

Н.Н. Ставнистый



**часть
1**

BASIC **в математике**

**решение задач
с помощью
компьютера**

$$y = \sqrt{x}$$

Н. Н. Ставнистый

QBASIC В МАТЕМАТИКЕ

РЕШЕНИЕ ЗАДАЧ С ПОМОЩЬЮ КОМПЬЮТЕРА

Часть 1

**Солон-Р
Москва
2008**

Н. Н. Ставнистый
QBASIC В МАТЕМАТИКЕ
РЕШЕНИЕ ЗАДАЧ С ПОМОЩЬЮ КОМПЬЮТЕРА

Часть 1

Данная книга представляет собой первую часть пособия по изучению основ программирования на языке QBASIC. Пособие построено как введение в курс прикладной математики, т. е. основное внимание уделено не языку программирования, как таковому, а методам решения математических задач на компьютере. При этом сам процесс решения математических задач рассматривается как частный случай обработки информации, а программирование — как наука составления алгоритмов обработки информации. Таким образом, процесс обучения основам прикладной математики базируется на системе понятий, используемых в школьном курсе основ информатики и вычислительной техники.

Ответственный за выпуск
Верстка
Обложка

С. Иванов
А. Павлов
Е. Жбанов

Введение

Пособие предназначено для учеников 7—9 классов. Главы I—IV посвящены освоению нового метода решения математических задач — алгоритмического, в отличие от традиционного «формульного» метода, изучаемого в средней школе. В процессе освоения нового метода решаются в основном обычные задачи из школьных учебников математики.

Главы V и VI посвящены решению новых, нетрадиционных для средней школы задач, для решения которых, собственно, и предназначен компьютер.

В главе V, на базе известных из школьного курса алгебры арифметических и геометрических прогрессий, изучается методика решения математических задач, сводящихся к рекуррентным формулам. Глава VI посвящена решению задач с использованием логических переменных. В отличие от традиционно принятого в учебниках информатики изложения основ математической логики, понятие логической величины и логических операций начинается формироваться в 7-м классе; и эта тема проходит через весь курс в качестве неотъемлемого компонента решения задач на компьютере.

Во второй части пособия, предназначенного для 10—11-х классов, будут рассмотрены структуры данных в QBASIC (типы переменных, массивы, записи, файлы), будет введено понятие базы данных и рассмотрены основы комбинаторики.

Данное пособие написано на основе 15-летнего опыта преподавания автором курса информатики в школе и ведения специальности — «программист-лаборант» в УПК.

Глава I

Решение задач на компьютере

§ 1. Компьютер — машина для обработки информации

Слово «компьютер» пришло к нам из английского языка. Computer в переводе на русский язык означает «вычислитель», т. е. автомат (или человек), который вычисляет, считает. В русском языке есть аналогичное название — электронная вычислительная машина (ЭВМ).

Возникли компьютеры как машины, предназначенные для автоматизации математических расчетов. Первые ЭВМ использовались как большие электронные счеты. Впоследствии область применения компьютеров значительно расширилась.

Современные компьютеры могут очень многое. Компьютеры могут играть с нами в разнообразные игры (компьютерные игры — самый распространенный способ использования компьютеров), могут работать с текстами — использование компьютера в качестве электронной пишущей машинки второй по распространенности способ использования. Компьютер помогает художникам-модельерам и режиссерам кино. В кинематографе даже сложилось особое направление — компьютерные спецэффекты.

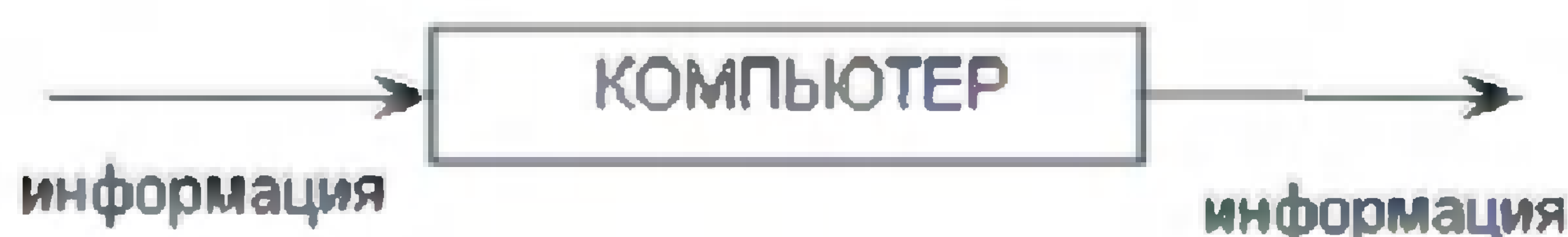
Однако все разнообразие работ, выполняемых современными компьютерами, можно охарактеризовать двумя словами: обработка информации. Что бы ни делал компьютер, какую бы работу ни выполнял, это все — обработка информации.

Если внимательно присмотреться к процессу работы на компьютере, то можно заметить, что между человеком и компьютером постоянно происходит обмен информацией. Мы нажимаем клавишу — и компьютер начинает работать. Нажимая клавишу, мы вводим в компьютер какую-то информацию, сообщаем компьютеру, что мы от него хотим. В результате своей работы компьютер также выдает нам информацию. Эта информация может появиться на экране, может быть распечатана принтером на бумаге, может быть записана на магнитный диск или послана по сети на другой компьютер. Обмен информацией с человеком — самая характерная черта работы компьютера.

Конечно, компьютер может обмениваться информацией не только с тем человеком, который в данный момент сидит перед экраном дисплея. Как уже говорилось, информацию можно пересылать с одного компьютера на другой. Информация может поступать с каких-то научных приборов, на

которых проводится эксперимент. Компьютер может быть центром сложной автоматизированной системы, которая управляет работой конвейера, цеха или даже целого предприятия. В общем, получать информацию компьютер может из самых разных источников.

Для нас сейчас неважно, откуда именно компьютер получает информацию. Важно лишь, что сущность работы компьютера заключается в обработке информации. Компьютер получает информацию из внешней среды и в качестве результата своей работы также выдает информацию:



Конечно, «информация», стоящая в правой части рисунка, — это не та «информация», которая стоит в левой части. «Информация», стоящая справа, — это результат работы компьютера, продукт его труда. «Информация», стоящая слева, — это исходное сырье для компьютера. Пропустив через себя полученную информацию, компьютер вырабатывает новую информацию и преподносит ее как результат своей работы, как продукт своего труда. Обычно говорят, что продукт получается в результате обработки сырья. Преобразование информации компьютером также называется обработкой.

Определение. Любые операции, производимые над информацией, называются обработкой информации.

Компьютер может обрабатывать различные виды информации: символьную, числовую, графическую. Все виды информации, с которыми работает компьютер, называются одним словом — «данные». Информация, которую компьютер получает, — это входные данные. Процесс получения данных компьютером называется вводом данных. Чаще всего данные вводятся с клавиатуры или с магнитного диска. Но существуют и другие средства ввода данных: манипулятор «мышь», сканер, модем.

Информация, которую компьютер выдает нам в результате своей работы, называется выходными данными. Выходные данные мы чаще всего видим на экране дисплея. Выходные данные можно также распечатать на принтере, записать на диск, переслать по сети другому компьютеру.

Работа компьютера заключается в обработке данных, т. е. в преобразовании входных данных в выходные. Обработка данных производится компьютером даже в тех случаях, когда на первый взгляд ничего не происходит. Например, мы нажали на клавишу с буквой «А». На экране появилась буква «А». В чем заключается здесь обработка информации?

Когда мы нажимаем клавишу, в «мозг» машины — процессор — посылается серия электрических импульсов. Для процессора эти импульсы означают команду: вывести на экран букву «А». Процессор «осмысливает» эту команду и, в свою очередь, посылает указание дисплею: вывести на экран букву «А». Получив эту команду, дисплей управляет движением электронного луча таким образом, чтобы на экране появилась буква «А».

Обработка входных данных здесь заключается в том, что процессор, получив сигнал от нажатой клавиши, рассылает по различным узлам машины

целую серию команд, выполнение которых и приводит к появлению символа на экране, причем описанная процедура появления буквы на экране очень упрощена. На самом деле одно нажатие клавиши может привести к выполнению нескольких сотен различных команд. Таким образом, даже простое высвечивание символа на экране в ответ на нажатие клавиши является результатом сложных процессов, происходящих в электронных блоках компьютера. Управляет работой всех блоков процессор, который посылает команды, преобразующие входные данные в выходные. Все эти команды объединяют одним словом — алгоритм.

Определение. Система команд обработки информации называется алгоритмом.

Алгоритм преобразует входные данные в выходные. Теперь процесс обработки информации компьютером можно представить следующим образом:



Алгоритм вывода символа на экран — это самый простой алгоритм, который компьютер может выполнить. Вы уже, наверное, знакомы с такими операциями, как копирование файлов, создание директорий, обработка в текстовом редакторе. Все эти операции выполняются при помощи алгоритмов, которые записаны в память компьютера.

§ 2. Алгоритм решения задачи

Итак, компьютер существует для того, чтобы обрабатывать информацию. А можем ли мы с вами обрабатывать информацию? Может ли человек, получив информацию, произвести новый информационный продукт?

Конечно, может. Мы постоянно получаем информацию из окружающего мира, и наш мозг обрабатывает (осмысливает) ее. Наиболее интенсивно идет обработка информации в школе. Самый близкий для нас пример обработки информации — решение математических задач.

Например, получает ученик задание: умножить 2 на 3 и результат написать на доске. Здесь входные данные — числа 2 и 3; выходные данные — число 6; алгоритм — операция умножения. Процесс решения этой задачи можно изобразить так же, как мы изображали работу компьютера:



Этой же схемой описывается процесс решения любой математической задачи: в условии задачи даются некоторые числа, которые служат входными данными; решая задачу, мы обрабатываем эти числа, совершаем над ними

математические операции, в результате чего получаем новые числа, которые и являются выходными данными. Те операции, которые выполняются в процессе решения задачи, представляют собой алгоритм решения задачи.

Определение. Алгоритм решения задачи — последовательность математических операций, которые нужно выполнить, чтобы решить данную задачу.

Таким образом, решение математических задач — частный случай обработки информации. Решая математические задачи, мы обрабатываем числовую информацию при помощи математических операций. Пока нам известны четыре такие операции — сложение, вычитание, умножение, деление.

Напишем алгоритм вычисления по формуле:

$$(34 + 15) : (5 + 2).$$

Решая этот пример, мы выполним три операции:

1. $34 + 15 = 49$.
2. $5 + 2 = 7$.
3. $49 : 7 = 7$.

Алгоритм получится, если мы запишем только названия этих операций, но не будем их выполнять и не будем записывать промежуточные результаты:

1. Сложить 34 и 15.
2. Сложить 5 и 2.
3. Результат 1-й операции разделить на результат 2-й операции.

Теперь вспомним, что алгоритм преобразует входные данные в выходные. Это значит, что при составлении алгоритма мы должны указать, какие числа у нас будут играть роль входных данных и что мы будем считать результатом работы алгоритма (т. е. обязательно нужно написать ответ). Таким образом, получаем окончательный вид алгоритма:

Входные данные — 4 числа: 34, 15, 5, 2.

Операции:

1. Сложить 1-е и 2-е число.
2. Сложить 3-е и 4-е число.
3. Результат 1-й операции разделить на результат 2-й операции.
4. Ответ: результат 3-й операции.

Как вы уже, наверное, заметили, алгоритм состоит из отдельных пунктов (шагов). Эту особенность алгоритма называют дискретностью. Слово «дискретность» на русский язык можно перевести как «прерывистость». Каждый шаг алгоритма завершается некоторым результатом, получением некоторого числа. Если эти результаты не являются выходными данными, то они называются промежуточными данными.

§ 3. Исполнитель алгоритма

Решение всех математических задач, с которыми вы до сих пор встречались в школе, можно представить как выполнение некоторого алгоритма. Почему же до сих пор вы не пользовались понятием алгоритма? Потому что в этом не было надобности. Решая задачу, вы составляли последова-

тельность операций, которые необходимо выполнить, и сами же выполняли эти операции.

Теперь мы будем решать задачи при помощи компьютера. На себя мы возьмем творческую часть решения — напомним последовательность операций, которые необходимо выполнить. Выполнять же эти операции будет компьютер.

Процесс решения задачи уже не будет представлять единого целого. Решать задачу будут два субъекта — человек и компьютер. Человек записывает все формулы, которые нужно вычислить, но сами вычисления не производит и ответа не получает. Вычисления проводит компьютер и доводит решение до числового результата. Компьютер здесь выступает в качестве исполнителя алгоритма. Алгоритм составляется человеком, который решает задачу. Будем этого человека называть программистом. Раньше программист и исполнитель были объединены в одном лице. Теперь они разделились.

Исполнителем алгоритма может быть не только компьютер. Представьте себе, что некоторый пятиклассник решает следующую задачу: на базу привезли в первый день помидоры на 8 машинах по 120 ящиков на машине и на второй день — на 6 машинах по 140 ящиков на машине. Сколько ящиков помидоров привезли на базу за два дня?

Пятиклассник получает решение в виде формулы:

количество ящиков равно $120 \times 8 + 140 \times 6$.

Рядом с ним сидит младший брат-второклассник, который научился выполнять на калькуляторе четыре арифметических действия и теперь развлекается с этим прибором. Пятиклассник решает извлечь пользу из этого увлечения своего брата и подает команду: «А ну-ка, умножь 120 на 8 и результат запиши!» Второклассник перемножает на калькуляторе два числа и записывает результат на листок бумаги. Следует вторая команда: «Умножь 140 на 6 и результат запиши!» Второклассник выполняет эту команду. Затем поступает третья команда: «А теперь сложи два полученных числа и результат сообщи мне».

Что же получилось? Каким образом решена задача? Пятиклассник получил решение задачи в виде некоторой формулы, но вычислять ее не стал. Чтобы получить результат, он сообщил своему брату алгоритм:

1. Умножить 120 на 8, результат записать.
2. Умножить 140 на 6, результат записать.
3. Сложить полученные числа.
4. Сообщить результат.

Здесь пятиклассник выступил в роли программиста, который составил алгоритм решения задачи и сообщил его своему брату-второкласснику. Второклассник сыграл роль исполнителя, который просто выполнил алгоритм и получил результат, ничего при этом не зная о самой задаче.

Теперь разберемся, что представляют собой шаги алгоритма. Каждая строка алгоритма — это указание исполнителю выполнить какую-то операцию: умножить, запомнить, сложить, сообщить. Мы неслучайно говорим, что пятиклассник давал команды второкласснику. Пункты (шаги) алгоритма называются иначе командами. Каждая команда алгоритма — это указание исполнителю выполнить конкретное действие. Алгоритм пред-

ставляет собой набор команд, предназначенный для некоторого исполнителя.

Исполнителем алгоритма может быть человек или некоторое автоматическое устройство. Главная особенность исполнителя состоит в том, что он может выполнять определенный набор операций. Алгоритм составляется из команд на выполнение этих операций. Операции, которые исполнитель может выполнять, образуют систему команд исполнителя.

Определение. Набор операций, доступных исполнителю, называется системой команд исполнителя.

Рассмотрим в качестве примера систему команд нашего исполнителя-второклассника. Очевидно, он умеет выполнять четыре арифметических действия. Кроме того, он умеет записать число на бумагу (запомнить) и сообщить результат программисту. Итого, получаем 6 команд:

1. Сложить два числа.
2. Вычесть из одного числа другое.
3. Перемножить два числа.
4. Разделить одно число на другое.
5. Запомнить результат (записать).
6. Сообщить результат.

Этих команд достаточно для того, чтобы составить алгоритм решения любой арифметической задачи.

§ 4. Алгоритм и программа

Когда исполнителем является человек, то алгоритм пишется на обычном (естественном) языке. Однако чаще всего алгоритмы составляются для различного рода автоматов: ЭВМ, станков с числовым программным управлением (ЧПУ), промышленных роботов. Автоматы обычных человеческих языков не понимают. Поэтому алгоритмы для таких исполнителей пишутся на специальных языках, которые называются языками программирования. Программа — это и есть алгоритм, записанный на языке программирования.

Определение. Алгоритм, записанный на языке программирования, называется программой.

Для ЭВМ создано очень много языков программирования. Это обусловлено тем, что современные компьютеры применяются для решения самых разнообразных задач из многих областей человеческой деятельности: это математические расчеты и математическое моделирование, информационно-поисковые системы и передача информации по компьютерным сетям, компьютерная графика и компьютерная мультипликация. Ни один язык программирования не может применяться во всех этих областях. Для каждой области создается свой особый язык, предназначенный для решения определенного круга задач (для определенной проблемы). Поэтому современные языки программирования называют проблемно-ориентированными языками.

Проблемно-ориентированные языки обладают одним ценным для программиста свойством: они позволяют не думать, для какого именно компьютера мы составляем программу. Если мы пишем программу на какой-либо версии Бейсика, то эта программа будет работать на любом компьютере, где имеется транслятор (переводчик) этого языка, будь то компьютер фирмы IBM или фирмы «Apple», «Искра» или «Ямаха». Получается так, что мы программируем для какого-то абстрактного исполнителя, который понимает язык Бейсик. Называют этого исполнителя Бейсик-машина. Если мы программируем на другом языке, например на Паскале, то исполнителем нашего алгоритма будет Паскаль-машина и т.п.

