕНЫЙ
138 LET C$=CHR$(159) : REM БИРЮЗОВЫЙ
```

Если на вашей машине есть цветное изображение, укажите соответствующие коды; в противном случае B$, R$ и т. д. исключите.

```
140 LET VX=0 : REM СКОРОСТЬ X, КМ/С
150 LET YY=0 : REM СКОРОСТЬ Y, КМ/С
160 LET V=0 : REM СКОРОСТЬ, КМ/МИН
170 LET HT=0.1 : REM ВЫСОТА, КМ
180 LET TH=0 : REM ВЕРТИКАЛЬНАЯ ТЯГА
190 LET TS=0 : REM ГОРИЗОНТАЛЬНАЯ ТЯГА
200 LET SW=0 : REM ДВИГАТЕЛЬ НЕ ВКЛЮЧЕН
210 LET R=6371 : REM РАДИУС ЗЕМЛИ
220 LET DT=R+HT : REM РАССТОЯНИЕ ОТ ЦЕНТРА ЗЕМЛИ
230 LET X=R+HT
240 LET Y=0
250 LET X$=STR$(X)
260 LET Y$=STR$(Y)
270 LET G1=R*R*9.81/1000 : REM УСКОРЕНИЕ СВОБОДНОГО ПАДЕНИЯ
280 LET G=G1/(DT*DT)
290 LET FUEL=10000
```

```
300 REM XXXXXXXXXXXXXXXX ПЕРВОНАЧАЛЬНЫЙ ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXX
310 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
320 PRINT
330 PRINT R$+"ЕСЛИ ГОТОВЫ ВКЛЮЧИТЬ
340 PRINT R$+"ДВИГАТЕЛЬ, НАЖМИТЕ Y"
350 GOTO 400
360 PRINT CS$: REM ОЧИСТИТЬ ЭКРАН
370 PRINT
380 LET SW=1
390 PRINT B$+"      ВЫ ЛЕТИТЕ"
400 PRINT HM$ : REM PRINT AT 0,0
410 FOR I=1 TO 18
420 PRINT
430 NEXT I
440 PRINT B$+"ПОКАЗАНИЯ ПРИБОРОВ"
450 PRINT G$+"ВЕРТ.ТЯГА  Z:-  C:+"
460 PRINT B$+"ГОРИЗ.ТЯГА B:-  M:+"
470 GOSUB 1010
```

Установите ваши контрольные приборы, прежде чем взлететь.

```
500 REM XXXXXXXXXXXXXXXXXX ПОКАЗАНИЯ ПРИБОРОВ XXXXXXXXXXXXXXXXXXXXXXXX
510 GET H$ : REM LET H$=INKEY$
520 IF H$="Z" THEN LET TH=TH-5
530 IF H$="C" THEN LET TH=TH+5
540 IF H$="B" THEN LET TS=TS-5
550 IF H$="M" THEN LET TS=TS+5
560 IF H$="Y" THEN GOTO 360
570 IF SW=0 THEN GOTO 470
600 REM XXXXXXXXXXXXXXXXXXXXXXXX СЧЕТ XXXXXXXXXXXXXXXXXXXXXXXXXXXX
610 LET FUEL=FUEL-ABS(TH)-ABS(TS)
620 IF FUEL<=0 THEN LET TS=0
630 IF FUEL<=0 THEN LET TH=0
640 IF FUEL<0 THEN LET FUEL=0
650 LET XX=(TH/1000-G)*X/DT+TS/1000*Y/DT
660 LET YY=(TH/1000-G)*Y/DT+TS/1000*X/DT
670 LET VX=VX+XX
680 LET VY=VY+YY
690 LET V=INT(SQR(VX*VX+VY*VY)*60)
700 LET X=X+VX-XX/2
710 LET X$=STR$(INT(X*10)/10)
720 LET Y=Y+VY-YY/2
730 LET Y$=STR$(INT(Y*10)/10)
740 LET DT=SQR(X*X+Y*Y)
750 LET HT=INT((DT-R)*10)/10
760 LET G=G1/(DT*DT)
770 GOSUB 1010
780 IF FUEL=0 AND HT>=0 THEN GOTO 650
790 IF HT>=0 THEN GOTO 510
800 REM XXXXXXXXXXXXXXXXXXXXXXXX КРУШЕНИЕ XXXXXXXXXXXXXXXXXXXXXXXXXX
810 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
820 PRINT
830 PRINT R$+"     КРУШЕНИЕ"
840 GOSUB 1010
850 REM ЗДЕСЬ ЗВУКОВЫЕ ЭФФЕКТЫ
851 POKE 36877,220
852 FOR I=15 TO 0 STEP -1
853 POKE 36878,I
854 FOR J=1 TO 200:NEXT J
855 NEXT I
856 POKE 36877,0
900 REM XXXXXXXXXXXXXXXXXXXXXXXX ПОВТОРИТЬ? XXXXXXXXXXXXXXXXXXXXXXXX
910 GET H$ : REM LET H$=INKEY$
920 IF H$<>"" THEN GOTO 910
930 PRINT
940 PRINT B$+"ПОВТОРИТЬ? Y ИЛИ N"
950 GET H$ : REM LET H$=INKEY$
960 IF H$<>"Y" AND H$<>"N" THEN GOTO 950
970 IF H$="Y" THEN GOTO 140
980 END : REM STOP
1000 REM XXXXXXXXXXXXXXXXXXXXXXXX ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXXX
1010 PRINT HM$; : REM PRINT AT 0,0
1020 PRINT R$+"      ОРБИТА"
1030 PRINT
1040 PRINT
1050 PRINT
1060 PRINT
1070 PRINT G$+"ВЕРТ.ТЯГА=";STR$(TH);"  "
1080 PRINT
1090 PRINT K$+"ГОРИЗ.ТЯГА=";STR$(TS);"  "
1100 PRINT
```

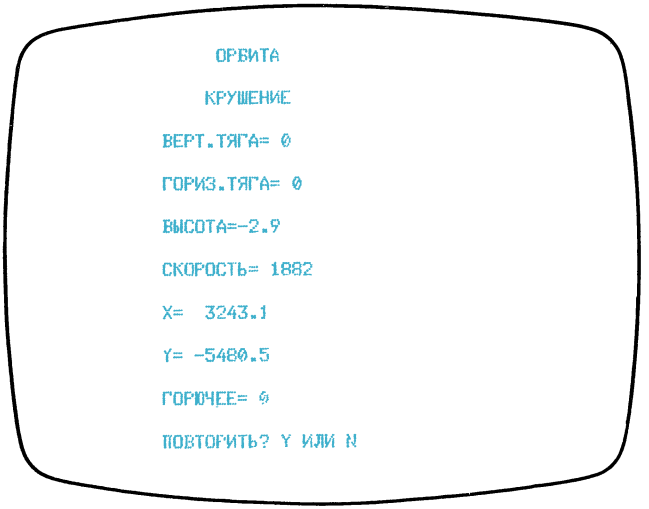
Если есть такая возможность, дайте свои звуковые эффекты.

```
1110 PRINT M$+"ВЫСОТА="STR$(HT);"  "
1120 PRINT
1130 PRINT C$+"СКОРОСТЬ=";STR$(V);" "
1140 PRINT
1150 PRINT R$+"X= ";X$;"  "
1160 PRINT
1170 PRINT R$+"Y= ";Y$;"  "
1180 PRINT
1190 PRINT M$+"ГОРЮЧЕЕ= ";STR$(FUEL);"  "
1200 RETURN
```

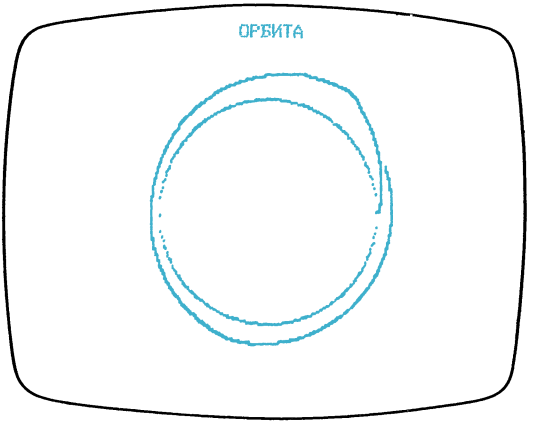
Дополнительные замечания см. в приложении.