Мы будем писать программы на языке программирования QBASIC, который представляет собой одну из версий популярного языка BASIC (Бейсик). Этот язык был создан в 1965 году в Дартмуртском колледже (США) и предназначался для обучения студентов началам программирования. Название языка BASIC представляет собой сокращение (аббревиатуру) английского выражения *Beginner's All-purpose Symbolic Instruction Code*, что переводится как «многоцелевой символический код для начинающих».

Языки программирования отличаются от обычных человеческих языков очень строгим правописанием. Если в учебниках математики операция умножения может обозначаться по-разному (крестиком, точкой или вообще пропускаться), то на языке QBASIC умножение обозначается только звездочкой «*». Пропускать этот значок в формулах нельзя. Формула $(a + b)(c + d)$ на языке QBASIC запишется как $(a + b)*(c + d)$. Если звездочку пропустить, то компьютер такую формулу просто не поймет.

Деление обозначается только косой чертой «/». Двоеточием «:» деление на языке QBASIC обозначать нельзя. Все математические выражения записываются в строчку, поэтому дробь $\frac{2}{3}$ на языке QBASIC запишется как $2/3$, а число $5\frac{2}{3}$ можно записать как $5 + 2/3$.

Переведем на язык QBASIC алгоритм решения задачи из предыдущего параграфа. Входные данные: числа 120, 8, 140, 6.

Обозначим количество ящиков, привезенных на базу в первый день, именем X1. Тогда первый пункт алгоритма будет выглядеть так:

1. Умножить 120 на 8, результат запомнить в переменной X1.

На языке QBASIC эта строчка выглядит так:

$X1 = 120 * 8.$

Решая задачу, мы записываем промежуточные результаты на бумагу. Компьютер записывает промежуточные данные во внутреннюю память. Для записи данных во внутреннюю память служат переменные. Переменная — это величина, записанная во внутреннюю память под определенным именем. В результате выполнения 1-й команды в памяти компьютера возникнет переменная с именем X1 и значением 960:

X1	960
----	-----

Переменную можно представить в виде листка бумаги, на котором записано значение 960 и проставлено имя X1.

Обозначим количество ящиков, привезенных на базу во второй день, именем X2. Тогда второй пункт алгоритма запишется так:

2. Умножить 140 на 6 и запомнить в переменной X2.

На языке QBASIC:

$X2 = 140 * 6.$

В памяти компьютера появится вторая переменная:

X1	960
X2	840

Третий пункт алгоритма:

3. Сложить X1 и X2, результат запомнить в переменной S.

$S = X1 + X2.$

В результате выполнения этого пункта в памяти возникнет переменная S:

X1	960
X2	840
S	1800

Значение переменной S — это выходные данные алгоритма. Это значение нужно вывести на экран. Команда вывода на экран переводится на язык QBASIC словом PRINT.

4. Вывести на экран значение переменной S.

PRINT S.

Теперь запишем алгоритм только на языке QBASIC:

$X1 = 120 * 8.$

$X2 = 140 * 6.$

$S = X1 + X2.$

PRINT S.

Получилась программа, решающая нашу задачу.

Как вы уже знаете, алгоритм состоит из команд на выполнение математических операций. Программа — это тоже алгоритм, и она также состоит из команд. Команда, записанная на языке программирования, называется оператором. Наша программа состоит из операторов двух типов. Первые три оператора — это операторы присваивания. Они дают компьютеру команду: занести в переменную некоторое значение. Например, первый оператор дает команду: занести в переменную X1 значение 960.

Последний оператор — оператор вывода на экран. Он дает команду: вывести на экран значение переменной S.

Решим при помощи компьютера еще одну задачу. Длина класса 7 метров, ширина — 5 метров, высота — 3 метра. В классе учатся 20 учеников. Сколько квадратных метров площади и сколько кубических метров воздуха приходится на одного ученика?

Решение задачи:

1. Вычислить площадь класса:

$$7 \times 5 = 35 \text{ кв. м.}$$

2. Вычислить, сколько кв. метров площади приходится на одного ученика:

$$35 : 20 = 1,75 \text{ кв. м.}$$

3. Вычислить объем класса:

$$35 \times 3 = 105 \text{ куб. м.}$$

4. Вычислить, сколько куб. метров воздуха приходится на одного ученика:

$$105 : 20 = 5,25 \text{ куб. м.}$$

Алгоритм решения задачи.

Входные данные: числа 7, 5, 3, 20.

1. Умножить 1-е число на 2-е (вычислить площадь класса).

2. Разделить результат на 5-е число (вычислить, сколько кв. метров площади приходится на одного ученика).

3. Умножить результат 1-й операции на 3-е число (вычислить объем класса).

4. Разделить результат на 5-е число (вычислить, сколько куб. метров воздуха приходится на одного ученика).

Перевод алгоритма на язык QBASIC:

```
S = 5*7
```

```
S1 = S / 20
```

```
V = S * 3
```

```
V1 = V / 20
```

```
PRINT S1
```

```
PRINT V1.
```

Как вы уже, наверное, поняли, хотя мы говорим «программа решает задачу», на самом деле задачу решает человек. Программа — это просто перечисление операций, которые нужно выполнить, чтобы получить результат. Написать эти операции нужно самому, иначе компьютер ничего не посчитает.

Зачем же нужен компьютер, если все приходится делать самому? Если все операции приходится записывать, так, наверное, их проще сразу же и выполнить, а не составлять программу для компьютера? Конечно, для решения тех задач, которые мы сейчас разбираем, компьютер не нужен. Однако современная наука и техника ставит перед учеными и инженерами такие задачи, которые невозможно решить без компьютера. Например, при проектировании сверхзвуковых самолетов приходится производить такие расчеты, над которыми компьютер, делающий 100 млн операций в секунду, работает 2 дня. Так что без компьютеров сверхзвуковой авиации просто не было бы.

Мы решаем на компьютере простые задачи для того, чтобы научиться языку программирования. В дальнейшем мы перейдем к более сложным задачам, которые без компьютера не решишь.

§ 5. Ввод данных

Посмотрим на рисунок, приведенный в § 1. Алгоритм получает входные данные извне и выводит «наружу» выходные данные. Для вывода данных на экран в языке QBASIC имеется оператор PRINT. Для ввода данных также имеется специальный оператор:

```
INPUT X.
```

Этот оператор дает машине команду: «Ввести с клавиатуры значение переменной X». Работу этого оператора рассмотрим на примере последней задачи из предыдущего параграфа. В программе, приведенной в § 4, размеры класса и количество учеников в классе уже написаны. Если мы хотим решить задачу с другими входными данными (например, взять количество учеников в классе не 20, а 25 или взять класс с другими размерами), то придется изменять программу. Если же мы поставим в начале программы оператор ввода данных, то она будет универсальной — при помощи такой программы можно будет рассчитывать площадь класса и объем воздуха, приходящиеся на одного ученика для классных комнат любых размеров и для любого количества учеников в классе.

Обозначим длину класса X1, ширину — X2, высоту — X3, количество учеников в классе — K. Тогда программа будет выглядеть следующим образом:

```
INPUT X1
INPUT X2
INPUT X3
INPUT K
S = X1 * X2
V = S * X3
S1 = S / K
```

```
V1 = V / K
PRINT S1
PRINT V1.
```

При выполнении 1-й строки программы на экране возникнет вопросительный знак. Это означает, что компьютер остановил выполнение программы и ждет наших действий: мы должны набрать на клавиатуре значение длины класса и нажать клавишу «Enter». Если у нас длина класса равна 7. Нажимаем «7» и «Enter». В памяти компьютера возникнет переменная X1 со значением «7»:

X1	7
----	---

После этого будет выполняться 2-я строка программы, на экране опять появится «?»: компьютер запросит значение переменной X2. Переменная X2 — это ширина класса, которая у нас равна 5. Нажимаем «5» и «Enter». В памяти компьютера возникнет еще одна переменная:

X1	7
X2	5

После выполнения всех строчек, в которых стоит оператор INPUT, в памяти компьютера появятся 4 переменных:

X1	7
X2	5
X3	3
K	20

В 5-й строке программы стоит команда: «Умножить значение переменной X1 на значение переменной X2, результат запомнить в переменной S». При выполнении этой команды в памяти возникнет переменная S со значением «35». Аналогичным образом после выполнения 6-й, 7-й и 8-й строчек программы в памяти возникнут переменные V, S1, V1:

X1	7
X2	5
X3	3
K	20
S	35
V	105
S1	1.75
V1	5.25

Здесь X1, X2, X3, K — входные данные, S1 и V1 — выходные данные, S, V — промежуточные данные.

Две последние строки программы дают команды: «Вывести на экран значение переменной S1» и «Вывести на экран значение переменной V1». В результате на экране появится ответ задачи:

1.75
5.25.

При помощи этой программы можно будет рассчитать площадь и объем воздуха, приходящегося на одного ученика не только для данной классной комнаты и для данного класса из 20 человек, но и для любых размеров классной комнаты и для класса с любым количеством учеников — нужно будет только ввести соответствующие данные. В таких случаях говорят, что программа решает не одну задачу, а класс задач с одним и тем же условием, но различными входными данными. Свойство алгоритма решать целый класс задач с различными входными данными называется массовостью алгоритма.

§ 6. Постановка задачи

Обычно в математике решение задачи представляется в виде формулы или в виде чертежа. При использовании компьютера решение задачи записывается в виде алгоритма. Алгоритм, как мы знаем, обрабатывает инфор-

мацию, преобразует входные данные в выходные. Прежде чем браться за составление алгоритма, необходимо определить, какие данные являются входными, какие выходными. Для этого из условия задачи нужно выделить, что дано и что требуется сделать. На языке математики формулировка условия задачи в виде «Дано»... «Требуется» называется постановкой задачи.

Рассмотрим с этой точки зрения наши задачи: сначала сформулируем их в виде «Дано»... «Требуется», затем выделим входные и выходные данные, затем составим алгоритм и напишем программу.

Задача 1. Дано: количество машин, пришедших на базу в 1-й день; количество ящиков на каждой из машин, пришедших на базу в 1-й день; количество машин, пришедших на базу во 2-й день, количество ящиков на каждой из машин, пришедших на базу во 2-й день.

Требуется: вычислить, сколько ящиков привезли на базу за 2 дня.

Алгоритм решения задачи.

Входные данные: $A1$ — количество машин, пришедших на базу в 1-й день;

$B1$ — количество ящиков на каждой из машин, пришедших на базу в 1-й день;

$A2$ — количество машин, пришедших на базу во 2-й день;

$B2$ — количество ящиков на каждой из машин, пришедших на базу во 2-й день.

Выходные данные: S — количество ящиков, привезенных на базу за 2 дня.

1. Ввести $A1$, $B1$, $A2$, $B2$.

2. Умножить $A1$ на $B1$, результат запомнить в переменной $X1$.

3. Умножить $A2$ на $B2$, результат запомнить в переменной $X2$.

4. Сложить $X1$ и $X2$, результат запомнить в переменной S .

5. Вывести на экран значение переменной S .

Программа.

```
INPUT A1
```

```
INPUT B1
```

```
INPUT A2
```

```
INPUT B2
```

```
X1= A1 * B1
```

```
X2 = A2 * B2
```

```
S = X1+X2
```

```
PRINT S.
```

Задача 2. Дано: длина класса, ширина класса, высота класса, количество учеников в классе.

Требуется: вычислить количество кв. метров площади, приходящихся на одного ученика; вычислить объем воздуха, приходящегося на одного ученика.

Алгоритм решения задачи.

Входные данные: $X1$ — длина класса;

$X2$ — ширина класса;

$X3$ — высота класса;

K — количество учеников в классе.

Выходные данные: $S1$ — площадь, приходящаяся на одного ученика;
 $V1$ — объем воздуха, приходящийся на одного ученика.

1. Ввести $X1$, $X2$, $X3$, K .
2. Умножить $X1$ на $X2$, результат запомнить в переменной S .
3. Умножить S на $X3$, результат запомнить в переменной V .
4. Разделить S на K , результат запомнить в переменной $S1$.
5. Разделить V на K , результат запомнить в переменной $V1$.
6. Вывести на экран значение переменной $S1$.
7. Вывести на экран значение переменной $V1$.

Программа.

```
INPUT X1
INPUT X2
INPUT X3
INPUT K
S = X1 * X2
V = S * X3
S1 = S / K
V1 = V / K
PRINT S1
PRINT V1.
```

Глава II

Основные понятия программирования

§ 1. Программа и операторы

Любой человеческий язык — это средство общения между людьми. Язык программирования — это средство общения между человеком и компьютером. Язык программирования служит для того, чтобы сообщить компьютеру алгоритм решения задачи.

Как мы уже знаем, алгоритм, написанный на языке программирования, называется программой. Программа — это человеческая речь, обращенная к компьютеру на понятном ему языке. Всякая человеческая речь состоит из предложений. Программа также состоит из предложений. Предложения языка программирования называются операторами (иногда их называют инструкциями).

Определение. Оператор — предложение языка программирования.

Вы уже знаете, что алгоритм состоит из команд. Следовательно, операторы языка программирования — это тоже команды. Каждый оператор — это перевод какой-либо команды алгоритма на язык, понятный компьютеру. Читая программу, компьютер выполняет команды, которые записаны при помощи операторов.

Вы уже знакомы с тремя операторами: оператором присваивания, оператором ввода данных с клавиатуры и оператором вывода данных на экран. Познакомимся с еще одним оператором:

CLS.

Этот оператор дает команду: очистить экран. Его ставят в начале программы, чтобы компьютер стер с экрана посторонние надписи и на экране остались бы только результаты работы программы.

Теперь напишем маленькую программку, которая будет умножать на 2 число, вводимое с клавиатуры, и результат выводить на экран:

```
CLS
INPUT X
A = X*2
PRINT A
```

Когда программа начнет работать, компьютер выполнит команду, которую дает 1-й оператор: «Очистить экран», и с экрана исчезнут все посторонние надписи, оставшиеся от предыдущей работы компьютера. Вторым оператором дается команда: «Ввести с клавиатуры значение переменной X». На этом месте программа остановит работу, на экране появится «?», и компьютер будет ждать, когда мы наберем на клавиатуре нужное число. После нажатия клавиши «Enter» набранное на клавиатуре число будет занесено в переменную X, затем это число удвоится и результат будет записан в переменную A. Наконец, значение переменной A будет выведено на экран.

Операторы в программе выполняются в том порядке, в котором они записаны. Этот порядок называется естественным ходом выполнения программы. Естественный ход выполнения программы может нарушаться при помощи так называемых операторов перехода, с которыми мы познакомимся позже.

Обычно операторы записываются по одному в строке — так их легче читать, особенно если операторы длинные. Но если операторы короткие, то в одной строке можно записать несколько операторов. В этом случае их разделяют двоеточием. Нашу маленькую программку можно записать в одну строчку следующим образом:

```
CLS: INPUT X: A = X*2: PRINT A.
```

Операторы, записанные в одной строке, выполняются последовательно слева направо.

Как и во всяком другом языке, предложения в QBASIC пишутся по определенным правилам. Совокупность этих правил называется синтаксисом языка. Правила синтаксиса языка программирования гораздо строже правил любого из обычных человеческих языков. Во-первых, правила синтаксиса языка программирования не знают исключений. Во-вторых, языки программирования не терпят синтаксических ошибок. Если при общении между людьми синтаксические ошибки зачастую прощаются, то при общении человека с ЭВМ синтаксические ошибки недопустимы. Программу, написанную с синтаксическими ошибками, машина просто не поймет.

В этой главе мы познакомимся с правилами написания некоторых операторов языка QBASIC.

§ 2. Переменные и идентификаторы

Определение. Переменная — область памяти, которой присвоено имя.

Понятие переменной в программировании отличается от понятия переменной в математике. В математике переменная — это величина, которая может принимать различные значения. В программировании переменная — это целая конструкция, которую можно изобразить следующим образом:

Имя	Значение
A	6

ПЕРЕМЕННАЯ

Переменная состоит из двух частей — имени и значения. Значение переменной — это та величина, с которой работает машина. Имя переменной — это адрес участка памяти, где записана эта величина. Память машины можно представить в виде книги, состоящей из страниц. Только страницы этой книги не пронумерованы числами, а размечены именами. На каждой странице стоит свое имя, и на каждой странице записана только одна величина.

Когда мы читаем книгу, то содержимое страниц книги не меняется. Точно так же не меняется содержимое переменной при считывании информации. Однако при записи нового значения в переменную старое значение стирается.

Имена переменных в программировании называют идентификаторами. В математике не существует строгих правил для записи имен переменных. По традиции переменную обозначают одной или двумя буквами или буквой с цифрой. В программировании, напротив, существуют строгие правила, которые надо соблюдать при написании имен переменных (наверное, поэтому программисты называют имена переменных так почтительно — идентификаторы).

Первое правило заключается в том, что для написания идентификаторов в QBASIC можно использовать только латинские буквы, цифры и точки. Никаких других значков, которые можно получить на экране, при написании идентификаторов использовать нельзя. При этом большие (прописные) и маленькие (строчные) буквы не различаются. Каждый идентификатор содержит обязательно хотя бы одну букву. Цифры и точки можно писать в идентификаторах, а можно не писать.

Идентификаторами могут служить следующие последовательности символов: AA, C, A10, TANYA, SCHOOL77, PETYA.IVANOV, A2B3. Поскольку прописные и строчные буквы не различаются, то последний идентификатор можно записать как a2b3 или даже A2b3. Нельзя писать A!23 или ABC?, поскольку знаки «!» и «?» не являются ни буквами, ни цифрами.

При написании имен переменных иногда используют символы !, #, \$, %, &, которые всегда пишутся крайними справа. Эти символы называются суффиксами и используются для обозначения типа переменной. Например, запись A%=1 означает, что переменная с именем A имеет тип «целый». С типами переменных мы познакомимся позднее.