```
        ОРБИТА

ЕСЛИ ГОТОВЫ ВКЛЮЧИТЬ
ДВИГАТЕЛЬ, НАЖМИТЕ Y

ВЕРТ.ТЯГА= 45

ГОРИЗ.ТЯГА= 5

ВЫСОТА= .1

СКОРОСТЬ= 0

X=  6371.1

Y=  0

ГОРЮЧЕЕ=  10000

ПОКАЗАНИЯ ПРИБОРОВ
ВЕРТ.ТЯГА  Z:-  C:+
ГОРИЗ.ТЯГА B:-  M:+
```

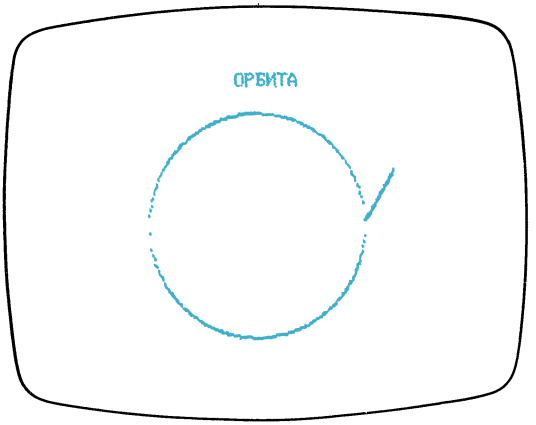
```
        ОРБИТА

     ВЫ ЛЕТИТЕ

ВЕРТ.ТЯГА= 0

ГОРИЗ.ТЯГА= 0

ВЫСОТА= 1318.9

СКОРОСТЬ= 868

X= -4216.9

Y= -6430.8

ГОРЮЧЕЕ=  0

ПОКАЗАНИЯ ПРИБОРОВ
ВЕРТ.ТЯГА  Z:-  C:+
ГОРИЗ.ТЯГА B:-  M:+
```



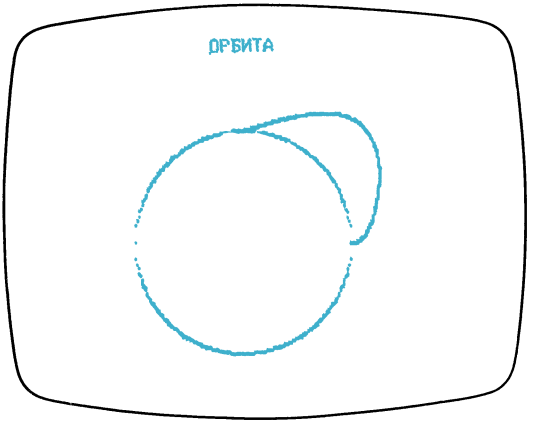
Можно предложить целый ряд усовершенствований программы ОРБИТА. Например, на экран дисплея можно выводить траекторию полета ракеты (см. рисунки). В программе не учитывалось сопротивление воздуха и проч. Можете это учесть. Если хотите, добавьте процедуру возвращения на Землю.



Успешный выход на орбиту



Отрыв от зоны земного притяжения



Крушение

Еще о дифференциальных уравнениях

В предыдущей главе мы познакомились с несколькими дифференциальными уравнениями. Дифференциальное уравнение, в котором производная некоторой переменной пропорциональна самой этой переменной, представляет собой весьма общее соотношение для окружающих нас явлений природы. Например, рост численности бактерий в колонии.

Рост живой клетки продолжается до тех пор, пока не происходит ее деление на две клетки. Эти «дочерние» клетки в свою очередь продолжают расти, пока тоже не разделятся на две, и так этот процесс длится и длится. Промежуток времени между двумя делениями называется *периодом генерации*. Клетки в колонии делятся не одновременно, да и период генерации у клеток не совпадает. Но поскольку число их в колонии достаточно велико, в среднем можно считать, что период генерации — величина постоянная, которую можно определить из наблюдений. Скорость роста колонии удовлетворяет следующему дифференциальному уравнению:

$$Y' = A * Y.$$

Здесь Y означает численность клеток в колонии, Y′ — производную, или скорость ее изменения и A — коэффициент, зависящий от среднего периода генерации.

Дифференциальное уравнение можно интерпретировать так: число появлений новых бактерий в любой момент пропорционально их численности. Таким образом, если в некий момент времени было $Y_{стар}$ бактерий, то за (малый) промежуток времени произойдет $A * Y_{стар}$ делений. Следовательно, через данный малый промежуток времени бактерий станет

$$Y_{нов} = Y_{стар} + A * Y_{стар}.$$

Вы, разумеется, узнали, метод Эйлера — Коши решения дифференциальных уравнений.

Используя выписанное выше соотношение рекуррентным образом, можно выяснить численность бактерий в колонии за время T. Следующая запись демонстрирует, как это реализуется в компьютере.

```
INPUT A
INPUT T          (Временной интервал)
INPUT Y          (Исходная численность колонии)
FOR I=1 TO T
LET Y=Y+A*Y
NEXT I
PRINT Y
```

Пусть, например, имеется колония бактерий, растущая в соответствии с дифференциальным уравнением

$$Y' = 0.00009627 * Y,$$

где рассматривается временной интервал в 1 секунду. В начальный момент в колонии было 1000 бактерий. Сколько бактерий будет через 10 мин? Сколько через 2 часа? Эти проблемы решает программа ЭКСПОНЕНЦИАЛЬНЫЙ РОСТ БАКТЕРИЙ. Нужно ввести время T в секундах и компьютер выдаст ответ.

```
10 REM                    *************************
20 REM                    *                       *
30 REM                    * ЭКСПОНЕНЦИАЛЬНЫЙ РОСТ *
40 REM                    *       БАКТЕРИЙ        *
50 REM                    *************************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXX
110 LET A=0.00009627
120 PRINT CHR$(147) : REM ОЧИСТИТЬ ЭКРАН
130 PRINT "ЭКСПОНЕНЦИАЛЬНЫЙ РОСТ"
135 PRINT "      БАКТЕРИЙ"
140 PRINT
150 PRINT "ВНАЧАЛЕ БЫЛО 1000 БАКТЕРИЙ "
160 LET Y=1000 : REM ИСХОДНАЯ ЧИСЛЕННОСТЬ
170 PRINT
180 PRINT "УКАЖИТЕ ВРЕМЕННОЙ ИНТЕРВАЛ"
190 PRINT "В СЕКУНДАХ ";
200 INPUT T
210 PRINT
220 FOR I=0 TO T                    Общие замечания можно найти в приложении.
230 LET Y=Y+A*Y
240 NEXT I
250 PRINT "ЧЕРЕЗ";T;"СЕКУНД"
260 PRINT "СТАЛО ";INT(Y)
300 REM XXXXXXXXXXXXXXXXXXXXXX  ПОВТОРИТЬ? XXXXXXXXXXXXXXXXXXXXXXXXXXXX
310 PRINT
320 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
330 GET G$ : REM LET G$=INKEY$
340 IF G$<>"Y" AND G$<>"N" THEN GOTO 330
350 IF G$="Y" THEN GOTO 160
```

```
ЭКСПОНЕНЦИАЛЬНЫЙ РОСТ
      БАКТЕРИЙ

ВНАЧАЛЕ БЫЛО
 1000 БАКТЕРИЙ

УКАЖИТЕ ВРЕМЕННОЙ ИНТЕРВАЛ
В СЕКУНДАХ ? 600

ЧЕРЕЗ 600 СЕКУНД
СТАЛО 1059

 ПОВТОРИТЬ? Y ИЛИ N

УКАЖИТЕ ВРЕМЕННОЙ ИНТЕРВАЛ
В СЕКУНДАХ ? 7200

ЧЕРЕЗ 7200 СЕКУНД
СТАЛО 2000

 ПОВТОРИТЬ? Y ИЛИ N
```

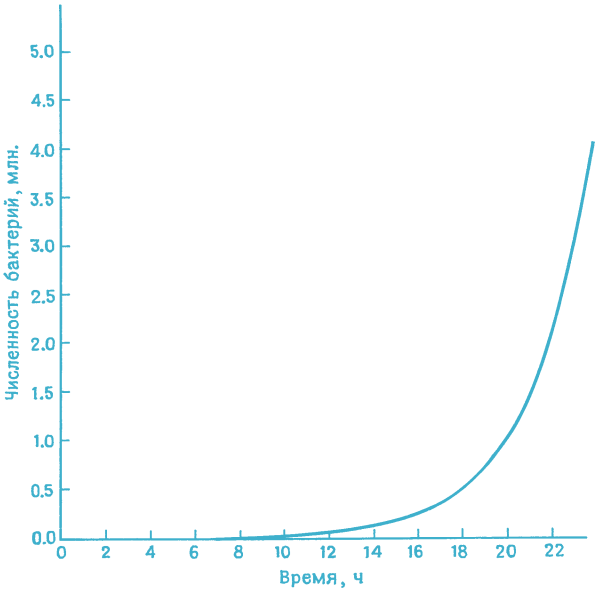
Заметим, что через 2 часа (или 7200 с) численность колонии удвоится. На самом деле, если удвоение численности колонии происходит за D секунд, то значение A в дифференциальном уравнении $Y' = A*Y$ равно (приблизительно) 0.69315/D.

Название ЭКСПОНЕНЦИАЛЬНЫЙ РОСТ БАКТЕРИЙ возникло по той причине, что решение нашего дифференциального уравнения имеет вид

$$Y = C*EXP(A*T),$$

где T — время, а C — константа, определяемая из наблюдений. Не будем здесь вдаваться в детали и выяснять, почему решение выражается именно так; если вы чуть-чуть знакомы с математическим анализом, сможете это проверить сами. Мы же будем довольствоваться приближенным решением, полученным при помощи метода Эйлера — Коши.

Само собой разумеется, настоящая жизнь не так проста. Если бы дифференциальное уравнение $Y' = A*Y$ на самом деле описывало скорость роста численности бактерий, то эти бактерии чрезвычайно быстро заполонили бы весь мир. Это хорошо видно из графика.



Миграция, смерть, истощение запасов пищи, температурные колебания — все отражается на численности бактерий. Чтобы результат соответствовал реальности, все эти факторы надо учитывать. Вкратце опишем, как это делается.

Численность всякой популяции имеет тенденцию к «экспоненциальному» росту, как в случае с бактериями. Это связано с тем, что рождаемость непосредственно связана с численностью популяции. Такой рост, однако, не может продолжаться вечно — размеры и ресурсы окружающего мира ограничены.

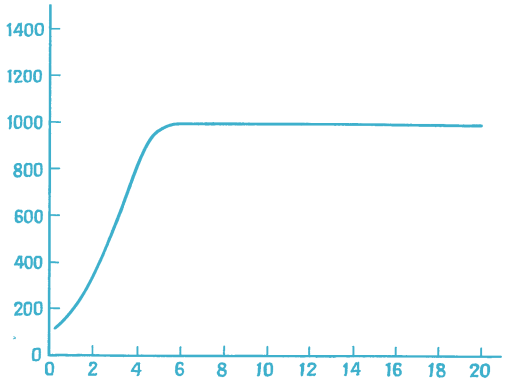
Изменение численности популяции более строго описывает такое дифференциальное уравнение:

$$Y' = A*Y - B*Y*Y.$$

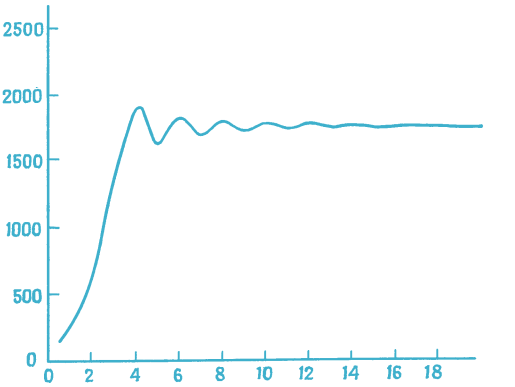
Здесь рождаемость все еще пропорциональна численности популяции (это член $A*Y$), но некоторая доля приходится на смертность. Эта доля зависит от численности популяции, и, вообще говоря, с увеличением численности смертность растет. Предполагается, что доля смертности равна $B*Y$, так что количество смертей, выраженное вторым членом уравнения, есть $(B*Y)*Y$.

При таком дифференциальном уравнении рост численности популяции не будет длиться бесконечно. Вообще говоря, график роста будет

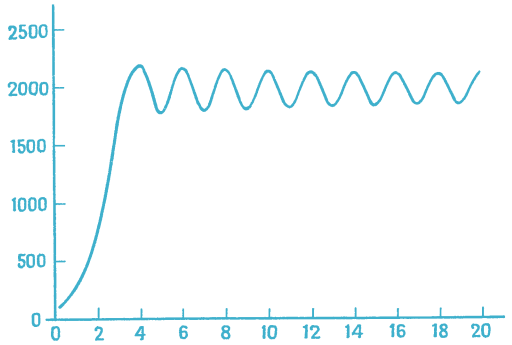
изображаться S-образной кривой. Вначале при сравнительно малых значениях Y рост экспоненциальный. Но при возрастании Y влияние, оказываемое смертностью на рост популяции, стремится к значению A/B. Заметим, что если в дифференциальном уравнении положить Y равным A/B, то получится выражение $Y' = 0$. Иначе говоря, скорость изменения численности популяции (ее производная) нулевая, так что популяция не меняет размера. Таким образом, как только численность популяции достигает значения A/B, она и далее продолжает быть той же. Бывает, что численность популяции никогда не застывает на значении A/B, а, превысив его, спускается вниз, затем снова поднимается и т. д. Что именно произойдет с численностью, зависит от величин A и B. Далее приводится несколько графиков для различных значений A.



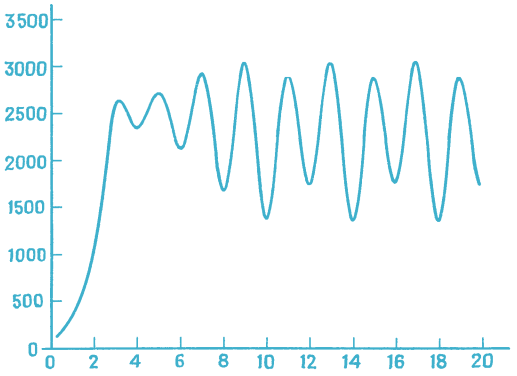
A = 1.Ø
B = Ø.ØØ1



A = 1.75
B = Ø.ØØ1



A = 2.Ø
B = Ø.ØØ1



A = 2.5
B = Ø.ØØ1

Понять, каким образом типичная популяция разрастается или вымирает, вам поможет программа РОСТ ПОПУЛЯЦИИ, в которой для решения дифференциального уравнения используется метод Эйлера — Коши. По команде INPUT введите значения A, B и исходную численность популяции. Программа выдаст численность популяции за 24-часовые периоды. Основной цикл строится в соответствии с уравнением

$$Y_{нов} = Y_{стар} + A * Y_{стар} - B * Y_{стар} * Y_{стар}.$$

```
 10 REM                    ******************
 20 REM                    *                  *
 30 REM                    * РОСТ ПОПУЛЯЦИИ  *
 40 REM                    *                  *
 50 REM                    ******************
 60 REM
 70 REM
100 REM XXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXX
110 PRINT CHR$(147); : REM ОЧИСТИТЬ ЭКРАН
120 PRINT "  РОСТ ПОПУЛЯЦИИ"
130 PRINT
140 PRINT " Y' = A*Y - B*Y*Y"
150 PRINT
160 PRINT "ЗНАЧЕНИЕ A";
170 INPUT A
180 PRINT
190 PRINT "ЗНАЧЕНИЕ B";
200 INPUT B
210 PRINT
220 PRINT "ИСХОДНАЯ ЧИСЛЕННОСТЬ"
230 PRINT "ПОПУЛЯЦИИ ";
240 INPUT Y
250 PRINT


300 REM XXXXXXXXXXXXXXXXXXXXXXXXXX СЧЕТ XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
310 FOR I=1 TO 24
320 LET Y=(1+A-B*Y)*Y
330 IF Y<0 THEN LET Y=0
340 PRINT INT(Y),
350 NEXT I


400 REM XXXXXXXXXXXXXXXXXXXXXXXXX ПОВТОРИТЬ? XXXXXXXXXXXXXXXXXXXXXXXXX
410 PRINT
420 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
430 GET G$ : REM LET G$=INKEY$
440 IF G$<>"Y" AND G$<>"N" THEN GOTO 430
450 IF G$="Y" THEN GOTO 110
```

Общие замечания по программированию см. в приложении.

Если вы предпочитаете иметь графики зависимости численности популяции от времени, то объедините программу РОСТ ПОПУЛЯЦИИ с программой ПОСТРОЕНИЕ ГРАФИКОВ из второй главы.

Борьба за существование. Если два биологических вида живут рядом, то они так или иначе взаимодействуют друг с другом в борьбе за существование. Например, один вид служит пищей для другого, как в

```
РОСТ ПОПУЛЯЦИИ

Y' = A*Y - B*Y*Y

ЗНАЧЕНИЕ A? 1.75

ЗНАЧЕНИЕ B? .001

ИСХОДНАЯ ЧИСЛЕННОСТЬ
ПОПУЛЯЦИИ ? 50

 135         353
 846         1610
 1834        1679
 1798        1711
 1777        1728
 1765        1738
 1758        1743
 1754        1746
 1752        1747
 1751        1748
 1750        1749
 1750        1749