Второе правило: на первом месте должна стоять обязательно буква. Поэтому надпись 1A не является идентификатором (на первом месте стоит не буква).

Третье правило говорит, что при написании идентификаторов можно использовать до 40 символов. Иногда в книгах пишут: не более 40 символов (как будто кто-то захочет написать идентификатор, состоящий из 100

символов. Всякий нормальный человек написал бы: «Можно использовать аж 40 символов!!!»).

Возможность использовать такое большое количество символов в идентификаторе позволяет давать переменным «осмысленные» имена. Например, если мы хотим некоторой переменной присвоить значение скорости 60 км/ч, то можно назвать эту переменную `skorost` и написать оператор `skorost=60`. Тогда при чтении программы будет ясно, что данная переменная имеет смысл скорости.

Все три правила можно объединить в одну фразу и дать следующее определение.

Определение. Идентификатор — это последовательность букв, цифр и точек, начинающаяся с буквы и содержащая до 40 символов.

§ 3. Арифметическое выражение

Из курса математики вам известно понятие «выражение». Обычно в школе работают с алгебраическими выражениями. Алгебраическое выражение состоит из чисел, имен переменных, знаков арифметических операций и скобок. В программировании такое выражение называется арифметическим выражением, поскольку значением такого выражения является число. При составлении программ используются еще и логические и символьные выражения, с которыми мы познакомимся позднее. Рассмотрим пока правила записи арифметических выражений. Правила эти следующие:

1. При записи десятичных дробей используется не десятичная запятая, а десятичная точка.

2. Все арифметические операции записываются в строчку. Если в числителе или знаменателе дроби стоит многочлен, его следует заключить в скобки.

3. Знаки арифметических операций опускать нельзя.

4. Арифметические операции выполняются слева направо в порядке приоритета. Приоритет арифметических операций такой же, как и в арифметике:

сначала выполняется возведение в степень, затем умножение и деление; затем сложение и вычитание.

Для изменения естественного приоритета операций применяются скобки (так же, как и в арифметике).

Для обозначения арифметических операций используются следующие знаки:

- $\wedge$ — возведение в степень,
- $*$ — умножение,
- $/$ — деление,
- $+$ — сложение,
- $-$ — вычитание.

Если в дробном числе нет целой части, то ноль перед десятичной точкой не пишется. Если мы напишем ноль перед десятичной точкой, то машина сама отбросит его. Машина сама отбросит также нули, стоящие в

дробной части в конце числа. Если в программе написать $x = 0.500$, то при вводе этой строки на экране появится $x = .5$.

При записи арифметических выражений необходимо строго следовать правилам синтаксиса, ничего не опускать и не допускать никаких отклонений. Если при записи формул в математике можно использовать разные знаки деления: косая черта «/», двоеточие «:», горизонтальная черта, то при записи арифметических выражений в программировании используется только один знак — косая черта.

Если в математике при записи формул знак умножения обычно опускается, то в программировании при записи арифметических выражений знак умножения опускать нельзя. Запись АВ машина воспримет не как арифметическую операцию умножения двух переменных, а как идентификатор переменной.

Рассмотрим несколько примеров.

Дробь $\frac{x}{a+b}$ нужно записывать как $x/(a+b)$. Если записать $x/a+b$, то сначала будет выполнено деление x/a , а затем к полученному результату будет прибавлено b . В результате получится $\frac{x}{a} + b$.

Дробь $\frac{x}{ab}$ можно записать как $x/(a*b)$ или $x/a/b$.

В простейшем случае арифметическое выражение может представлять собой просто число или идентификатор.

§ 4. Оператор присваивания

Определение. Процесс занесения значения в переменную называется присваиванием.

В языке QBASIC имеется три оператора, позволяющие занести значение в переменную (т. е. осуществить присваивание). Один из них так и называется — оператор присваивания. Полная запись (формат) оператора:

LET идентификатор = выражение .

Служебное слово LET по-английски значит «пусть». Таким образом, перевод оператора LET $A = 2*3$ звучит так: «Пусть переменной А присваивается значение, равное результату умножения $2*3$ ».

В первоначальном варианте Бейсика любой оператор должен был начинаться со служебного слова. Впоследствии Бейсики стали конструировать так, чтобы в операторе присваивания служебное слово LET можно было не писать. Поэтому формат оператора присваивания изображают следующим образом:

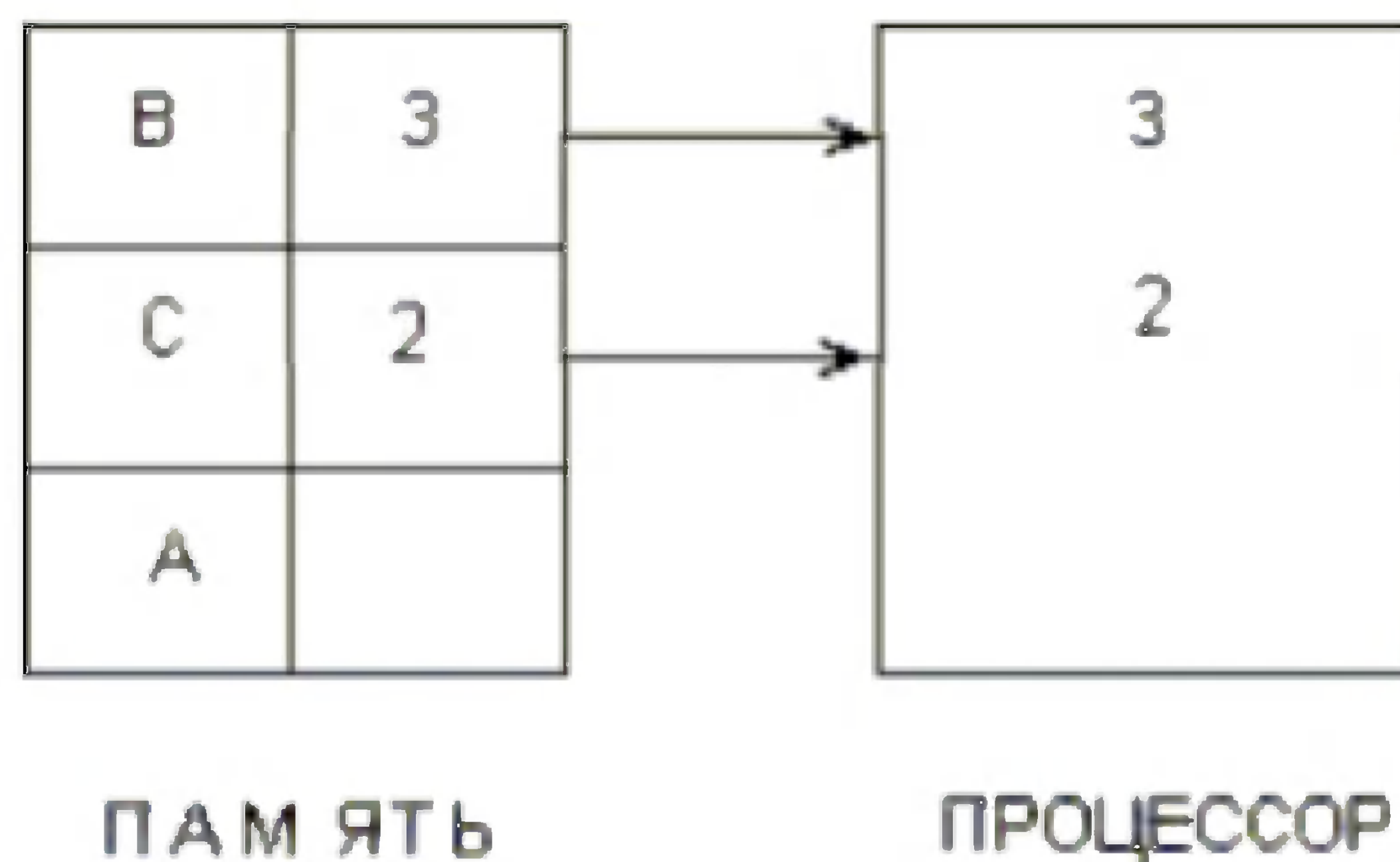
[LET] идентификатор = выражение .

В квадратные скобки заключен необязательный элемент — т. е. слово, которое можно не писать в программе.

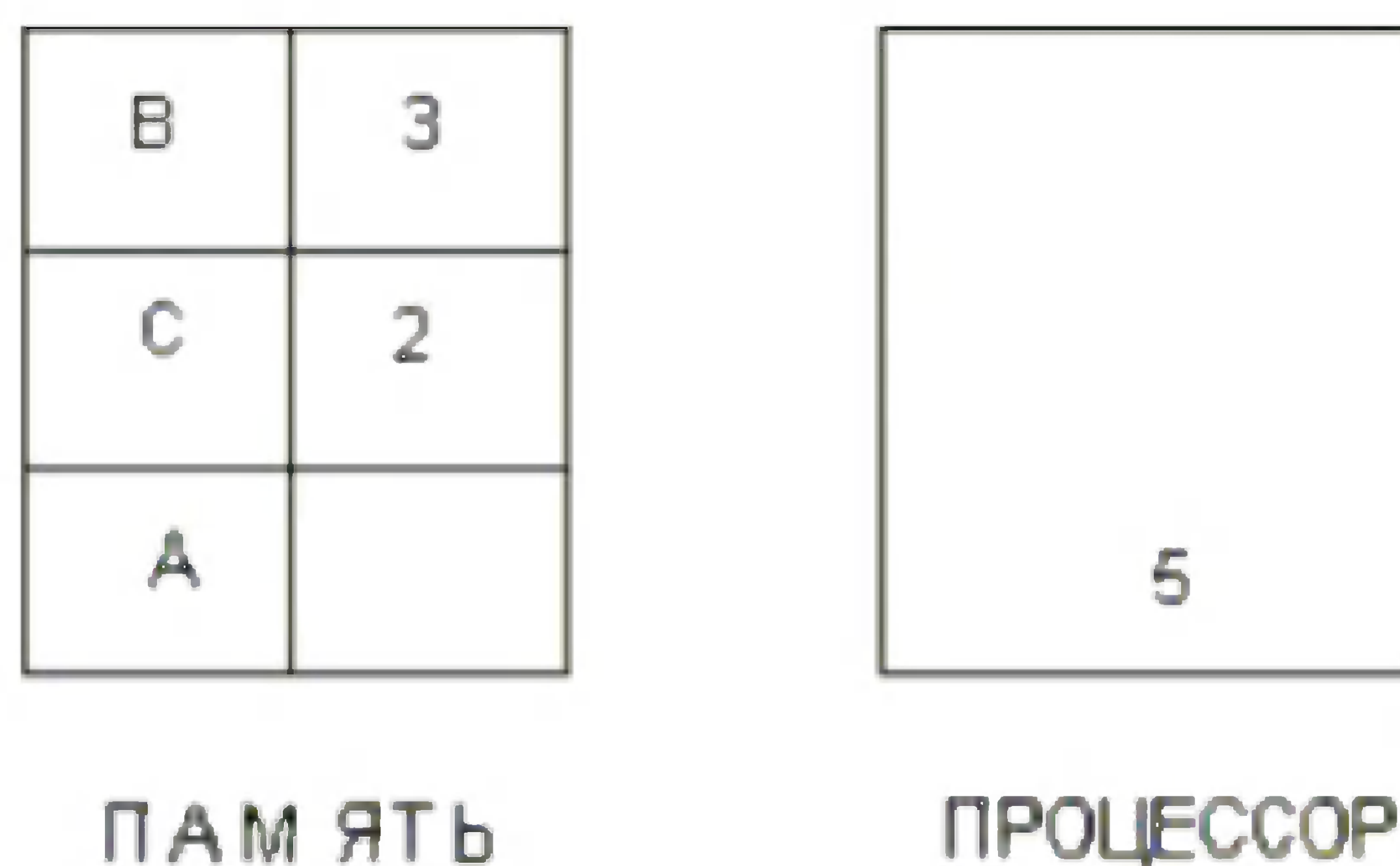
Выполняется оператор присваивания справа — налево: сначала выполняются действия, стоящие справа от знака равенства, затем результат заносится в переменную, имя которой стоит слева от знака равенства. Рассмотрим, что происходит в памяти машины при выполнении оператора:

$$A = B + C.$$

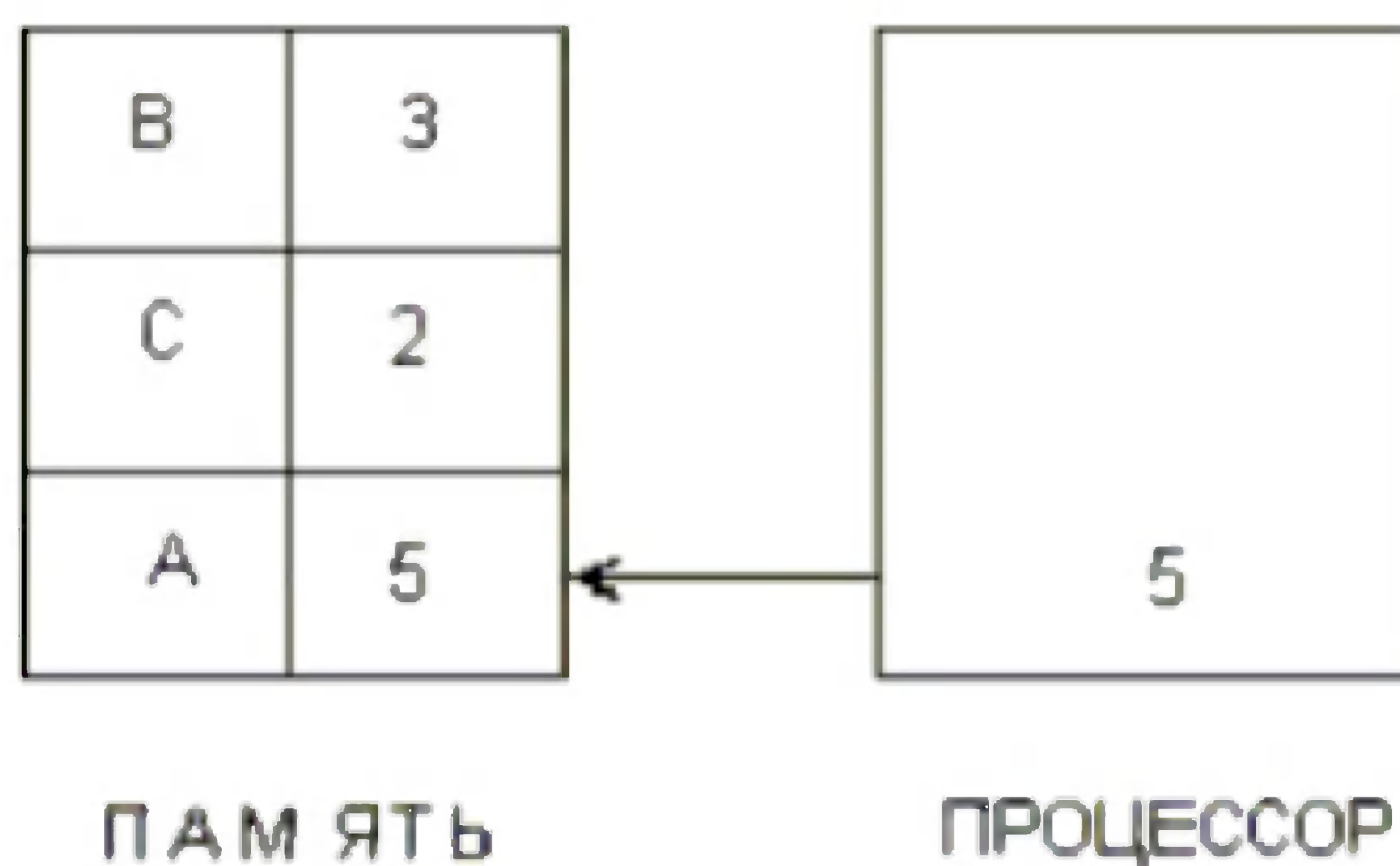
Сначала из памяти извлекаются значения переменных B и C и помещаются в процессор:



Затем эти значения складываются



и результат записывается в переменную с именем A :



При этом один и тот же идентификатор может стоять и в левой, и в правой части оператора присваивания, например:

$A = A + 1.$

В математике такая формула просто не имеет смысла. Однако это не формула, это оператор присваивания. Выполняя этот оператор, машина извлечет из памяти число, записанное в переменной A , увеличит его на единицу и запишет в ту же переменную. Перевести на русский язык этот оператор можно так: «Увеличить значение переменной A на 1».

Как видим, оператор присваивания — это не математическое равенство. Оператор присваивания заставляет машину совершать множество операций: извлекать из памяти значения переменных, имена которых стоят справа от знака равенства, выполнять арифметические операции и записывать результат в переменную, имя которой стоит слева от знака равенства. Поэтому значок « $=$ » читается не «равно», а «присваивается». Например, оператор $A = 2 + 3$ переводится на русский язык как «переменной A присваивается значение, равное $2+3$ », а не как « A равно $2+3$ ».

§ 5. Ввод данных с клавиатуры

Числа, с которыми работает машина, в программировании называются данными. Те числа, которые вводятся в память в начале выполнения программы, носят название входных данных. Те числа, которые выводятся на экран как результат работы программы, называются выходными данными. Числа, которые циркулируют внутри машины в процессе работы — это промежуточные данные.