ПОВТОРИТЬ? Y ИЛИ N
```

случае кроликов и лис. Кролики питаются дарами земли, а лисы поедают кроликов. Изменения в численности популяций кроликов и лис описывают следующие дифференциальные уравнения:

$$R' = A*R - B*R*R - C*R*F,$$

$$F' = -D*F + E*R*F.$$

Здесь R — количество кроликов, F — количество лис, а A, B, C, D и E — величины, зависящие от особенностей окружающей среды.

В качестве примера рассмотрим такие уравнения:

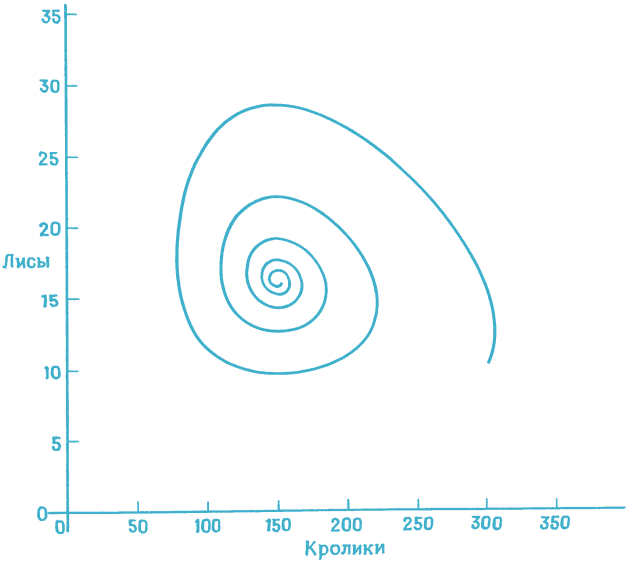
$$R' = 0.04*R - 0.00005*R*R - 0.002*R*F,$$

$$F' = -0.03*F + 0.0002*R*F.$$

Если бы не было лис, то численность кроликов установилась около значения 800. При наличии любого количества лис кроликов станет заведомо меньше чем 800. Если бы кроликов не было вовсе, то лисы вскоре бы вымерли. Произошло бы это не сразу, так как, обреченные на голодную смерть, они стали бы поедать ягоды и другие плоды.

Программа КРОЛИКИ и ЛИСЫ демонстрирует, что происходит с кроликами и лисами от месяца к месяцу. Введите сначала по команде INPUT исходные количества кроликов и лис. Вам предоставляется воз-

можность выбора: получать на экране либо численные значения, либо графическое изображение. При графическом выводе на экране появляется точка *, соответствующая определенным количествам кроликов и лис. По горизонтали ей соответствует значение R/20, т. е. одна двадцатая числа кроликов, а по вертикали — F/2, т. е. половина численности лис. Графическое изображение дает спираль, что свидетельствует о попеременном возрастании и убывании численности кроликов и лис (см. график).




```
10 REM                          ******************
20 REM                          *                  *
30 REM                          * КРОЛИКИ И ЛИСЫ  *
40 REM                          *                  *
50 REM                          ******************
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXX
110 LET CS=CHR$(147) : REM  КОД ОЧИСТКИ ЭКРАНА
120 LET PX=22 : REM КОЛИЧЕСТВО ТОЧЕК ПО ГОР.
130 LET PY=23 : REM КОЛИЧЕСТВО ТОЧЕК ПО ВЕРТ.
140 LET A=0.04
150 LET B=0.00005
160 LET C=0.002
170 LET D=0.03
180 LET E=0.0002
```

Попробуйте положить B = 0 и посмотрите, что из этого выйдет. Измените также значения A, C, D и E.

```
190 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
200 PRINT "  КРОЛИКИ И ЛИСЫ"
210 PRINT
220 PRINT "СКОЛЬКО КРОЛИКОВ";
230 INPUT R
240 PRINT
250 PRINT "СКОЛЬКО ЛИС";
260 INPUT F
270 PRINT
280 PRINT
290 PRINT "ВАРИАНТЫ:"
300 PRINT
310 PRINT "1. ЧИСЛЕННЫЙ ВЫВОД"
320 PRINT
330 PRINT "2. ВРЕМЕННЫЙ"
340 PRINT "ГРАФИЧЕСКИЙ ВЫВОД"
350 PRINT
360 PRINT "3. ПОСТОЯННЫЙ"
370 PRINT "ГРАФИЧЕСКИЙ ВЫВОД"
380 PRINT
390 PRINT "КАКОЙ ВАРИАНТ";
400 INPUT T
410 LET T=INT(T)
420 IF T<1 OR T>3 THEN GOTO 400
430 PRINT
440 IF T>1 THEN GOTO 610
```

Введите численность кроликов и лис.

Имеется три возможности.

```
500 REM %%%%%%%%%%%%%%%%%%%%%% ВАРИАНТ 1 %%%%%%%%%%%%%%%%%%%%%%%%%%
510 PRINT "КРОЛИКИ   ЛИСЫ"
520 PRINT
530 FOR I=1 TO 20
540 PRINT INT(R);TAB(10) INT(F)
550 GOSUB 910
560 NEXT I
570 GOTO 840
600 REM %%%%%%%%%%%%%%%%%%%% ВАРИАНТЫ 2 И 3 %%%%%%%%%%%%%%%%%%%%%%
610 PRINT CS$; : REM ОЧИСТИТЬ ЭКРАН
620 POKE 36879,170 : REM ЦВЕТНОЕ ИЗОБРАЖЕНИЕ
630 PRINT "  КРОЛИКИ И ЛИСЫ"
640 FOR I=1 TO 500
650 LET V=INT(F/2)
660 IF V>PY-2 THEN LET V=PY-2
670 IF V<=0 THEN LET V=0
680 LET U=INT(R/20)
690 IF U>PX-1 THEN LET U=PX-1
700 IF U<=0 THEN LET U=0
710 GOSUB 1010 : REM PLOT U,V
720 GOSUB 910
730 IF T=2 THEN GOSUB 1110
740 NEXT I
800 REM %%%%%%%%%%%%%%%%%%%%%% ПОВТОРИТЬ? %%%%%%%%%%%%%%%%%%%%%%%%
810 GET G$ : REM LET G$=INKEY$
820 IF G$="" THEN GOTO 810
830 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
840 PRINT " ПОВТОРИТЬ? Y ИЛИ N"
850 GET G$ : REM LET G$=INKEY$
860 IF G$<>"Y" AND G$<>"N" THEN GOTO 850
870 IF G$="Y" THEN GOTO 190
880 END : REM STOP
```

На экран выводится численность кроликов и лис по прошествии 20 месяцев.

Строится график зависимости между численностями лис и кроликов за период 500 месяцев.

Дополнительные замечания см. в приложении.

Пока вы не нажмете одну из клавиш, дисплей не изменится.

```
900 REM XXXXXXXXXXXXXXXXXXXXXXXXXX СЧЕТ XXXXXXXXXXXXXXXXXXXXXXXXXXXX
910 LET R=(1+A-B*R-C*F)*R
920 LET F=(1-D+E*R)*F
930 RETURN
1000 REM XXXXXXXXXXXXXXXXX ПОСТРОЕНИЕ ПОСРЕДСТВОМ POKE XXXXXXXXXXXXXXX
1010 POKE 8164+(PEEK(44))=18)*3584+U-22*V,42
1020 RETURN
```

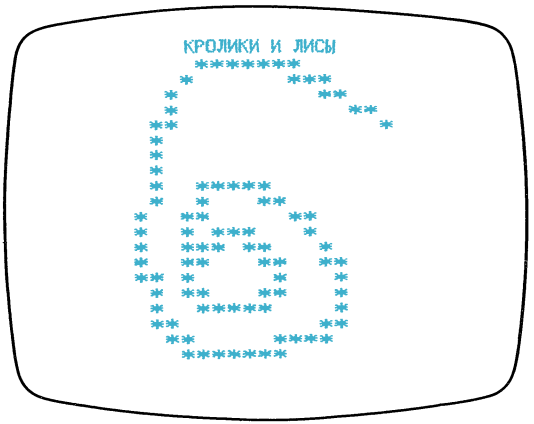
Вычисление R и F проводится методом Эйлера — Коши.

Если график строится посредством POKE, укажите соответствующие значения.

```
1100 REM XXXXXXXXXXXXXX ОТМЕНА ПОСТРОЕНИЯ ПОСРЕДСТВОМ POKE XXXXXXXXXXXXX
1110 POKE 8164+(PEEK((44))=18)*3584+U-22*V,32
1120 RETURN
```

```
  КРОЛИКИ И ЛИСЫ

СКОЛЬКО КРОЛИКОВ? 340

СКОЛЬКО ЛИС? 35

ВАРИАНТЫ:

1. ЧИСЛЕННЫЙ ВЫВОД

2. ВРЕМЕННЫЙ
ГРАФИЧЕСКИЙ ВЫВОД

3. ПОСТОЯННЫЙ
ГРАФИЧЕСКИЙ ВЫВОД

КАКОЙ ВАРИАНТ? 1
```

```
КРОЛИКИ    ЛИСЫ

 340        35
 324        36
 308        37
 292        38
 277        39
 263        40
 248        41
 235        41
 222        42
 209        42
 198        43
 186        43
 176        43
 166        44
 156        44
 148        44
 139        43
 132        43
 125        43
 118        43
```



Вирус. Воспользовавшись отдельными идеями из рассуждений о кроликах и лисах, можно построить одну интересную игру. Игра эта отнюдь не претендует на полное отражение действительности.

В игре ВИРУС предполагается, что в страну занесен (возможно, внеземными цивилизациями) новый вирус, и палата полна больных, зараженных вирусом. Под воздействием вируса здоровые клетки перерождаются и воспроизводят себе подобных. Под воздействием трансформированных клеток перерождаются другие здоровые клетки. У тяжело больных пациентов развиваются ужасающие симптомы, не поддающиеся описанию. Больные вскоре погибают при резком уменьшении лейкоцитов.

Изобретен новый препарат, но нет времени для проведения всего комплекса необходимых испытаний. Этот препарат цитотоксичен, т. е. он разрушает не только трансформированные клетки, но и здоровые. Экспериментально было показано, что более 600 мг препарата, накопленного в организме, смертельны для больного. Кроме того, препарат выводится из организма довольно медленно. Через час после приема выводится только 83% лекарства.

Содержание лейкоцитов в крови ваших пациентов уже снизилось почти до уровня 7000. Если оно опустится ниже 1500, пациент умрет. Чтобы помочь вам, компьютер каждый час выдает сведения о лейкоцитозе и количестве трансформированных клеток. Каждый час больным можно давать некоторую дозу лекарства. Но помните, что доза, превышающая 600 мг, для больного смертельна. Не забывайте и о свойстве препарата накапливаться в организме.

С пациентом неожиданно может произойти анафилактический шок (аллергическая реакция на препарат), с мгновенным смертельным исходом.

Ваша цель — спасти жизнь как можно большему количеству людей.

В программе ВИРУС используются такие дифференциальные уравнения:

$$X' = 0.01*X - 0.000001*X*X - 0.0001*X*Y - 0.0005*X*D,$$

$$Y' = -0.005*Y - 0.0000001*Y*Y + 0.00001*X*Y - 0.001*Y*D.$$

Здесь X обозначает количество здоровых клеток, Y — количество трансформированных клеток и D — содержание препарата в теле больного. В качестве элементарного интервала времени взят отрезок в 12 мин. Это означает, что в течение часа уравнения используются 5 раз. По прошествии каждых 12 мин содержание в организме препарата D уменьшается посредством умножения на 0.7. Таким образом, по прошествии часа в организме еще остается около 17% препарата, и это количество продолжает оказывать воздействие на клетки.

```
10 REM                          *********
20 REM                          *       *
30 REM                          * ВИРУС *
40 REM                          *       *
50 REM                          *********
60 REM
70 REM
100 REM XXXXXXXXXXXXXXXXXXX НАЧАЛЬНАЯ УСТАНОВКА XXXXXXXXXXXXXXXXXXXXXXXXX
110 LET CS$=CHR$(147) : REM КОД ОЧИСТКИ ЭКРАНА
120 LET B$=CHR$(31) : REM ГОЛУБОЙ
130 LET R$=CHR$(28) : REM КРАСНЫЙ
140 LET M$=CHR$(156) : REM ФИОЛЕТОВЫЙ
150 LET G$=CHR$(30) : REM ЗЕЛЕНЫЙ
160 LET C$=CHR$(159) : REM БИРЮЗОВЫЙ
170 LET Y$=CHR$(158) : REM ЖЕЛТЫЙ
```

Если на вашей машине есть цветное изображение, используйте соответствующие коды; в противном случае удалите B$, R$ и т. д.

```
180 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
190 PRINT R$+"         ВИРУС"
200 PRINT
210 PRINT B$+"У ВАС 10 ПАЦИЕНТОВ"
220 PRINT
230 PRINT B$+" РЕКОМЕНДОВАН ПРЕПАРАТ,"
240 PRINT B$+" НО НЕ БОЛЕЕ 600 МГ"
250 PRINT
260 PRINT R$+"ПРИ ЛЕЙКОЦИТОЗЕ"
270 PRINT R$+"НИЖЕ 1500"
280 PRINT R$+"ПАЦИЕНТ УМИРАЕТ"
290 PRINT
300 PRINT G$+" СПАСИТЕ КАК МОЖНО"
310 PRINT G$+" БОЛЬШЕ - ЖЕЛАЕМ УСПЕХА"
```

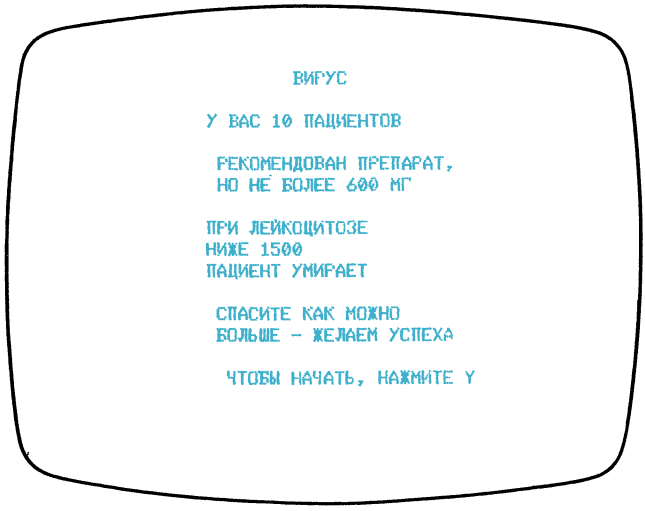
Здесь вкратце излагаются правила.

```
320 PRINT
330 PRINT Y$+"  ЧТОБЫ НАЧАТЬ, НАЖМИТЕ Y"
340 GET H$ : REM LET H$=INKEY$
350 IF H$<>"Y" THEN GOTO 340
360 LET PATIENT=0
370 LET T=9.1 : REM ВРЕМЯ В ЧАСАХ
380 LET T$="A.M."
390 LET N=0 : REM СПАСЕННЫЕ ПАЦИЕНТЫ
400 REM XXXXXXXXXXXXXXXXXXXXXX НАЧАЛО XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
410 PRINT CS$ : REM ОЧИСТИТЬ ЭКРАН
420 LET PATIENT=PATIENT+1
430 PRINT B$+"ПАЦИЕНТ ";PATIENT
440 PRINT
450 LET X=6900+INT(RND(1)*200) : REM ЛЕЙКОЦИТОЗ
460 LET Y=100+INT(RND(1)*20) : REM ТРАНСФОРМИРОВАННЫЕ КЛЕТКИ
470 LET D=0 : REM ПРЕПАРАТ
480 LET P=RND(1)
500 REM XXXXXXXXXXXXXXXXXXXXXX ДИСПЛЕЙ XXXXXXXXXXXXXXXXXXXXXXXXXX
510 PRINT R$+"ЛЕЙКОЦИТОЗ";X
520 PRINT
530 PRINT M$+"ТРАНСФ.КЛЕТОК";Y
540 PRINT
550 PRINT C$+" *  *  *  *  *  *  *"
560 PRINT
600 REM XXXXXXXXXXXXXXXXX ПРОВЕРКА НА ОКОНЧАНИЕ XXXXXXXXXXXXXXXXXX
610 IF X<1500 THEN GOTO 1210
620 IF Y=0 THEN GOTO 1310
700 REM XXXXXXXXXXXXXXXXXXXXXX СЧЕТ XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
710 PRINT B$+"ВРЕМЯ: ";INT(T+1-12*INT(T/12));T$
720 PRINT
730 PRINT G$+"КОЛИЧЕСТВО ПРЕПАРАТА?"
740 INPUT Q
750 PRINT
760 LET D=D+Q
770 IF D>0 AND P<0.08 THEN GOTO 1010
780 IF D>600 THEN GOTO 1110
790 FOR I=1 TO 5
800 LET X=X*(1.01-0.0001*(0.01*X+Y+5*D))
810 LET Y=Y*(0.995-0.001*(0.01*(0.01*Y-X)+D))
820 LET D=0.7*D
830 NEXT I
840 IF Y<0 THEN LET Y=0
850 LET X=INT(X)
860 LET Y=INT(Y+0.3)
870 LET Y=T+1
880 LET T$="P.M."
890 IF 2*INT(T/24)=INT(T/12) THEN LET T$="A.M."
900 LET T1=INT(T-24*INT(T/24))
910 IF T1=23 THEN LET T$="ПОЛНОЧЬ"
920 IF T1=11 THEN LET T$="ПОЛДЕНЬ"
930 GOTO 510
1000 REM XXXXXXXXXXXXXXXX АНАФИЛАКТИЧЕСКИЙ ШОК XXXXXXXXXXXXXXXXXXXX
1010 PRINT R$+"У ВАШЕГО ПАЦИЕНТА"
1020 PRINT R$+"АНАФИЛАКТИЧЕСКИЙ ШОК"
1030 PRINT
1040 GOTO 1210
1100 REM XXXXXXXXXXXXXXXXXX ПЕРЕДОЗИРОВКА XXXXXXXXXXXXXXXXXXXXX
1110 PRINT R$+"ВАШ ПАЦИЕНТ ПОЛУЧИЛ"
1120 PRINT R$+"ВЫСОКУЮ ДОЗУ ПРЕПАРАТА"
1130 PRINT
```

Имеется 8 шансов из 100, что у пациента будет аллергическая реакция. Если желаете, можете изменить эту цифру в строке 770.

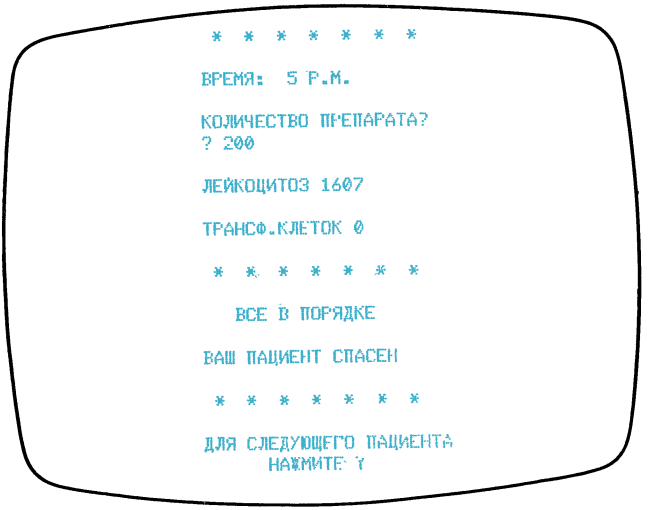
```
1200 REM XXXXXXXXXXXXXXXXXX ПАЦИЕНТ СКОНЧАЛСЯ XXXXXXXXXXXXXXXXXXXXX
1210 PRINT R$+"ВАШ ПАЦИЕНТ ТОЛЬКО ЧТО"
1220 PRINT R$+"       СКОНЧАЛСЯ"
1230 PRINT
1240 GOTO 1410
1300 REM XXXXXXXXXXXXXXXXXXXX ПАЦИЕНТ СПАСЕН XXXXXXX XXXXXXXXXXXXXXXXX
1310 PRINT G$+"   ВСЕ В ПОРЯДКЕ"
1320 PRINT
1330 PRINT G$+"ВАШ ПАЦИЕНТ СПАСЕН"
1340 PRINT
1350 LET N=N+1