Занести в память компьютера входные данные можно при помощи оператора присваивания. Но оператор присваивания обычно используется для работы с промежуточными данными. Для ввода исходных данных применяется оператор ввода данных с клавиатуры или оператор чтения данных.

С оператором ввода данных с клавиатуры мы уже знакомы. В простейшем случае выглядит этот оператор так:

`INPUT A.`

Бейсиковское слово `INPUT` происходит от английского выражения `to put in` — вставлять, входить. Выражение `INPUT A` можно перевести на русский язык как «Ввести в память значение переменной A ». Когда машина выполняет этот оператор, на экране появляется «?», и выполнение программы прерывается: машина ждет наших действий. Теперь мы должны набрать на клавиатуре нужное число и нажать клавишу «Enter». Например, если мы наберем на клавиатуре число 16 и нажмем «Enter», то в переменную A будет записано число 16. Результат будет такой же, как при выполнении оператора присваивания $A = 16$.

Если нужно присвоить значения нескольким переменным (например, трем), то имена всех переменных можно указать в одном операторе ввода:

`INPUT A, B, C.`

В этом случае после появления на экране «?» необходимо набрать три числа через запятую, например:

6, 15, 18

и нажать клавишу «Enter». Тогда переменной A будет присвоено значение 6, переменной B — 15, переменной C — 18.

Когда работает оператор INPUT, на экране появляется один вопросительный знак независимо от того, сколько идентификаторов написано в нем. При работе программы будет неясно, сколько же чисел нужно вводить. Поэтому в операторе INPUT предусмотрена возможность вывода подсказки на экран. Подсказка пишется в кавычках перед именами вводимых переменных. В нашем примере можно написать:

INPUT «Введите три числа»; A, B, C.

Тогда при работе программы на экране появится надпись:

Введите три числа?

Знак вопроса по-прежнему означает, что машина ждет ввода чисел с клавиатуры.

Такой режим работы называется режимом диалога между машиной и человеком. Машина сообщает нам, что нужно сделать в данный момент, а мы в ответ производим необходимые действия. Режим диалога называют также интерактивным режимом.

В заключение напишем формат оператора ввода данных с клавиатуры:

INPUT «подсказка»; список идентификаторов.

§ 6. Чтение данных

Еще одну возможность присвоить переменной значение предоставляет оператор чтения данных. Формат оператора:

READ список идентификаторов
DATA список значений.

Оператор пишется в двух строчках. Несмотря на это, оператор чтения данных представляет собой единый оператор READ... DATA.

Работает оператор следующим образом. В строчке READ пишутся имена переменных, которым мы хотим присвоить значения. Напишем для начала одну переменную:

READ A.

Прочитав эту строчку, машина ищет строку со словом DATA. Это слово может стоять в любом месте программы, необязательно рядом со словом READ. Обычно строку DATA ставят в начале или в конце программы, чтобы эта строка не мешала читать основной текст.

Машина просматривает все строки программы, начиная с первой. Найдя строку DATA, она заносит стоящую там величину в переменную A. Если мы напишем

```
DATA 16,
```

то переменная A примет значение 16.

В строчке READ можно поставить несколько имен. Тогда в строчке DATA нужно ставить столько же чисел:

```
READ A,B,C  
DATA 15,18,8.
```

При запуске программы первое число отмечается специальным указателем, который называется «счетчик данных». Когда оператор чтения данных начинает работу, то число, отмеченное счетчиком данных, заносится в первую переменную, а счетчик данных перескакивает на следующее число. Машина читает следующий идентификатор в строчке READ, заводит соответствующую переменную и заносит в нее число, на которое указывает счетчик данных, и т. д.

Если в строчке DATA будет записано больше чисел, чем идентификаторов в строчке READ, то ничего страшного не произойдет: счетчик данных остановится на очередном числе и будет находиться там до конца выполнения программы. А вот если чисел будет меньше, чем идентификаторов, то при записи очередной переменной машина не обнаружит соответствующего значения и выдаст сообщение об ошибке.

При использовании оператора READ... DATA все исходные данные сразу заносятся в программу и машина работает в автономном режиме, т. е. самостоятельно; вмешательства человека при этом не требуется. Этот оператор удобно использовать при отладке программ.

§ 7. Вывод данных на экран

В результате работы программы мы получаем число или несколько чисел, которые хранятся в соответствующих переменных. Для того чтобы увидеть результаты работы программы, необходимо вывести значения этих переменных на экран. Вывод данных на экран производится оператором вывода. Если мы хотим вывести на экран значение переменной A, то нужно написать:

```
PRINT A.
```

Тогда на экране появится значение переменной A.

В операторе PRINT можно написать несколько идентификаторов. При этом на экране появятся значения соответствующих переменных, записанные в строчку. Разделять идентификаторы можно запятой, можно точкой с запятой. Разница в том, что при разделении идентификаторов запятыми числа на экране будут отстоять дальше друг от друга.

Если мы напишем в операторе вывода только идентификаторы, то на экране появятся только числа. Не всегда бывает ясно, что эти числа означают. Для того, чтобы вывод был более ясен, в операторе PRINT преду-

смотрена возможность вывода поясняющих надписей. Надписи, как и в операторе INPUT, заключаются в кавычки. Если в переменной A записано значение 15 и мы напишем оператор

```
PRINT «A =»; A,
```

то на экране получим

```
A = 15.
```

Такой вывод более понятен, чем просто число 15.

Конечно, в кавычках можно писать не только имя выводимой переменной, но и любую поясняющую надпись. Например, если в переменной A записана скорость, то можно написать

```
PRINT «Скорость равна»; A.
```

В результате на экране получим:

```
Скорость равна 15.
```

Можно после идентификатора A указать размерность скорости:

```
PRINT «Скорость равна»; A; «км/ч».
```

Тогда получится:

```
Скорость равна 15 км/ч.
```

Оператор вывода имеет еще одну возможность. Он может выводить на экран результаты арифметических операций. В оператор PRINT можно записать арифметическое выражение, результат будет выведен на экран. Например, в результате выполнения оператора PRINT 6*3 на экране появится число 18.

Следующая маленькая программка выводит на экран среднее арифметическое двух чисел, вводимых с клавиатуры:

```
CLS  
INPUT «Введите два числа»; A,B  
PRINT «Среднее арифметическое чисел»; A,B; «равно»;  
(A+B)/2.
```

Здесь в операторе PRINT присутствуют все три описанных нами типа выводимых величин: надписи, значения переменных, значение арифметического выражения. Весь этот набор объединяют одним названием: список вывода. Формат оператора вывода на экран можно записать следующим образом:

```
PRINT список вывода.
```

Оператор PRINT может и не содержать список вывода. Такой оператор просто перебрасывает курсор на новую строку. Его можно использовать для разделения выводимых надписей. Если мы напишем:

```
PRINT A; B
PRINT C,
```

то получим на экране вывод чисел в двух строках:

```
15 8
7.
```

Если же мы напишем

```
PRINT A; B
PRINT
PRINT C,
```

то между верхней и нижней строками появится пустая строка:

```
15 8

7.
```

§ 8. Координатный вывод

Когда мы выводим значение переменной при помощи оператора PRINT, то числа появляются в левом верхнем углу экрана. Это не всегда удобно. Иногда бывает необходимо вывести число или надпись в каком-то другом месте экрана. В QBASIC имеется оператор координатного вывода, который позволяет указать то место на экране, куда будет производиться вывод. Выглядит этот оператор так:

LOCATE номер строки, номер столбца.

Всего на экране помещается 25 строк. В каждой строке можно разместить 80 символов. Оператор LOCATE перебрасывает курсор в ту позицию на экране, координаты которой указаны в качестве параметров. Если после оператора LOCATE написать оператор PRINT, то выводимое значение появится в том месте экрана, где находится курсор. Например, если мы хотим вывести значение переменной A в центре экрана, то нужно написать:

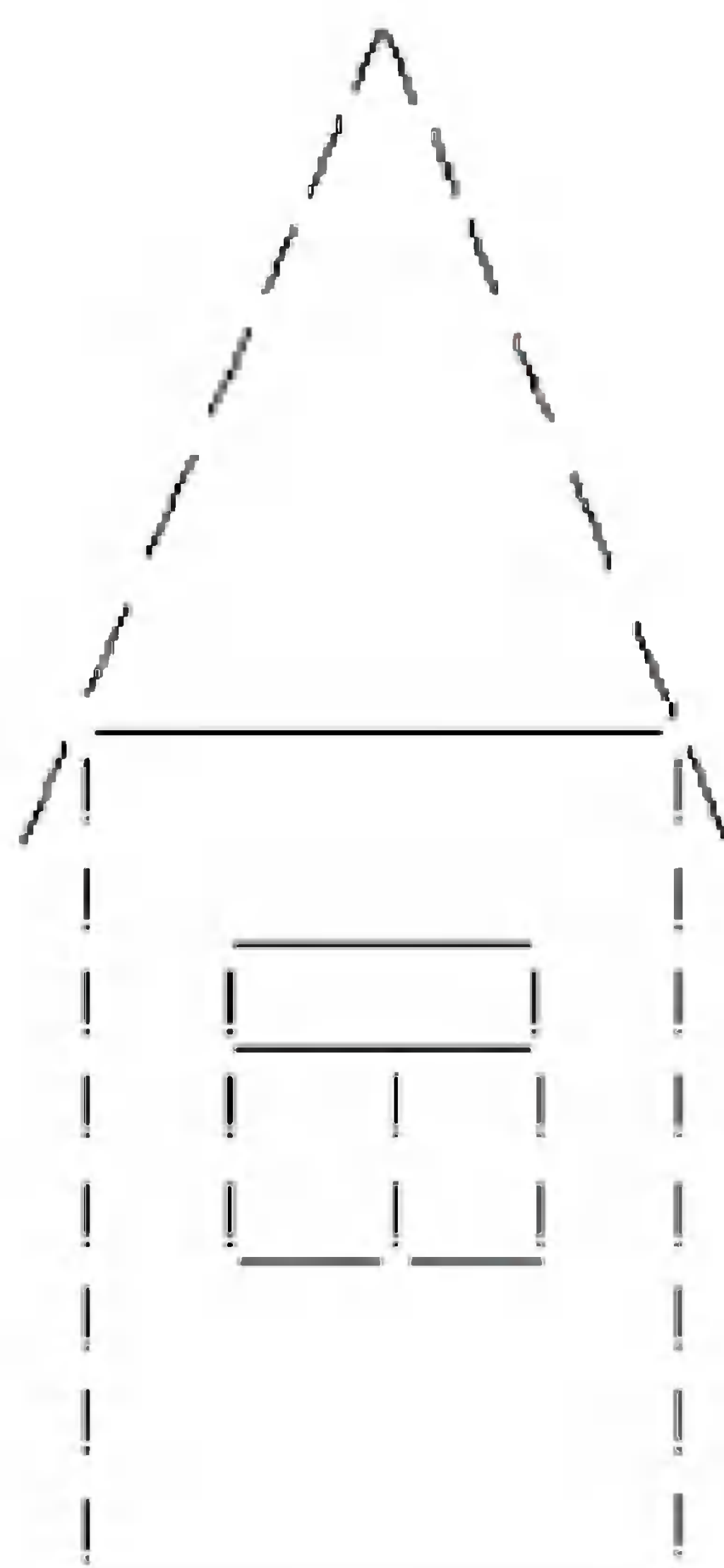
```
LOCATE 12, 40
PRINT A.
```

При помощи оператора LOCATE можно изобразить на экране несложные рисунки. Например, следующая программа рисует прямоугольник:

```
CLS
LOCATE 9, 20
PRINT «_____»
LOCATE 15, 20
PRINT «_____»
LOCATE 10, 19
PRINT «!»
LOCATE 11, 19
PRINT «!»
```

```
LOCATE 12, 19
PRINT «!»
LOCATE 13, 19
PRINT «!»
LOCATE 14, 19
PRINT «!»
LOCATE 15, 19
PRINT «!»
LOCATE 10, 42
PRINT «!»
LOCATE 11, 42
PRINT «!»
LOCATE 12, 42
PRINT «!»
LOCATE 13, 42
PRINT «!»
LOCATE 14, 42
PRINT «!»
LOCATE 15, 42
PRINT «!»
```

В качестве упражнения попробуйте изобразить на экране такой домик:



После первых двух параметров в операторе `LOCATE` (номер строки и номер столбца) можно указывать третий параметр, который определяет режим отображения курсора на экране. Если в качестве третьего параметра написать 0, то курсор будет виден на экране, если написать 1, то курсор не будет виден.

Четвертый и пятый параметры в операторе `LOCATE` позволяют изменять размер курсора. Оба эти параметра могут принимать значения от 0 до 31. Курсор будет иметь максимальный размер при четвертом параметре,

равном 0, и пятом параметре, равном 31. При увеличении четвертого параметра и уменьшении 5-го размер курсора будет уменьшаться.

Полный формат оператора координатного вывода:

LOCATE строка, столбец, режим, размер1, размер2,

где: **строка** — номер строки; **столбец** — номер столбца; **режим** — режим отображения курсора: 0 — видимый, 1 — невидимый; **размер1** и **размер2** — параметры размера курсора.

§ 9. Служебные слова и операнды

Как уже говорилось, операторы языка QBASIC представляют собой предложения языка программирования. Предложения обычного «человеческого» языка делятся на составные части: подлежащее, сказуемое, определение, дополнение и т. п. Подлежащее и сказуемое являются главными членами предложения, остальные — второстепенными. Предложения языка программирования также можно разбить на составные части, выделить главные и второстепенные члены предложения. Главными членами предложений языка программирования (операторов) являются служебные слова и операнды.

Оператор дает предписание машине выполнить определенное действие над какими-то объектами. Служебные слова описывают это действие. Операнды указывают, над какими именно объектами производится действие. Рассмотрим структуру известных нам операторов.

Оператор ввода данных с клавиатуры начинается со служебного слова **INPUT**. Это слово означает команду «Ввести с клавиатуры значения переменных». Далее следуют имена переменных, значения которых будут вводиться. Имена переменных и являются операндами.

Например, оператор

INPUT A, B, C

дает команду: «Ввести с клавиатуры значения переменных A, B, C». Здесь **INPUT** — служебное слово, A, B, C — операнды.

Оператор вывода на экран начинается со служебного слова **PRINT**. Это слово означает команду «Вывести на экран». Далее в операторе пишется список вывода — перечень объектов, которые необходимо вывести на экран. В списке вывода могут фигурировать имена переменных, надписи, числа, арифметические выражения. Операндами являются имена переменных и числа, входящие в арифметические выражения, а также надписи, заключенные в кавычки.

В стандарте Бейсика любой оператор должен начинаться со служебного слова. Поэтому оператор присваивания начинается со служебного слова **LET**, которое дает машине команду «присвоить». Далее следуют операнды, разделенные знаком равенства. Слева от знака равенства стоит один операнд — имя переменной, которой присваивается значение. Справа — один или несколько операндов, соединенных знаками арифметических операций. Операнды, соединенные знаками арифметических операций, образуют арифметическое выражение.

В простейшем случае справа от знака равенства стоит один операнд:

```
LET A = 3.
```

Этот оператор присваивает переменной с именем A значение 3. Если в качестве операнда фигурирует просто число, то это число называется константой. Арифметическое выражение может содержать как константы, так и имена переменных (идентификаторы). Оператор

```
LET B = 2 * A
```

присваивает переменной B значение, равное удвоенному значению переменной A. Здесь операндами являются константа 2 и имена переменных A, B.

В современных версиях Бейсика допускается написание оператора присваивания без служебного слова LET. Поэтому вышеприведенные операторы можно записывать так:

```
A = 3  
B = 2 * A.
```

Оператор присваивания — единственный оператор, который можно писать без служебного слова. Все остальные операторы начинаются обязательно со служебного слова. Зато операнды не всегда пишутся в операторах. Мы уже встречали два оператора без операндов: PRINT и CLS.

Кроме служебных слов и операндов в операторах пишутся «второстепенные члены предложения»: знаки математических операций, имена стандартных функций, значки «запятая», «точка с запятой». Рассмотрим роль некоторых «второстепенных членов предложения».

Запятая, поставленная в операторе INPUT между идентификаторами, просто разделяет их и не несет никакой смысловой нагрузки. Если же поставить запятую после комментария, то это приведет к некоторым последствиям: при работе оператора после комментария не появится «?». Этот значок появится только в том случае, если после комментария стоит «;».

В операторе PRINT запятая и точка с запятой указывают, сколько позиций будет отведено под выводимые переменные. Если выводимые переменные разделены запятыми, то под каждое значение будет отведено 14 позиций. Если после идентификатора поставить точку с запятой, то следующее значение будет выведено сразу же после предыдущего.

Имена стандартных функций (например, SQR или ABS), как и имена переменных, называются идентификаторами. Стандартная функция — это заранее заготовленная программа, проводящая вычисления по некоторой формуле. Эта программа хранится в памяти машины и вызывается по мере надобности. Имя стандартной функции как раз и служит для вызова этой программы.