1400 REM XXXXXXXXXXXXXXXXXX СЛЕДУЮЩИЙ ПАЦИЕНТ XXXXXXXXXXXXXXXXXXXXXXXX
1410 PRINT Y$+" *  *  *  *  *  *  *"
1420 PRINT
1430 IF PATIENT>9 THEN GOTO 1510
1440 PRINT C$+"ДЛЯ СЛЕДУЮЩЕГО ПАЦИЕНТА"
1450 PRINT C$+"       НАЖМИТЕ Y"
1460 GET H$ : REM LET H$=INKEY$
1470 IF H$<>"Y" THEN GOTO 1460
1480 GOTO 410
1500 REM XXXXXXXXXXXXXXXXXXXXX ПОВТОРИТЬ? XXXXXXXXXXXXXXXXXXXXXXXXX
1510 PRINT B$+"  ИЗ 10 ПАЦИЕНТОВ
1520 PRINT B$+"  ВЫ СПАСЛИ";N
1530 PRINT B$+"  ЗА ";INT(T-9); "ЧАСОВ"
1540 PRINT
1550 PRINT G$+" ПОВТОРИТЬ?  Y ИЛИ N"+B$
1560 GET H$ : REM LET H$=INKEY$
1570 IF H$<>"Y" AND H$<>"N" THEN GOTO 1560
1580 IF H$="Y" THEN GOTO 180
```

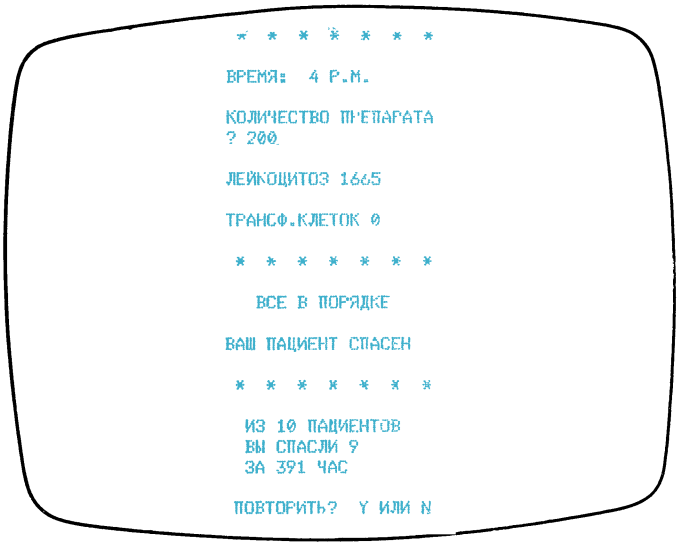
В программе ВИРУС звуковые эффекты не предусмотрены. При желании можете их ввести.



```
ПАЦИЕНТ  1

ЛЕЙКОЦИТОЗ 7025

ТРАНСФ.КЛЕТОК 118

 *  *  *  *  *  *  *

ВРЕМЯ:  10 A.M.

КОЛИЧЕСТВО ПРЕПАРАТА?
? 600

ЛЕЙКОЦИТОЗ 2761

ТРАНСФ.КЛЕТОК 17

 *  *  *  *  *  *  *
```

```
 *  *  *  *  *  *  *

ВРЕМЯ:  11 P.M.

КОЛИЧЕСТВО ПРЕПАРАТА?
? 80

ЛЕЙКОЦИТОЗ 1179

ТРАНСФ.КЛЕТОК 2

 *  *  *  *  *  *  *

ВАШ ПАЦИЕНТ ТОЛЬКО ЧТО
      СКОНЧАЛСЯ

 *  *  *  *  *  *  *

ДЛЯ СЛЕДУЮЩЕГО ПАЦИЕНТА
      НАЖМИТЕ Y
```


* * * * * * *
ВРЕМЯ: 5 P.M.
КОЛИЧЕСТВО ПРЕПАРАТА?
? 200
ЛЕЙКОЦИТОЗ 1607
ТРАНСФ.КЛЕТОК 0
* * * * * * *
ВСЕ В ПОРЯДКЕ
ВАШ ПАЦИЕНТ СПАСЕН
* * * * * * *
ДЛЯ СЛЕДУЮЩЕГО ПАЦИЕНТА
НАЖМИТЕ Y


* * * * * * *
ВРЕМЯ: 4 P.M.
КОЛИЧЕСТВО ПРЕПАРАТА
? 200
ЛЕЙКОЦИТОЗ 1665
ТРАНСФ.КЛЕТОК 0
* * * * * * *
ВСЕ В ПОРЯДКЕ
ВАШ ПАЦИЕНТ СПАСЕН
* * * * * * *
ИЗ 10 ПАЦИЕНТОВ
ВЫ СПАСЛИ 9
ЗА 391 ЧАС
ПОВТОРИТЬ? Y ИЛИ N

Приложение

Новый облик вашей программы

Замечания по изменению программ

Цель настоящего приложения — помочь вам в случае необходимости кое-что переделать в программах. Все программы в книге составлены таким образом, что их легко приспособить к вашему персональному компьютеру.

Практические вопросы. Введите программу с клавиатуры, обращая внимание на комментарии (REM); в процессе ввода делайте любые очевидные изменения. Информацию относительно тех или иных изменений, которые могут понадобиться, вы найдете далее в приложении. При вводе программы с клавиатуры, будьте внимательны; особое внимание обращайте на запятые и точки с запятой, число 1 и букву I, и, наконец, число 0 и букву O.

После того как программа введена, было бы благоразумно сохранить (SAVE) ее на всякий случай. После этого попробуйте ее выполнить (RUN). Если она будет работать неправильно, сначала проверьте, верно ли вы ее набрали. Затем убедитесь, что все необходимые изменения внесены. Программы в книге были тщательно оттестированы, так что должны работать нормально.

Наконец, придайте программе внешний лоск, используя все имеющиеся на вашем компьютере цветовые и звуковые средства.

Комментарии (REM). В программы включено достаточное количество комментариев. При необходимости ими можно благополучно пренебречь. В отдельных случаях (например, на некоторых микрокомпьютерах Sinclair), комментарии, следующие за оператором, нужно удалить. В некоторых комментариях приводятся возможные варианты для различных трансляторов с Бейсика.

PRINT CS$. Очень часто встречающаяся команда. Ее назначение — очистить экран и передвинуть курсор в исходное положение — (в верхний левый угол экрана).

На многих персональных компьютерах используется команда CLS, на других вам придется использовать еще какой-то подходящий код. Например, на компьютерах Commodore — это LET

CS$ = CHR$(147), хотя возможны и другие варианты. Для ряда машин правильной будет команда CHR$(12). Это, например, микрокомпьютеры BBC, Nascom II и UK 101, хотя на первых двух можно использовать и CLS. Команда CHR$(12) на машинах фирмы Research Machines очищает экран, но сдвигает курсор в левый нижний угол. В более современных моделях используется команда CHR$(31) для очистки экрана и перемещения курсора в левый верхний угол.

На некоторых микрокомпьютерах экран очищается при помощи литеры авторегистра CHR$(27), за которой следует другая литера. Так, на машинах Apple это CS$ = CHR$(27) + CHR$(64), хотя очистить экран можно и командой HOME. На компьютерах Sirius 1 и Victor 9000 используется CS$ = CHR$(27) + CHR$(69).

Загляните в ваше руководство — из него вы узнаете, какой именно командой нужно очищать экран.

PRINT HM$. Эта команда используется для перемещения курсора в его исходное положение (левый верхний угол), экран при этом не очищается.

Если имеется такая возможность, используйте PRINT AT 0,0. Это допустимо, например, на микрокомпьютерах Sinclair. В противном случае надо выяснить, какой код CHR$ вам подходит. Например, в компьютерах Commodore — это CHR$(19), хотя возможны варианты. На машине Atom — это $30, тогда как для TRS 80 подойдет команда CHR$(28). На последних моделях компьютеров Research Machines используется CHR$(29). На компьютерах Apple исходное положение устанавливается командой VTAB 1:HTAB 1.

На некоторых машинах исходное положение достигается при помощи литеры авторегистра CHR$(27), за которой следует другая литера. Так, например, на Sirius 1 и Victor 900 вы можете использовать HM$ = CHR$(27) + CHR$(72).

Инверсное изображение. Иногда используется обратное, или инверсное изображение букв на экране. Это средство не самое необходимое, но оно все-таки улучшает отдельные программы.

На многих персональных компьютерах имеется команда, которая включает или выключает обратное изображение. Например, на компьютерах Commodore для включения используется команда CHR$(18), а для выключения — CHR$(146). На некоторых машинах применяется авторегистр — литера CHR$(27), за которой следует другая литера. В компьютере Apple используются слова INVERSE и NORMAL соответственно для включения и выключения обратного изображения. На машине ZX81 марки Sinclair обратное изображение получается добавлением 128 к коду литеры. Таким образом, CHR$(38) дает

A, тогда как CHR$(166) дает инверсное A. На компьютерах Spectrum для получения обратного изображения используется команда CHR$(20)+CHR$(1), а для ее отмены — команда CHR$(20)+CHR$(0).

GET G$: REM LET G$=INKEY$. Эта команда используется для получения однолитерной цепочки, причем здесь не требуется нажатия клавиш RETURM, NEWLINE или ENTER. Если на клавиатуре нет набранных литер, то получится пустая цепочка.

Пусть, например, имеется такая программа:

```
510 GET G$ : REM LET G$=INKEY$
520 IF G$="Z" THEN LET TH=TH-1
530 IF G$="C" THEN LET TH=TH+1
540 IF G$="B" THEN LET A=A-1
550 IF G$="M" THEN LET A=A+1
```

В строке 510 мы не хотим, чтобы программа ждала, когда будет нажата какая-нибудь клавиша. Иначе говоря, мы хотим, чтобы программа прошла дальше, на следующую строку, если никакая клавиша не нажата.

В некоторых персональных компьютерах это же достигается с помощью одной из следующих команд:

```
GET G$        (кроме машин Apple)
LET G$=INKEY$
LET G$=GET$(0)
LET G$=INKEY$(0)
LET G$=INPUT$(1)
```

На некоторых машинах бывает нужно заблаговременно объявить G$="" ; сверьтесь с вашим руководством.

Если ни одной из перечисленных команд в вашем компьютере нет или ни одна из них не дает нужного результата, воспользуйтесь PEEK или USR. Например, команда PEEK (−16384) − 128 на Apple дает код ASCII, соответствующий нажатой клавише, так что можно записать следующее:

```
LET G$=CHR$ (PEEK(−16384)−128)
```

После использования G$ следует поместить команду POKE — 16368,0 для очистки буфера.

Проверьте по вашему руководству , какой именно командой следует воспользоваться.

Иногда в программах встречаются такие две строки:

```
540 GET G$ : REM LET G$=INKEY$
550 IF G$ < >"Y" AND G$< >"N" THEN GOTO 540
```

Это означает, что программа *должна* ждать, пока клавиша не будет нажата. На некоторых компьютерах эти две строки в случае необходимости можно объединить в одну. Так, на машине Tandy TRS 80 можно использовать просто

```
540 G$=INSTR$(1)
```

Команда GET G$ на Apple соответствует ожиданию, пока клавиша не будет нажата, так что вместо двух строк достаточно одной:

```
540 GET G$
```

Сверьтесь по вашему руководству относительно имеющихся возможностей.

INPUT. Эта команда позволяет вводить в компьютер данные непосредственно с клавиатуры. Программа остановится, напечатав, как правило, знак вопроса на экране, подождет, пока человек не наберет данные и не нажмет какую-нибудь из клавиш RETURN, NEWLINE и ENTER. Ответ обычно появляется на экране. Иногда (как, например, на микрокомпьютерах Sinclair) ответ исчезает с экрана после того, как были нажаты клавиши NEWLINE или ENTER. В таких случаях после каждой команды INPUT следует просто добавить еще одну строку, чтобы печатать (PRINT) то значение, которое вводится. Предположим, в вашей программе имеются такие строки:

```
150 PRINT "TYPE IN NUMBER";
160 INPUT X
170 IF X>0 THEN GOTO 600
```

Если значение X не остается на экране, нужно просто добавить строку

```
165 PRINT X
```

На многих марках персональных компьютеров в команде INPUT используется приглашение. В примере, приведенном выше, это позволяет заменить строки 150 и 160 одной строкой

```
150 INPUT "TYPE IN NUMBER";X
```

Этот прием позволяет сократить некоторые программы.

RND(1). Конструкция RND(1) используется для обозначения случайного числа, заключенного между 0 и 1. На некоторых компьютерах используется просто RND — в таком случае следует провести соответствующие изменения.

Чтобы получить случайное число, заключенное между A и B, нужно использовать выражение

```
A + (B - A)*RND(1)
```

На некоторых компьютерах вместо него используется следующее:

```
A + RND(B - A)
```

На ряде машин по команде RND(1) генерируется фиксированная последовательность случайных чисел. В таких случаях RND(1) надо заменить на RND(X), где X — случайно выбранное целое число.

Сверьтесь по вашему руководству, как вам надо поступать, если вы хотите получать случайные числа.

DIM. В программах всегда нужно указывать размер массива. На многих компьютерах это необязательно, если размер равен 10 или меньше. Кроме того, для указания размера каждого массива используется отдельная строка. Часто все команды DIM можно разместить на одной строке. Если доступны оба варианта, то такие, например, строки

```
110 DIM A(7)
130 DIM B$(20)
140 DIM C(15)
```

можно заменить следующей:

```
110 DIM A(7),B$(20),C(15)
```

Сверьтесь по вашему руководству, как вам следует поступать.

PLOT U,V. Имея дело с компьютером, на котором предусмотрена команда PLOT, следует помнить о том, что исходной точкой считается левый нижний угол экрана. Если на вашем компьютере в качестве исходной точки берется верхняя левая, то изображение может оказаться перевернутым, если вы не сделаете соответствующих изменений. Например, вместо PLOT U,V надо писать PLOT U, SY-V.

Конкатенация цепочек. Для конкатенации цепочек используется знак +. На некоторых микрокомпьютерах — это запятая или знак точка с запятой. Сверьтесь по вашему руководству, как нужно поступать; в случае необходимости сделайте необходимые изменения.

Подцепочки цепочек. Следует избегать использования такой операции. Она весьма нестандартна для Бейсика.

Пробелы 1. В трансляторе с Бейсика фирмы Microsoft после числа всегда добавляется пробел. Положительному числу предшествует пробел, отрицательному — знак минус. В некоторых версиях Бейсика (например, в Бейсике для машины Sinclair) подобное не предусмотрено.

В программах настоящей книги *обычно* предполагается, что числа будут печататься, как в Microsoft BASIC. Если в вашей машине используется другая разновидность Бейсика, то нужно добавлять пробелы после и (или) перед числом. Например, типичный в подобных случаях строкой будет

```
140 PRINT "THE VALUE IS ";V
```

Чтобы она стала удобочитаемой, ее следует представить в таком виде:

```
140 PRINT " THE VALUE IS ";V
```

Иногда следующий за числом пробел, предусмотренный Microsoft BASIC, нам только мешает. Чтобы избежать этого, надо превратить число в цепочку (STR) и затем ее напечатать. Например,

```
420 PRINT STR$(X);
```

Иногда допустимо заменить такую строку следующей:

```
420 PRINT X;
```

Поэкспериментируйте самостоятельно и найдите нужный вариант.

Пробелы 2. В ряде версий Бейсика допускается отсутствие пробелов в строке. Например, вместо строки

```
750 IF G$="Y" THEN GOTO 180
```

у вас может быть

```
750 IFG$="Y"THENGOTO180
```

Пользуясь таким способом, старайтесь все-таки сохранить пробелы там, где это для вас важно.

LET. В большинстве компьютеров команда LET необязательна (за исключением микрокомпьютеров Sinclair). В наших программах LET всегда использовалась. Если слово LET необязательно, вы можете

при желании его опустить. Например, следующие две строки

```
690 LET A=0
700 IF B<0 THEN LET A=1
```

можно заменить такими:

```
690 A=0
700 IF B<0 THEN A=1
```

IF...THEN GOTO. Слово GOTO, стоящее за командой IF...THEN, на большинстве персональных компьютеров необязательно. Если хотите, можете его опустить. Например, строку

```
310 IF G$=""THEN GOTO 300
```

можно заменить на

```
310 IF G$="" THEN 300
```

На некоторых компьютерах, напротив, можно удалить THEN и оставить GOTO (есть типы компьютеров, на которых это надо делать обязательно). Так, в нашем примере исходную строку можно (в некоторых случаях — нужно) заменить следующей:

```
310 IF G$="" GOTO 300
```

Поэкспериментируйте самостоятельно и сверьтесь по руководству, как вам можно или нужно поступать.

Групповые команды. В некоторых машинах допускается иметь на одной строке несколько команд, отделенных друг от друга двоеточием. Все программы в этой книге написаны без использования групповых команд. Исключение представляют комментарии (REM), которые иногда включались в качестве второй команды на одной строке. Эти комментарии можно без ущерба удалять из программ. Многие программы можно сократить и, возможно, улучшить, используя групповые команды. Например, следующие строки