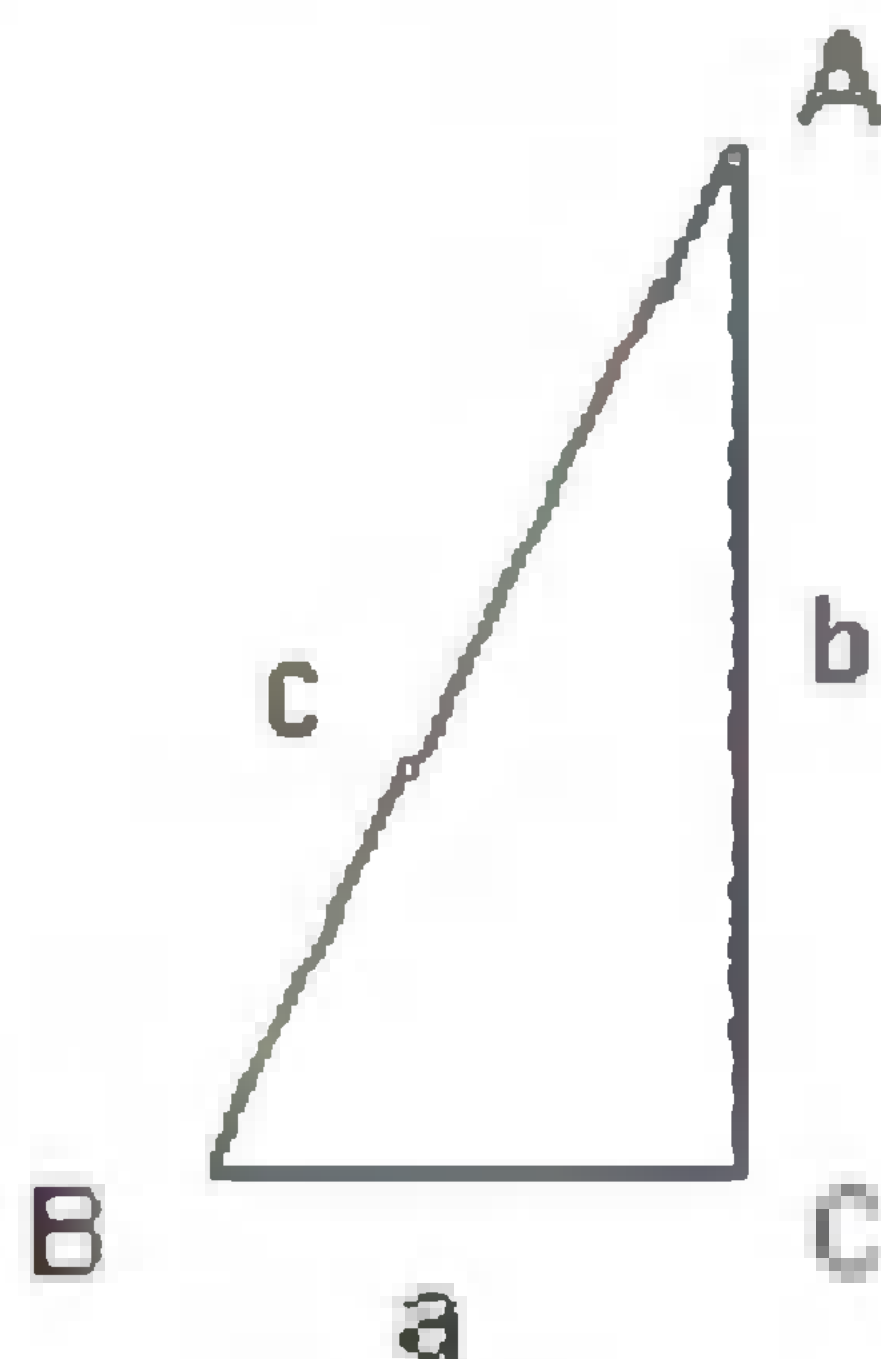
Имена стандартных функций и служебные слова — это словарный запас языка программирования. Значения этих слов хранятся в памяти машины. Такие слова в программировании называются зарезервированными словами.

Определение. Последовательности символов, значения которых хранятся в памяти машины, называются зарезервированными словами.

Встретив в программе зарезервированное слово, машина «вспоминает» его значение и выполняет действие, соответствующее этому слову. Поэтому зарезервированные слова нельзя использовать в качестве имен переменных.

§ 10. Некоторые сведения из геометрии

Рассмотрим прямоугольный треугольник ABC.



Будем в дальнейшем обозначать углы треугольника прописными буквами, а стороны — строчными.

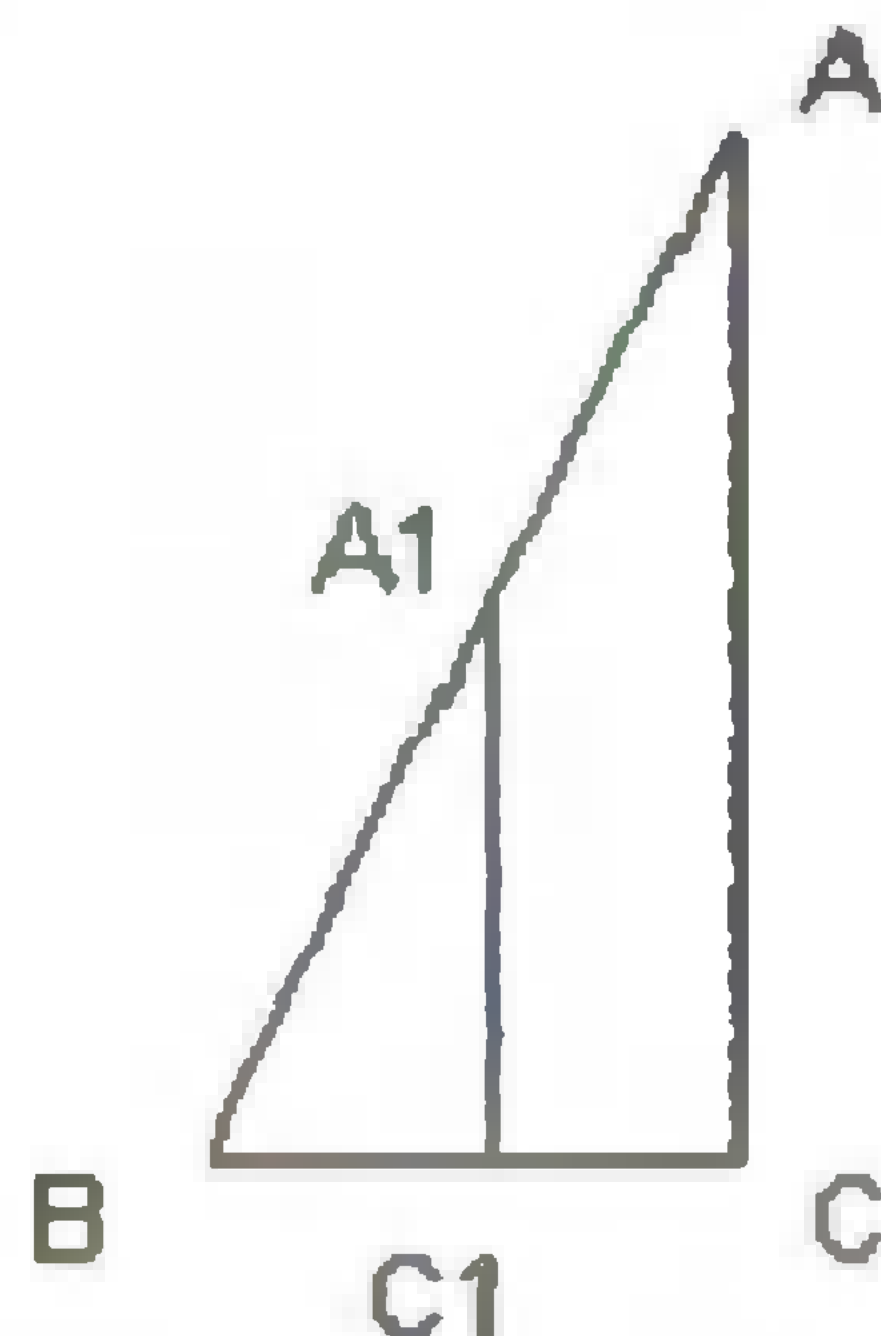
Сторона, лежащая против прямого угла, называется гипотенузой. Две другие стороны называются катетами. Гипотенузу принято обозначать буквой c , катеты — буквами a и b . Длины сторон прямоугольного треугольника связаны отношением:

$$c^2 = a^2 + b^2.$$

Эта формула носит название теоремы Пифагора.

Теореме Пифагора удовлетворяет треугольник со сторонами $a=3$, $b=4$, $c=5$. То, что такой треугольник является прямоугольным, было известно уже в Древнем Египте (хотя египтяне и не знали теоремы Пифагора). Поэтому треугольник со сторонами 3, 4, 5 называется египетским треугольником.

Проведем в треугольнике ABC отрезок A_1C_1 , параллельный стороне AC. Получим треугольник A_1BC_1 , подобный треугольнику ABC.

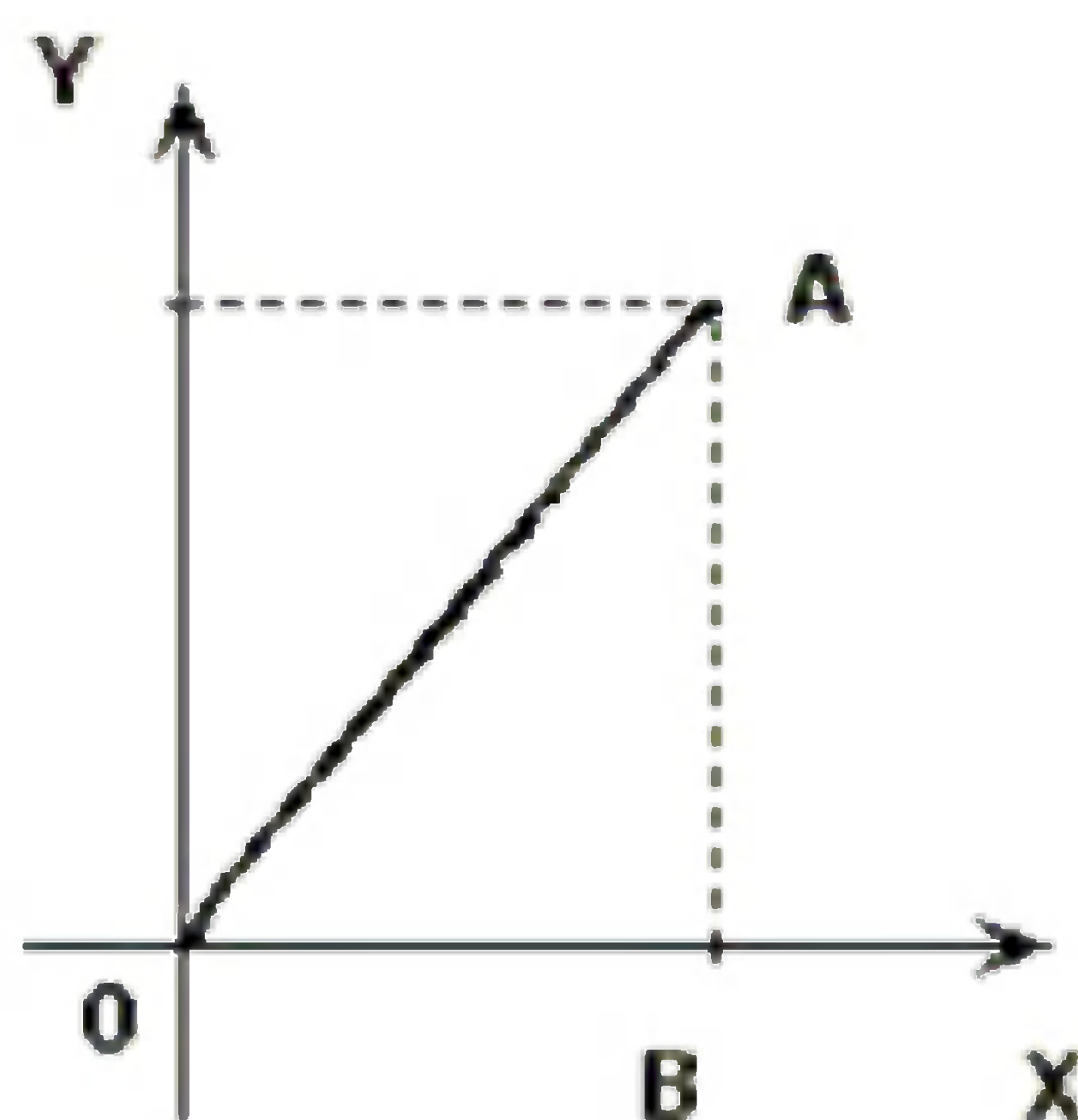


Для таких треугольников справедливы пропорции:

$$\frac{AB}{A_1B} = \frac{AC}{A_1C_1}$$

$$\frac{AB}{A_1B} = \frac{CB}{C_1B}$$

Рассмотрим теперь точку на координатной плоскости. Обозначим координаты точки буквами X и Y . Пусть точка $O(0,0)$ — начало координат. Пусть отрезок AB перпендикулярен оси OX . Вычислим расстояние от точки A до начала координат.

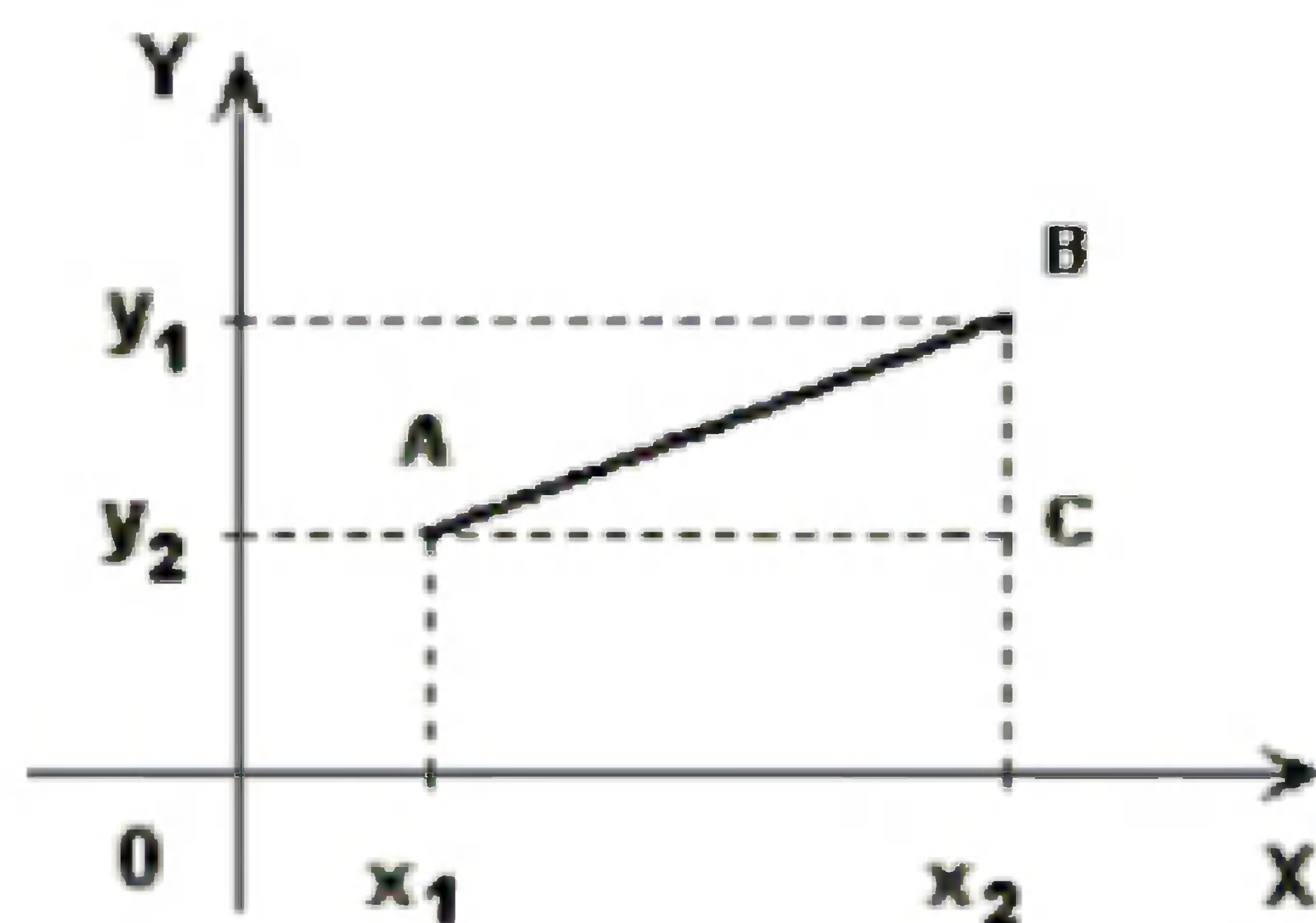


Треугольник OAB — прямоугольный. Гипотенуза AO — это искомое расстояние R . Очевидно, катет $AB = Y$, катет $OB = X$. Тогда:

$$R^2 = OB^2 + AB^2 = X^2 + Y^2$$

$$R = \sqrt{X^2 + Y^2}$$

Вычислим теперь расстояние между двумя точками $A(X_1, Y_1)$ и $B(X_2, Y_2)$.



Треугольник ABC — прямоугольный. Гипотенуза AB — искомое расстояние R .

Катеты равны: $AC = X_2 - X_1$, $BC = Y_2 - Y_1$. Отсюда:

$$R = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}.$$

§ 11. Тестирование программ

Итак, мы написали программу решения задачи, и в результате работы программы на экране появилось какое-то число. Это число является ответом на поставленную задачу. Правильен ли этот ответ?

Очень часто такой вопрос вызывает удивление: «Конечно ответ правилен! Ведь машина считала!!!»

На самом деле машина ничего не считает. Машина только выполняет набор команд, которые записаны в программе в виде операторов. Считает человек на машине. Правильность ответа зависит только от человека, но никак не от машины.

Если машина выполнила программу и на экране появилось число, то это значит, что в программе нет синтаксических ошибок. Но это еще не значит, что программа решает поставленную задачу. Ситуация здесь такая же, как при письменном ответе на экзамене: можно написать ответ без грамматических ошибок и в то же время не ответить на заданный вопрос.

Для проверки правильности работы программы проводят тестирование — проводят расчеты с такими исходными данными, для которых известен ответ. Такие исходные данные называются тестом.

Определение. Исходные данные, для которых известен конечный результат, называются тестом.

Если программа просто проводит вычисления по какой-либо формуле, то тестирование провести несложно. Достаточно «вручную» просчитать эту формулу с любыми исходными данными, а затем эти же данные ввести в машину. Если результаты расчетов совпадут, значит, программа составлена правильно.

Но не всегда решение задачи можно представить в виде формулы. Чаще всего ЭВМ применяется для решения таких задач, решение которых не выражается в виде формулы, а получается в результате выполнения некоторого алгоритма. Конечно, в принципе можно выполнить алгоритм «вручную», без ЭВМ. Но компьютер применяется для решения таких задач, где требуется проводить много вычислений или обрабатывать огромные объемы информации. Алгоритмы решения таких задач «вручную» выполнить просто невозможно. Поэтому поиск подходящего теста для проверки правильности работы программы — непростая задача.

Рассмотрим несколько примеров.

Задача 1. Треугольник задан координатами своих вершин. Вычислить периметр и площадь треугольника.

Алгоритм решения задачи следующий.

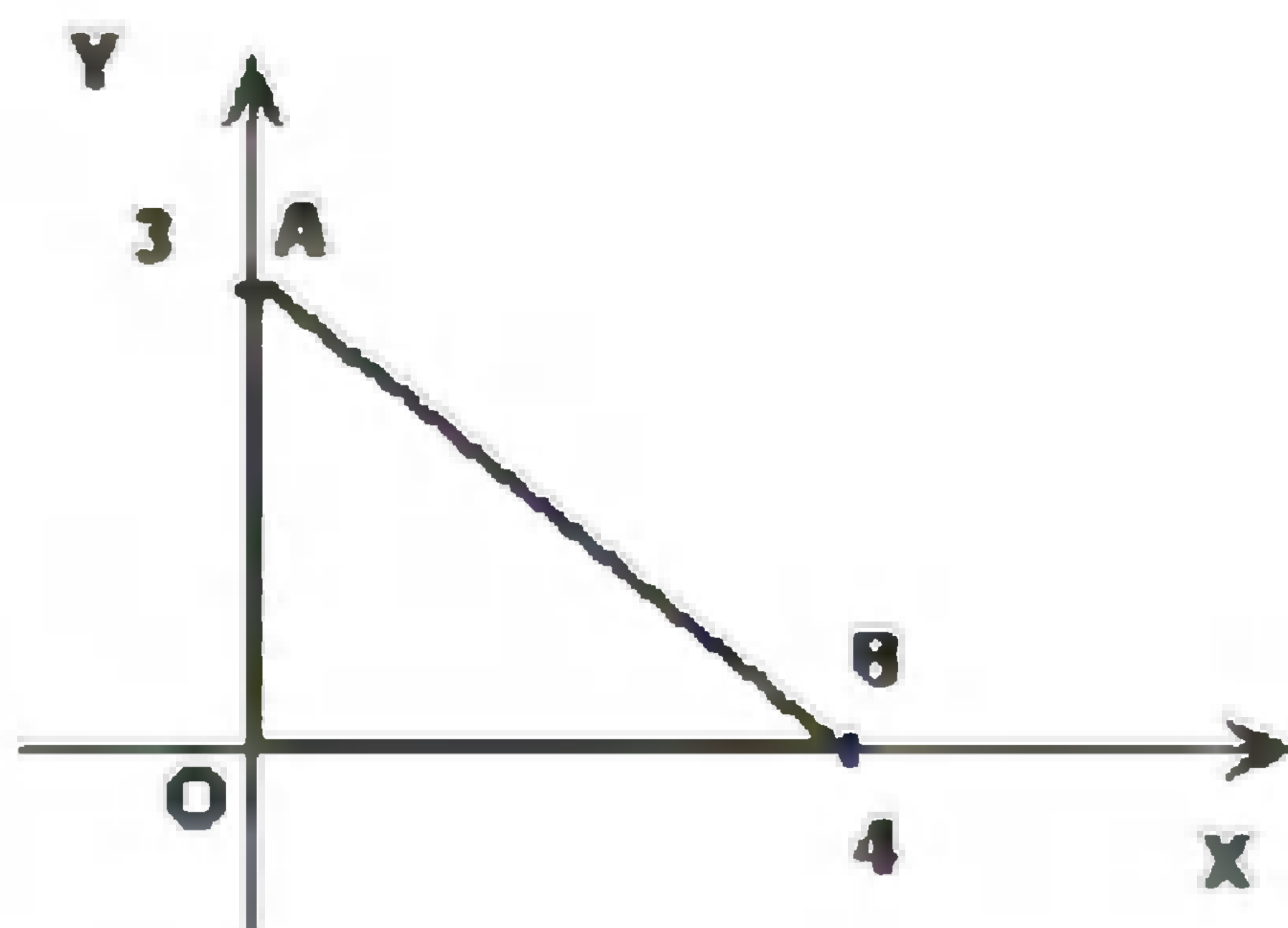
1. Вычислить расстояния между всеми парами точек. Эти расстояния являются длинами сторон треугольника.
2. Вычислить сумму сторон. Эта сумма является периметром.
3. Вывести периметр на экран.
4. Вычислить полупериметр (разделить периметр пополам).

5. По формуле Герона вычислить площадь треугольника.

6. Вывести площадь треугольника на экран.

В результате работы программы на экране появятся два числа — периметр и площадь. Чтобы быть уверенным в том, что программа составлена правильно, нужно использовать в качестве исходных данных координаты вершин такого треугольника, у которого легко можно посчитать периметр и площадь. Для этих целей можно использовать египетский треугольник. Периметр этого треугольника равен 12, площадь — 6.

Треугольник нужно разместить на координатной плоскости таким образом, чтобы можно было легко определить координаты вершин. Для этого вершину прямого угла помещаем в начало координат, а две другие вершины располагаем на осях:

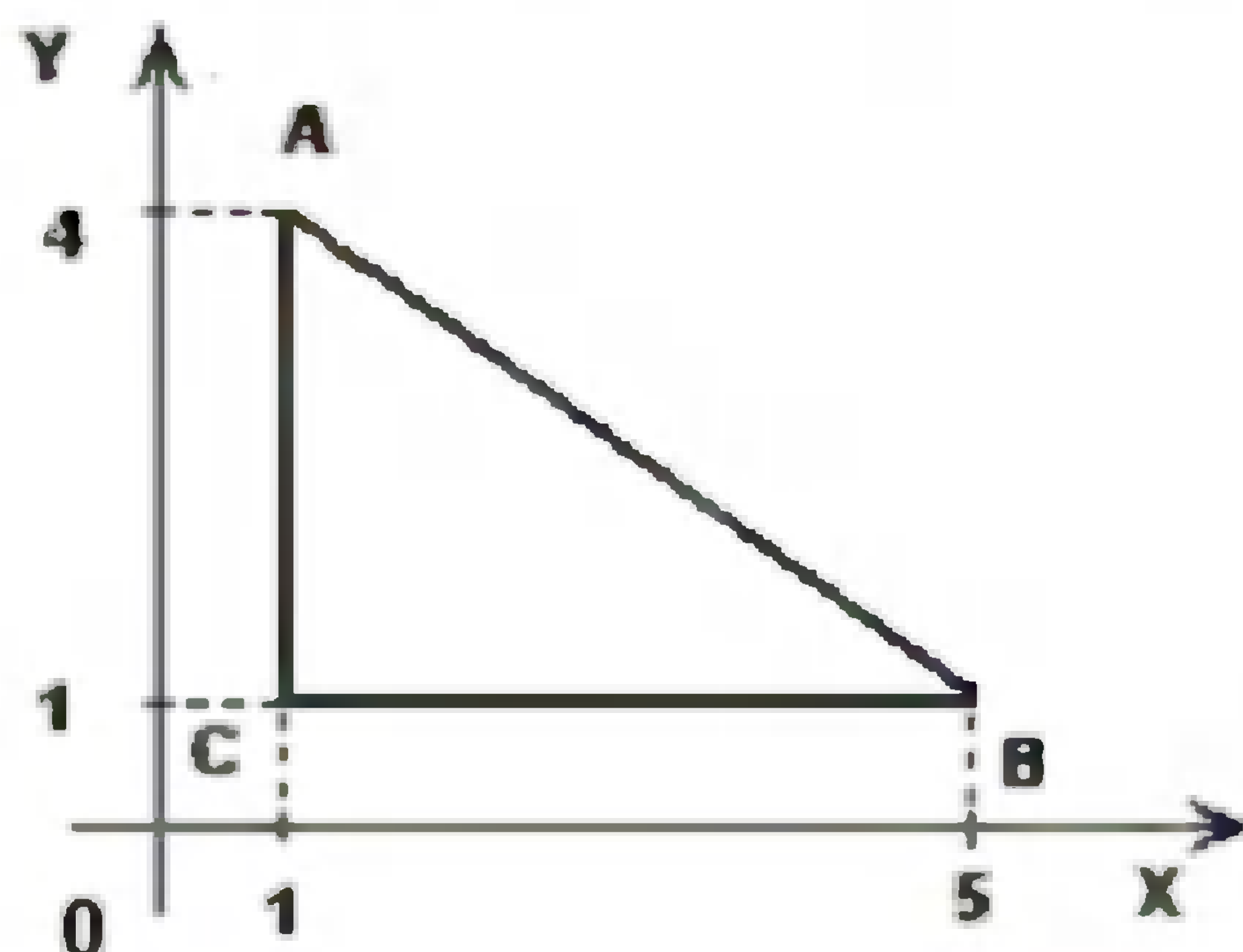


Тогда вершины треугольника будут иметь координаты (0,0), (4,0), (0,3). Введя эти числа в качестве исходных данных, мы должны получить на экране ответ: площадь — 6, периметр — 12.

Этот тест не очень удачный. Если, вычисляя длину катета по формуле

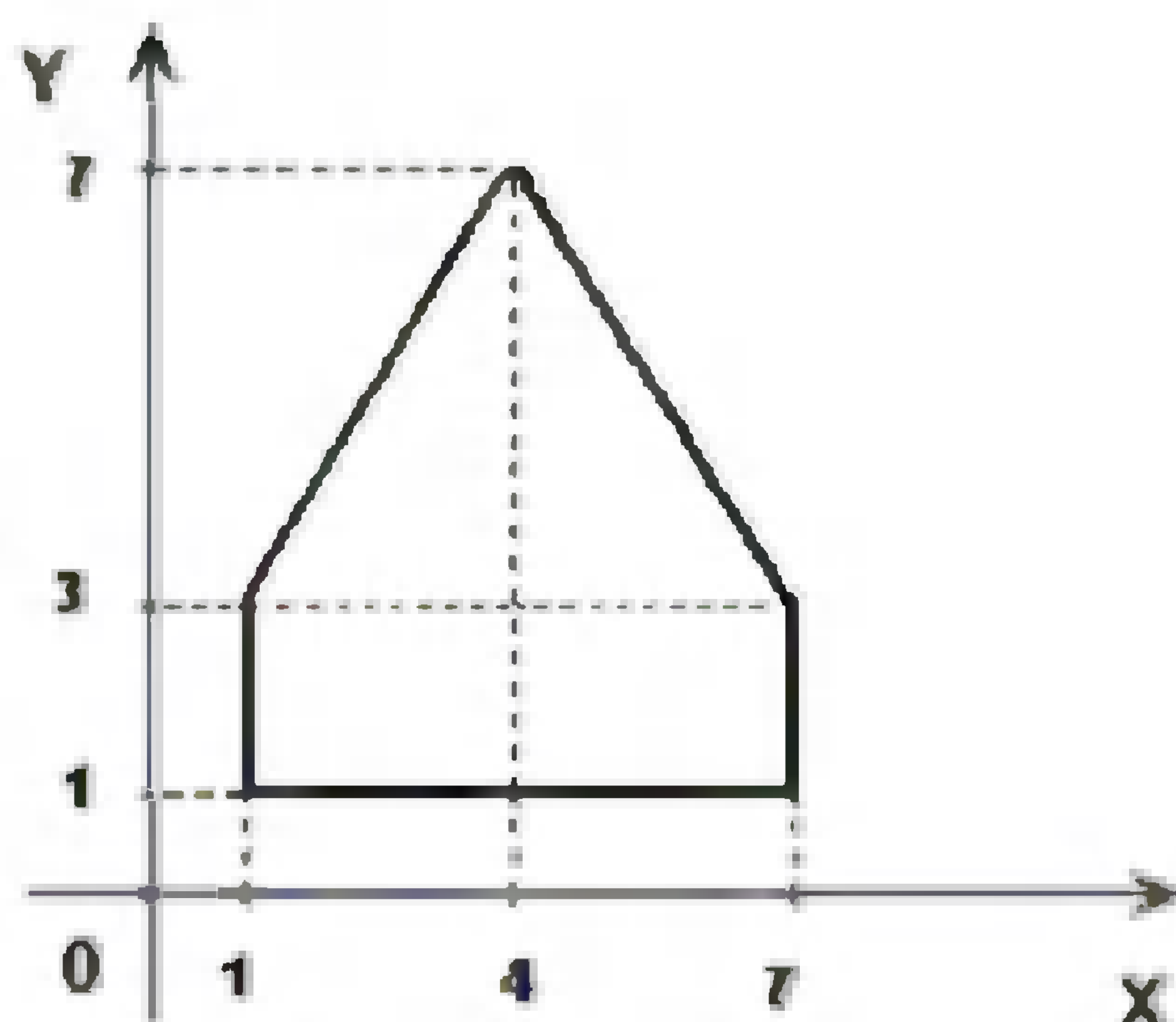
$$R = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2},$$

мы в скобках вместо знака «—» поставим «+», то при $X_1 = 0$, $Y_1 = 0$ эта ошибка никак не скажется на результате. Ошибка будет «затушевана» неудачно выбранным тестом. Более удачный тест — это сдвинутый треугольник с координатами вершин (1,1), (5,1), (1,4):



Задача 2. Многоугольник задан координатами своих вершин. Вычислить периметр и площадь многоугольника.

И здесь для построения теста можно использовать египетский треугольник. Только построения будут более сложными. В качестве теста можно взять пятиугольник в виде домика, стена которого представляет собой прямоугольник со сторонами 2,6,2,6, а крыша составлена из двух египетских треугольников. Для такого пятиугольника легко определить координаты вершин и посчитать периметр и площадь.



§ 12. Блок-схема алгоритма

До сих пор мы записывали алгоритмы двумя способами: на естественном языке и на языке программирования (QBASIC). Сейчас мы познакомимся еще с одним способом записи алгоритмов — графическим.

Сущность графического метода заключается в том, что алгоритм записывается не в виде последовательности команд, а изображается при помощи геометрических фигур: овалов, прямоугольников, ромбов, параллелепипедов. Такое графическое изображение алгоритма называют блок-схемой, или структурной схемой алгоритма.