```
720 LET A=0
730 LET B=0.002
740 LET X=0.1
750 IF Y<0 THEN LET Z=0
760 IF Y<0 THEN GOTO 910
```

можно ужать до двух строк:

```
720 LET A=0:LET B=0.002:LET X=0.1
750 IF Y<0 THEN LET Z=0:GOTO 910
```

IF...THEN...ELSE. Такая команда не использовалась ни в одной из программ книги. Если на вашем компьютере это возможно, воспользуйтесь этим приемом, чтобы сократить одну — две строки. Так, например, две строки

```
680 IF K<0 THEN GOTO 210
690 GOTO 310
```

можно заменить всего одной:

```
680 IF K<0 THEN GOTO 210 ELSE GOTO 310
```

Квадратные и круглые скобки. Скобки присутствуют во всех функциях. На машинах Sinclair скобки необязательны для функций TAB, VAL, SIN. На таких машинах скобки при желании можно опустить, но будьте осторожны — иногда они необходимы.

NEXT I. На некоторых компьютерах вам не нужно специфицировать переменную после NEXT, как в NEXT I. Программа, в которой переменные не специфицируются, выполняется быстрее, иногда значительно быстрее.

Сжатие программ. Программы в этой книге написаны так, что они почти готовы для работы на вашем персональном компьютере. Понятно, что они не так «опрятны» и «коротки», как если бы они предназначались для какого-то конкретного компьютера.

Многие из советов, высказанных в этой главе, годятся и для вашего компьютера. Следуя им, можно сократить, а возможно, и улучшить немало из представленных в книге программ. Вы убедитесь, что любую программу можно сократить на четверть, а возможно, даже вдвое.

Переписывая программы и внося в них что-нибудь от себя, вы получите дополнительную дозу развлечений.

Дополнительные замечания относительно VIC20. Ряд программ предполагает использование графических устройств с высокой разрешающей способностью. Для них необходимо увеличить память машины на 3K и более.

Предостережение. Если вы добавите к памяти 8K и более, то областью высокого разрешения будет область, где машина предполагает хранить свой Бейсик. Следовательно, нужно переместить область Бейсика, используя указатель 44. Таким образом, прежде чем набрать или загрузить (LOAD) одну из этих программ, наберите следующее:

```
POKE 44,32:POKE 8192,0:NEW
```

Проверьте, правильно ли вы набрали и нажмите клавишу RETURN. Если вы уже набрали программу без этой команды, запомните ее (SAVE), введите приведенную выше команду и затем загрузите (LOAD) программу.

Наконец, если по какой-то причине одна из программ с областью высокого разрешения остановится где-то посередине вычислений, вы не увидите ничего, кроме полной бессмыслицы. В таком случае следует одновременно нажать клавиши RUN STOP и RESTORE.

Указатель

АНАЛИЗ ОЧЕРЕДИ 121
Арифметическая прогрессия 15
Арифметические последовательности 16
АРИФМЕТИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ 16
Арифметический ряд 16
Архимед 26

В ПОИСКАХ СОКРОВИЩ 59
Вероятностная функция распределения 113
Вирус 176
ВИРУС 177
ВЛОЖЕНИЕ КАПИТАЛА 81
Выбрасывание карт 71, 72
Выбрасывание монет 71, 73
Выбрасывание пальцев 71, 74, 85
ВЫЧИСЛЕНИЕ СИНУСА 28

Гармонический ряд 22
ГАРМОНИЧЕСКИЙ РЯД 23
Геометрическая последовательность 19
Геометрическая прогрессия 17, 18
ГЕОМЕТРИЧЕСКИЕ ПОСЛЕДОВАТЕЛЬНОСТИ 19
Геометрический ряд 18
ГОЛОВОЛОМКА 16 КВАДРАТОВ 101
ГОЛОВОЛОМКА 25 КВАДРАТОВ 106
График функции 37, 47
ГРАФИКИ В ПОЛЯРНЫХ КООРДИНАТАХ 48
Группа 92
Группа перестановок 95
Групповые команды 188
Гусеница на резине 25

Декарт 37
Декартовы координаты 37, 47, 56
Детерминант 66
Дифференциальное уравнение 145, 165
Длина окружности 27

Знакочередующийся ряд 26

ИГРА ДВУХ ЛИЦ С НУЛЕВОЙ СУММОЙ 75, 90, 91
Изолинии 138
ИЗОЛИНИИ 140
Инверсное изображение 183
Интервал между поступлениями 109
ИССЛЕДОВАНИЕ МАТРИЦ 66
Карта изолиний 138
Комментарии (REM) 182
Коммутативная группа 96
Конкатенация 186
Координаты 36
Коши 148
КРОЛИКИ И ЛИСЫ 173

Лейбниц 26
Лойд 100
Любители пива 90
Любящий муж 25

Матрица 62, 65, 69
Матрица тождественного преобразования 65
Метод Монте-Карло 109
Метод Эйлера-Коши 165
МОДНЫЙ МАГАЗИН 124

Некоммутативная группа 96

Обратная перестановка 95
ОБСЛУЖИВАНИЕ В БАНКЕ 117
Оптимальная стратегия 75, 84
ОРБИТА 160
ПАДАЮЩИЙ КАМЕНЬ 147
Перестановка 92
Период генерации 165
Пи 26
Платежная матрица 72, 79, 85, 90
Площадь круга 36
ПОВЕРХНОСТИ 136
Подцепочки 187
Показательное распределение 116
ПОЛЕТ 153
Полярные координаты 45, 46
Порядок перестановки 94
ПОСЛЕДОВАТЕЛЬНОСТИ И РЯДЫ 30
Последовательность 15
ПОСТРОЕНИЕ ГРАФИКОВ 40
ПОСТУПЛЕНИЕ КЛИЕНТОВ I 110
ПОСТУПЛЕНИЕ КЛИЕНТОВ II 112
ПОСТУПЛЕНИЕ КЛИЕНТОВ III 115
Приближенное решение 84
ПРИБЛИЗИТЕЛЬНЫЙ АНАЛИЗ ИГРЫ 86, 90
Пробелы 187

ПРЫГАЮЩИЙ МЯЧ 149
Прямоугольный треугольник 57
Пуассон 113
Пуассоновское распределение 113

Радианы 27
Расходящийся ряд 24
Решение уравнения 145
РОСТ ПОПУЛЯЦИИ 171

Синус 27
Скобки 189
Снежинки 20
Стратегические игры 70
Сходящийся ряд 24

Теорема Пифагора 57
Теория групп 92
Теория игр 70
Теория массового обслуживания 109
ТЕСТ ИНТЕЛЛЕКТУАЛЬНОСТИ 31
Тождественная перестановка 94
Точка минимакса 73
Транспозиция 100

Факториал 28
Функция в полярных координатах 46
Функция двух переменных 130
Функция накопленных вероятностей см. *Вероятностная функция распределения*

Цепочка 187

ЧЕТЫРЕ КВАДРАТА 97
Чет-нечет 71
Численное решение 146

Шарп 26

Эйлер 26, 148
Экспоненциальное распределение см. *Показательное распределение*
ЭКСПОНЕНЦИАЛЬНЫЙ РОСТ БАКТЕРИЙ 166

DIM 186
GET G$ 184
INPUT 185
LET 187
NEXT 189
PLOT 186
PRINT CS$ 182
PRINT HM$ 183
RND(1) 186
SINCLAIR GRAPH PLOTTING 42
SINCLAIR POLAR GRAPHICS 57
VIC 20 GRAPH PLOTTING 44
ZX 81 INVESTMENT GAME 84

Научно-популярное издание

Чес Косневски

ЗАНИМАТЕЛЬНАЯ МАТЕМАТИКА И ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

Научный редактор М.В. Хатунцева
Мл. научн. редактор Р.И. Пяткина
Художник А.В. Шипов
Художественный редактор В.И. Шаповалов
Технический редактор Н.И. Борисова
Корректор Р.Л. Вибке

ИБ № 5975

Подписано к печати 18.05.87. Формат 60 × 90¹⁄₁₆. Бумага офсетная № 1.
Гарнитура таймс. Печать офсетная. Объем 6,00 бум. л. Усл. печ. л. 12,00. Усл. кр.-отт. 24,48.
Уч.-изд. л. 12,68. Изд. № 1/4717. Тираж 50 000 экз. Зак. 159. Цена 80 к.

Набрано в издательстве «Мир» на фотонаборном комплексе «Компьюграфик»
129820, ГСП, Москва, 1-й Рижский пер., 2.

Можайский полиграфкомбинат Союзполиграфпрома при Государственном комитете СССР
по делам издательств, полиграфии и книжной торговли.
143200, Можайск, ул. Мира, 93.

80 к.

Эта книга научит вас некоторым полезным вещам как в информатике, так и в математике. Она станет для вас неисчерпаемым источником идей. Знания, которые вы приобретете, позволят вам самостоятельно писать и более сложные программы, чем помещенные в книге.