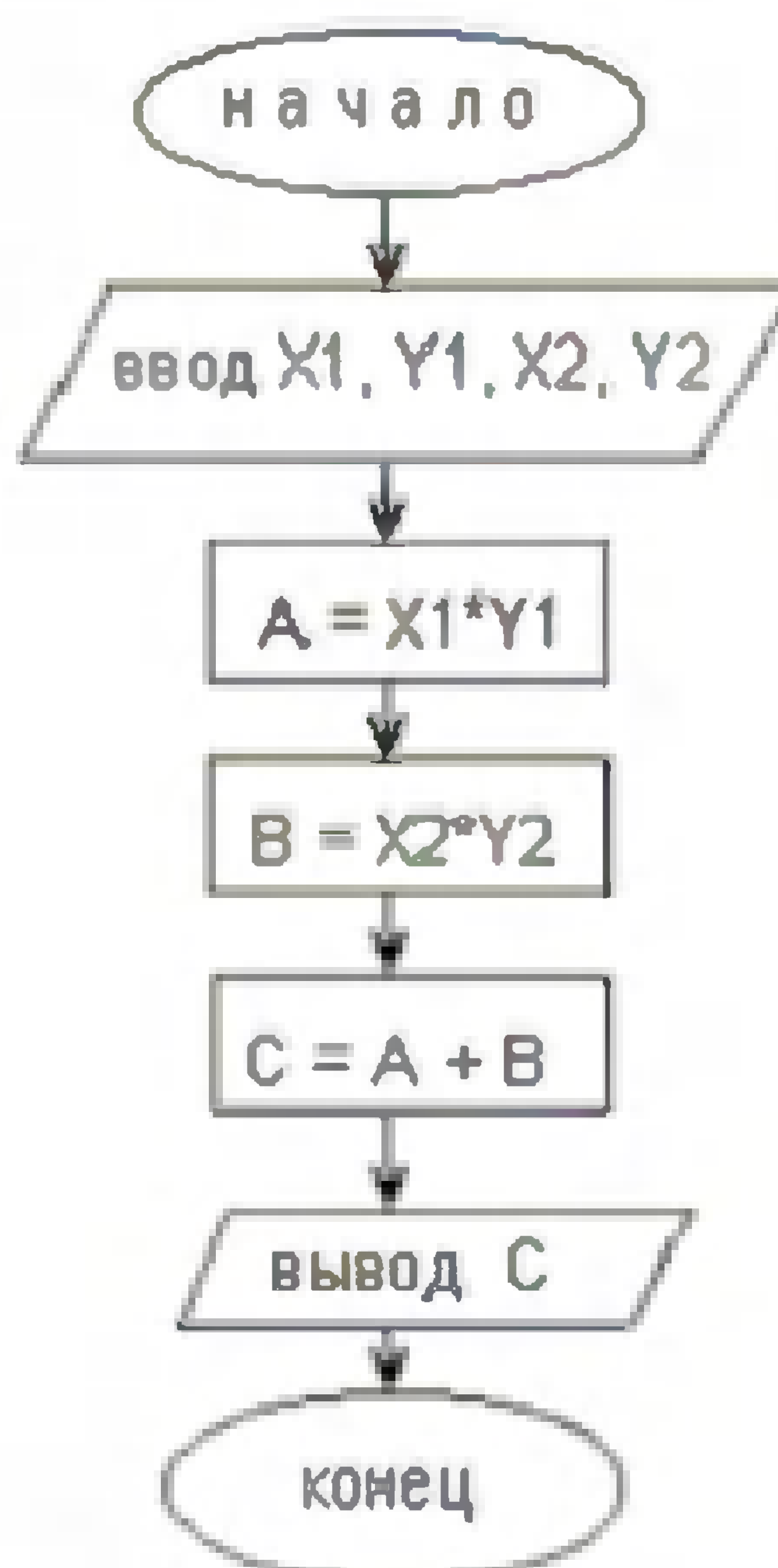
На графической схеме алгоритм ограничивается двумя овалами, обозначающими начало и конец алгоритма. Внутри овалов пишут слова «начало» и «конец»:



Вводимые и выводимые величины заключаются в параллелограммы. Команда ввода величины A будет выглядеть следующим образом:



Арифметические операции заключаются в прямоугольники. Блок-схему нашей задачи о помидорах можно изобразить так:



Конечно, нет смысла изображать графически линейный алгоритм. Но скоро мы познакомимся с такими алгоритмическими конструкциями, как ветвление и цикл. Алгоритмы с ветвлениями и циклами лучше воспринимаются, когда они представлены в виде графической схемы. Графическая схема отражает логическую структуру алгоритма, поэтому ее называют структурной схемой.

Графическая схема алгоритма рисуется не для исполнителя, а для человека (программиста). Графическая схема имеет целью объяснить, как работает алгоритм, а не дать руководство к исполнению. Поэтому в прямоугольниках можно писать не только отдельные команды, но и более сложные действия, состоящие из нескольких команд. Такие сложные действия называют блоками команд, отсюда другое название графической схемы — блок-схема.

§ 13. Правила составления алгоритма

Алгоритм составляется для некоторого исполнителя. Если исполнителем является человек, то составить алгоритм несложно: человек человека всегда поймет. А вот если исполнителем является какое-либо устройство (автомат или ЭВМ), то дело усложняется: машина ничего не понимает.

Она лишь выполняет определенные операции. Алгоритм строится из команд на выполнение этих операций. Чтобы машина выполнила нужные нам операции и мы получили результат, необходимо записывать команды, строго соблюдая правила записи алгоритмов, не записывая ничего лишнего и не упуская ничего существенного.

Первое правило заключается в том, что перед составлением алгоритма из условия задачи нужно выделить те величины, которые будут являться входными данными и те величины, которые будут являться выходными данными. Для этого условие задачи формулируется в виде «Дано... Требуется».

Не всегда в условии задачи явно говорится, что даны числа. Например, задача о вычислении объема куба может звучать так: «Дан куб с ребром a . Вычислить объем куба». Здесь написано «Дан куб», но куб — это не величина, а геометрическая фигура. Поэтому в разделе «Дано» следует написать: «Дано число a , которое является ребром куба». Входные данные алгоритма будут состоять из одного числа a .

Второе правило: алгоритм строится из отдельных законченных действий, которые называются шагами алгоритма. Это правило называется дискретностью алгоритма.

Шаги, которые совершает машина при выполнении программы, задаются операторами. Каждый оператор программы — это шаг алгоритма. После каждого шага должен быть указан шаг, который будет выполняться следующим. Это правило построения алгоритма называется детерминированностью (определенностью) алгоритма.

Те программы, с которыми мы имели дело до сих пор, выполнялись последовательно, в порядке записи операторов. Такие алгоритмы называются линейными. В линейном алгоритме каждый шаг обязательно выполняется, причем только один раз. Линейные алгоритмы на практике используются редко. Большинство алгоритмов включает в себя различного рода переходы. Бывают «переходы вперед», когда некоторые шаги пропускаются; бывают «переходы назад», когда некоторые шаги выполняются повторно — в любом случае после каждого шага исполнитель должен получать указание, какой шаг выполняется следующим.

Алгоритм не может работать бесконечно. Любой алгоритм должен когда-нибудь закончить работу и выдать результат. Это правило называется результативностью алгоритма. Расплывчатое выражение «когда-нибудь» часто заменяется более конкретным выражением: «за конечное число шагов», и правило результативности алгоритма формулируется так: алгоритм должен закончить работу за конечное число шагов.

При составлении алгоритма всегда нужно указывать, какой шаг выполняется первым, а какой — последним. Шаг, который выполняется первым, называется точкой входа в алгоритм; шаг, который выполняется последним, называется точкой выхода. Таким образом, алгоритм имеет одну точку входа и одну точку выхода.

В языках программирования обычно предусматриваются специальные операторы, которые обозначают начало программы и конец. В Бейсике нет оператора, обозначающего начало программы. В качестве такого оператора можно использовать оператор CLS.

Программа на Бейсике завершает свою работу после выполнения последнего оператора. Тем не менее в Бейсике есть оператор END, который обозначает конец программы. Этот оператор обычно ставят, если необходимо завершить программу раньше времени (т. е. перейти на последний оператор из середины программы).

В заключение перечислим все правила построения алгоритма:

1. Прежде чем составлять алгоритм (писать программу), необходимо выделить из условия задачи входные и выходные данные.
2. Алгоритм строится из отдельных (дискретных) шагов.
3. После каждого шага указывается, какой шаг будет выполняться следующим.
4. Алгоритм заканчивает работу за конечное число шагов.
5. Алгоритм имеет одну точку входа и одну точку выхода.

Глава III

Ветвления и циклы

1. Условный оператор

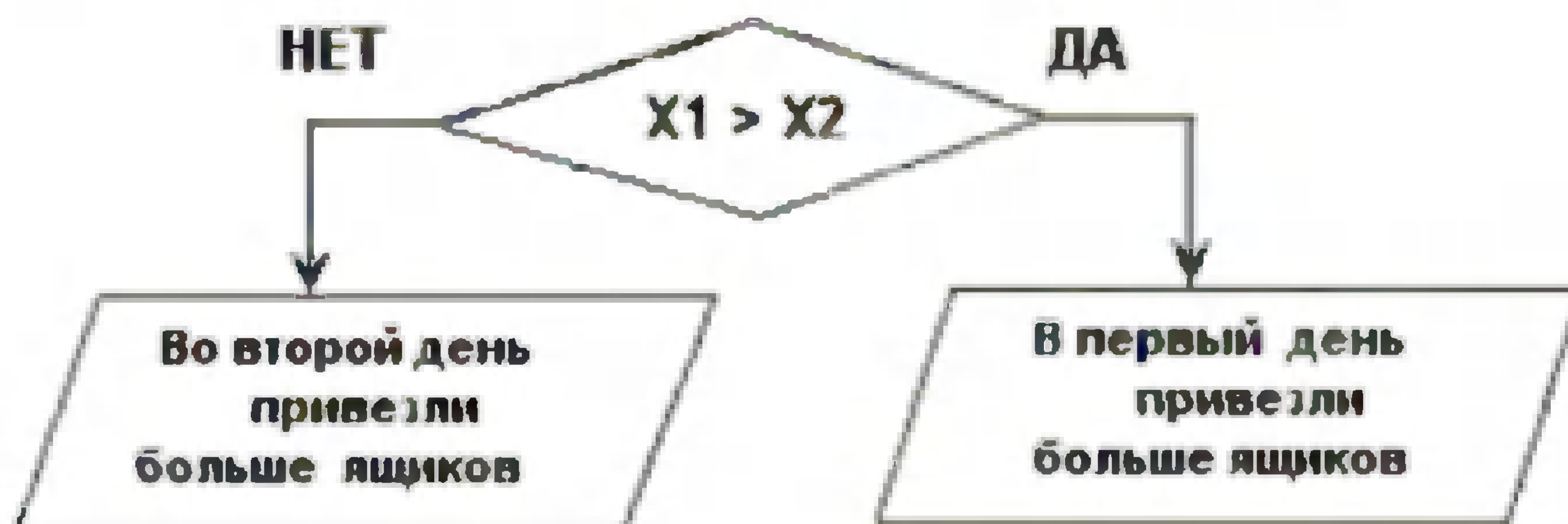
Изменим раздел «Требуется» в задаче 1, которую мы решали в главе I. Напишем так: определить, в какой день привезли больше ящиков — в первый или во второй. Конечно, если мы возьмем условие задачи в том виде, в каком оно приведено в § 3, то ответ будет один: в первый день привезли больше ящиков. Однако программа, приведенная в § 6, не содержит конкретных чисел. В программе стоят имена переменных: A1, B1, A2, B2, X1, X2. Мы заранее не знаем, какие числа будут стоять в условии задачи и какой получится результат. Поэтому машине нужно дать команду: «Проверить, в какой день привезли больше ящиков — в 1-й или во 2-й».

Такая команда в языке QBASIC дается при помощи условного оператора:

```
IF X1 > X2 THEN PRINT «В первый день привезли больше  
ящиков» ELSE PRINT «Во второй день привезли больше ящиков»
```

Перевод этого оператора на русский язык выглядит так: «Проверить условие $X1 > X2$. Если это условие выполняется, вывести на экран надпись «В первый день привезли больше ящиков», в противном случае вывести на экран надпись «Во второй день привезли больше ящиков».

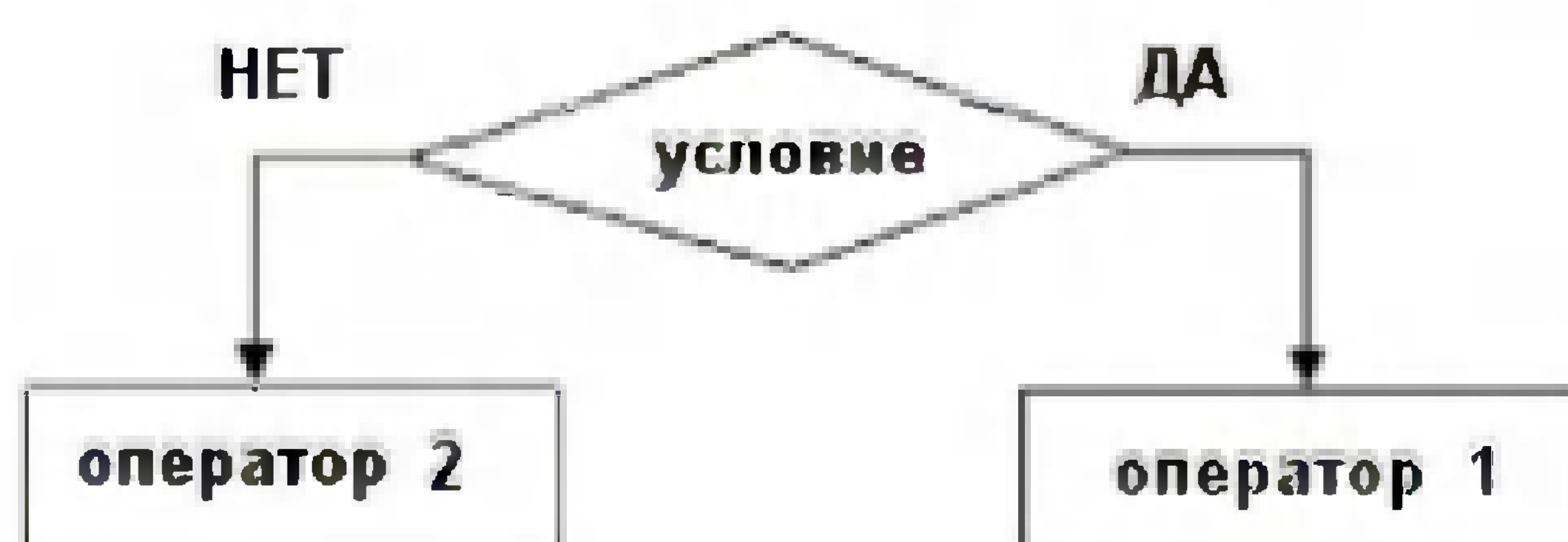
Здесь в зависимости от результатов проверки условия $X1 > X2$ выполняется один из двух операторов вывода: или тот, который стоит после слова THEN (PRINT «В первый день привезли больше ящиков»), или тот, который стоит после слова ELSE (PRINT «Во второй день привезли больше ящиков»). На графической схеме этот выбор изображается следующим образом:



Условие помещается в ромбике. Из ромбика выходят две ветви. Одна ветвь (ДА) соответствует случаю, когда условие выполняется; эта ветвь приводит к оператору, стоящему после слова THEN в условном операторе.

Вторая ветвь (НЕТ) соответствует случаю, когда условие не выполняется. Эта ветвь приводит к оператору, стоящему после слова ELSE.

В программировании эта конструкция называется ветвлением. Общий вид ветвления:



В QBASIC ветвление осуществляется при помощи условного оператора:

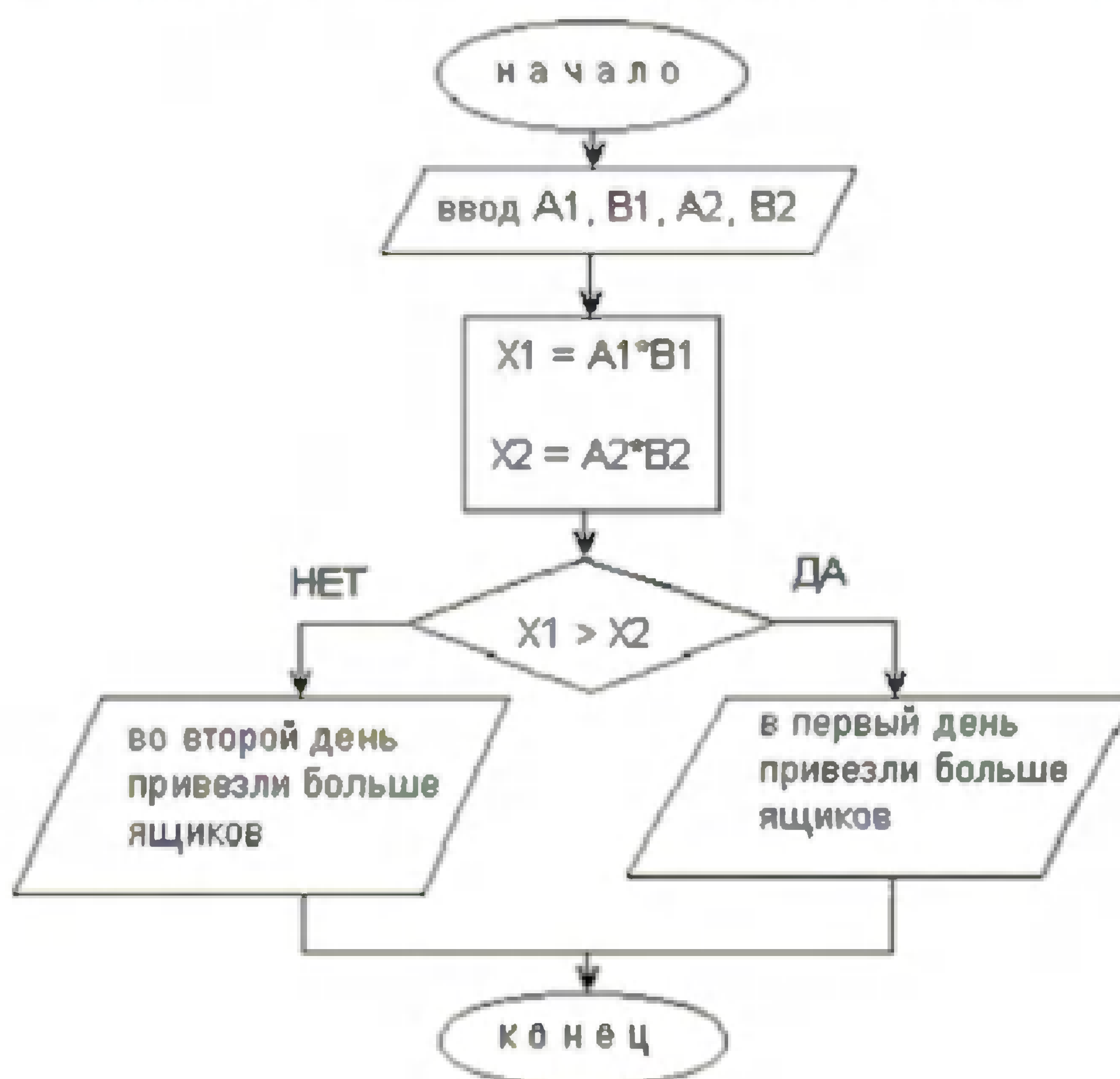
IF условие THEN оператор 1 ELSE оператор 2.

Условный оператор применяется, когда в зависимости от результатов проверки некоторого условия необходимо выполнить один из двух операторов. Конечно, в качестве «оператор 1» и «оператор 2», можно использовать любые операторы, а не только операторы вывода.

При написании условий в условных операторах используются математические операции, которые называются операциями отношений:

- > — больше;
- < — меньше;
- >= — больше или равно (не меньше);
- <= — меньше или равно (не больше);
- не равно;
- = — сравнение (проверка на равенство).

В нашей задаче «условие» — это математическое неравенство $A > B$. В заключение нарисуем полностью блок-схему алгоритма решения задачи:



Соответствующая программа:

```
CLS
INPUT A1, B1, A2, B2
X1 = A1*B1
X2 = A2*B2
IF X1 > X2 THEN PRINT «В первый день привезли больше
ящиков»
ELSE PRINT «Во второй день привезли больше ящиков».
```

§ 2. Оператор безусловного перехода

Рассмотрим следующую задачу. Пусть дано число. Определить, попадает ли оно в отрезок $[-1, 1]$. Число x попадает в этот отрезок, если выполняется двойное неравенство $-1 < x < 1$. Проверять двойные неравенства мы пока не умеем, поэтому можно проверить два других неравенства: $x < -1$ и $x > 1$. Если хотя бы одно из них выполняется, то число не попадает в заданный отрезок. Если ни одно из этих условий не выполняется, то число попадает в заданный отрезок. Получается следующая последовательность действий:

1. Проверить условие $x < -1$. Если оно выполняется, вывести на экран надпись «число не попадает в заданный отрезок» и закончить работу.
2. Проверить условие $x > 1$. Если оно выполняется, вывести на экран надпись «число не попадает в заданный отрезок» и закончить работу.
3. Вывести на экран надпись «число попадает в заданный отрезок» и закончить работу.

Очевидно руководствуясь этими указаниями, мы решим задачу, поскольку до третьего пункта мы дойдем только в том случае, если не будут выполняться ни первое, ни второе условия. Однако эти указания еще не являются алгоритмом, поскольку алгоритм имеет только одну точку выхода, т. е. команда «закончить работу» должна встречаться только один раз. У нас эта команда стоит во всех трех пунктах.

Чтобы превратить полученную последовательность действий в алгоритм, нужно поставить команду «закончить работу» последним пунктом:

1. Проверить условие $x < -1$. Если оно выполняется, вывести на экран надпись «число не попадает в заданный отрезок» и перейти на пункт 4.
2. Проверить условие $x > 1$. Если оно выполняется, вывести на экран надпись «число не попадает в заданный отрезок» и перейти на пункт 4.
3. Вывести на экран надпись «число попадает в заданный отрезок».
4. Закончить работу.

Команде «закончить работу» соответствует оператор END, который уже упоминался в главе II. А вот команда «перейти на...» — новая для нас команда. Эта команда называется командой безусловного перехода. Ей соответствует оператор безусловного перехода, который выглядит следующим образом:

GOTO метка.

Этот оператор дает машине указание «перейти на строчку, помеченную данной меткой». В качестве метки в QBASIC можно использовать целое число или идентификатор. Напишем программу, соответствующую нашему алгоритму, используя целое число в качестве метки:

```
CLS
INPUT «Введите число»; X
IF X < -1 THEN PRINT «Число не попадает в заданный
отрезок»: GOTO 10
IF X > 1 THEN PRINT «Число не попадает в заданный
отрезок»: GOTO 10
PRINT «Число попадает в заданный отрезок»
10  END
```

Если в качестве метки используется идентификатор, то метка отделяется от оператора двоеточием. Перепишем эту же программу, используя в качестве метки идентификатор LAB (от англ. LABEL — метка):

```
CLS
INPUT «Введите число»; X
IF X < -1 THEN PRINT «Число не попадает в заданный
отрезок»: GOTO LAB
IF X > 1 THEN PRINT «Число не попадает в заданный
отрезок»: GOTO LAB
PRINT «Число попадает в заданный отрезок»
LAB: END
```

При использовании меток в программе операторы обычно сдвигают вправо, чтобы метки выделялись в тексте программы. Тогда программу легче читать.

В других версиях Бейсика, в которых строки обязательно нумеруются, идентификатор в качестве метки использовать нельзя. В этих версиях в качестве меток используются номера строк. Наша программа на GW-BASIC будет выглядеть следующим образом:

```
10  CLS
20  INPUT «Введите число»; X
30  IF X < -1 THEN PRINT «Число не попадает в заданный
отрезок»: GOTO 60
40  IF X > 1 THEN PRINT «Число не попадает в заданный
отрезок»: GOTO 60
50  PRINT «Число попадает в заданный отрезок»
60  END
```

Оператор безусловного перехода часто применяют для контроля правильности вводимых данных. Например, если мы хотим извлекать квадратный корень из вводимого числа, то хорошо бы перед вычислением квадратного корня проверить, не является ли данное число отрицательным.

Если мы по ошибке введем отрицательное число, то при вычислении квадратного корня произойдет сбой. Чтобы этого не случилось, делаем контроль вводимых данных:

```
CLS
10 INPUT «Введите число»; X
  IF X < 0 THEN PRINT «Ошибка ввода»: GOTO 10
  Y = SQR(X)
  PRINT «Корень квадратный из числа»; X; «Равен»; Y
```

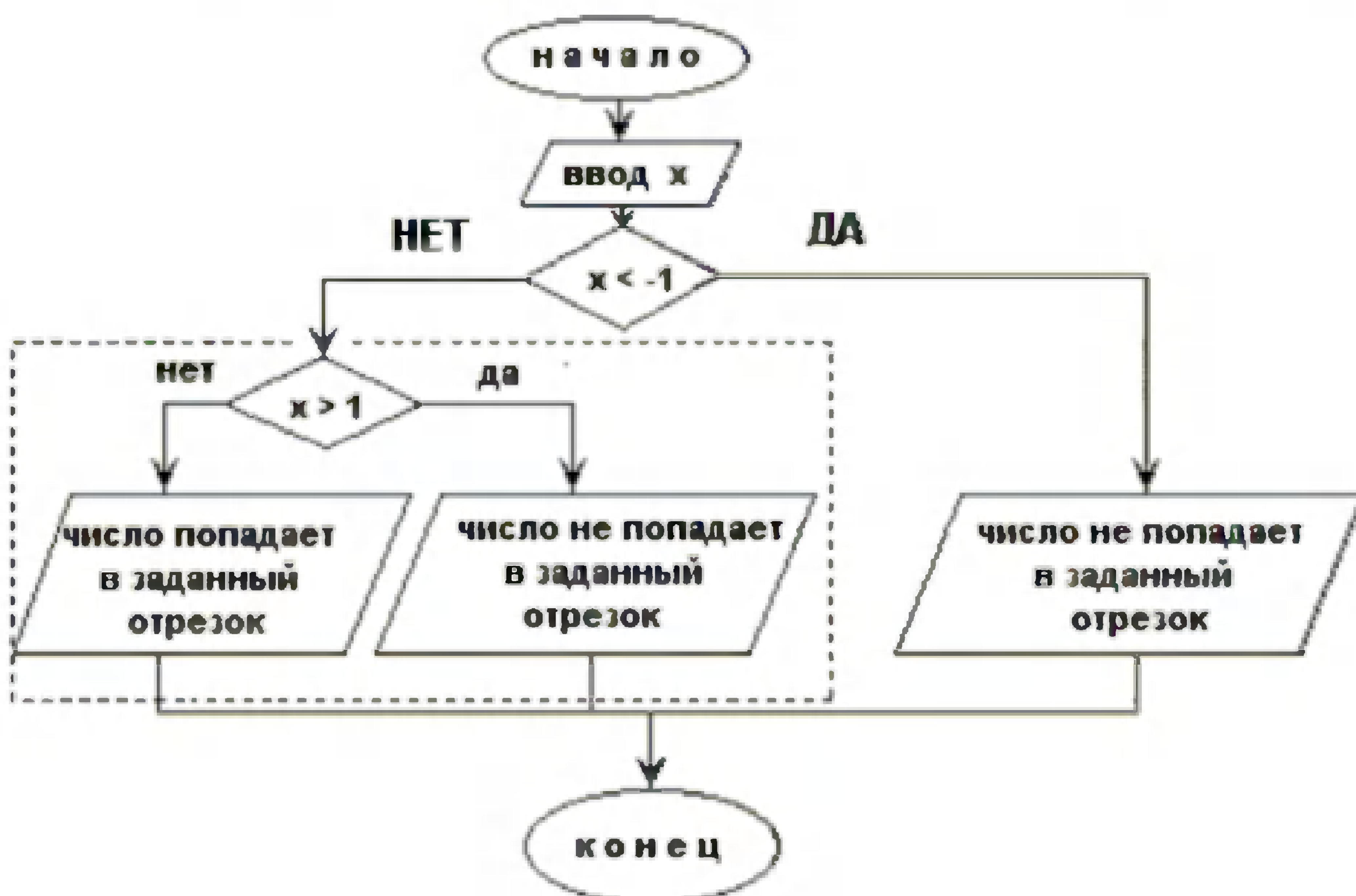
Теперь при вводе отрицательного числа программа вернется на строку 10 и еще раз запросит данные.

Иногда для контроля правильности вводимых данных одно и то же число запрашивается два раза и введенные величины сравниваются. Если они не совпадают, данные запрашиваются еще раз:

```
CLS
10 INPUT «Введите число»; X
  INPUT «Введите число еще раз»; Y
  IF X < > Y THEN PRINT «Ошибка ввода»: GOTO 10
```

§ 3. Вложенные условные конструкции

Рассмотрим графическую схему алгоритма решения задачи из предыдущего параграфа:



Сравним ее со схемой ветвления, приведенной в § 1:



Здесь прямоугольнику «оператор 1» соответствует вывод надписи «точка не попадает в заданный отрезок», а прямоугольнику «оператор 2» соответствует новая условная конструкция — проверка условия $X > 1$ (на рис. эта конструкция обведена пунктиром). Получается, что одна условная конструкция «вложена» в другую.

Было бы логично писать новую условную конструкцию там, где ей положено быть — после слова ELSE в условном операторе, проверяющем условие $X < -1$. Однако мы обрываем этот оператор и используем оператор GOTO, чтобы перейти в конец программы. Новая условная конструкция пишется как отдельный оператор, не связанный с предыдущим.

При такой записи программы ее структура не соответствует логической организации алгоритма. Когда программа короткая, это нестрашно. Но если программа громоздкая, то приходится применять много операторов перехода, что затрудняет чтение программы. Нужно стремиться к тому, чтобы форма записи программы соответствовала логической структуре алгоритма.

В языке QBASIC существует такая форма записи условного оператора, которая позволяет отразить логическую структуру алгоритма. Выглядит эта запись так:

```
IF условие THEN
    блок операторов
ELSE
    блок операторов
END IF.
```

Отметим разницу между двумя формами записи оператора. Прежде мы записывали весь оператор в одной строке. Теперь после слова THEN следует начинать новую строку. После слова ELSE также следует начинать новую строку. Признаком конца условного оператора служат слова END IF.

После служебных слов THEN и ELSE в новом операторе можно написать много операторов — столько, сколько нужно. Конечно, и в старом операторе после слова ELSE можно написать несколько операторов, разделенных двоеточием. Но в Бейсике длина строки ограничена 256 символами. Кроме того, длинные строки в программе трудно читать.

Запишем нашу программу в новой форме:

```
CLS
INPUT «Введите число»; X
```

```
IF X < -1 THEN  
    PRINT «Точка не принадлежит заданному отрезку»  
ELSE  
    IF X > 1 THEN PRINT «не принадлежит» ELSE PRINT  
    «принадлежит»  
END IF.
```

§ 4. Логические операции

Рассмотрим точку на координатной плоскости. Как известно, положение точки на плоскости задается двумя координатами (x , y). При каких значениях координат точка будет лежать в верхней полуплоскости (т. е. выше оси OX)? Очевидно, координаты точек, лежащих в верхней полуплоскости, удовлетворяют условию $y > 0$. Аналогично, если точка лежит в правой полуплоскости (правее оси OY), то ее координаты удовлетворяют условию $x > 0$.

Теперь сформулируем условие, которому удовлетворяют координаты точек, лежащих в 1-й координатной четверти (рис. 1). Эти точки одновременно находятся выше оси OX и правее оси OY , т. е. одновременно выполняются два условия: $y > 0$ и $x > 0$. Выражение « $x > 0$ и $y > 0$ » является составным условием, сложенным из двух простых: « $x > 0$ » и « $y > 0$ ». Союз «и» здесь играет роль связки, соединяющей два простых условия в одно, более сложное. Такое «сложное» условие, составленное из нескольких более простых, будем называть составным условием.

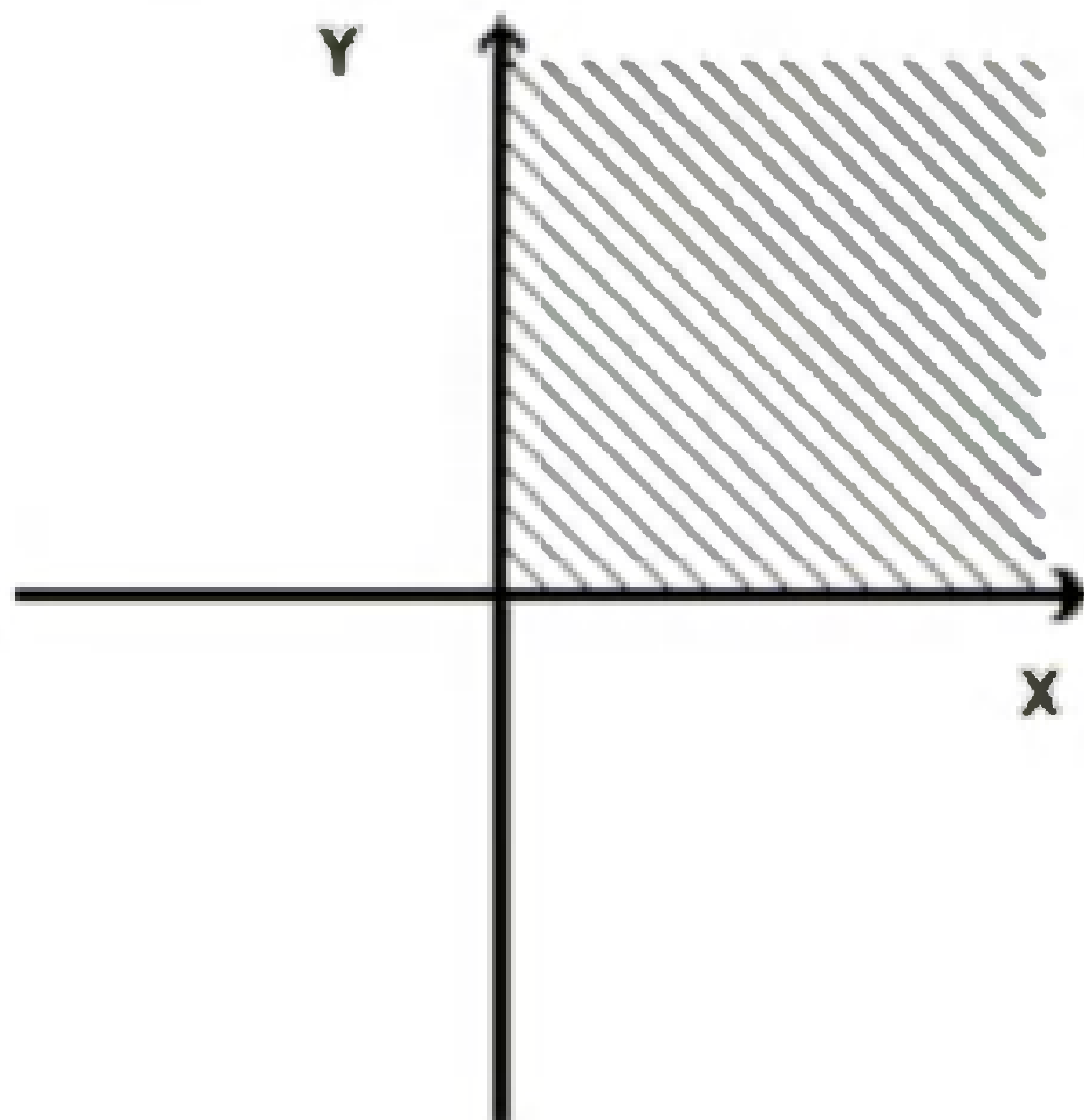


Рис. 1

Процесс объединения нескольких условий в одно составное называется логической операцией. На языке QBASIC логическая операция, соответствующая союзу «и», обозначается английским словом AND. Называется эта операция логическим умножением.

Теперь можно написать программу, решающую следующую задачу: определить, лежит ли точка с координатами (x, y) в 1-й четверти:

```
CLS
INPUT «Введите координаты точки»; x, y
IF X > 0 AND Y > 0 THEN PRINT «Точка лежит в 1-й
четверти» ELSE PRINT «Точка не лежит в 1-й четверти»
```

Составное условие « $x > 0$ and $y > 0$ » выполняется только тогда, когда выполнены оба простых условия: « $x > 0$ » и « $y > 0$ ». Выражение « $x > 0$ and $y > 0$ » можно сформулировать следующим образом: условия « $x > 0$ » и « $y > 0$ » выполняются одновременно. Условный оператор в нашей программе можно перевести на русский язык так: «Если одновременно выполняются условия $x > 0$ и $y > 0$, то точка лежит в 1-й четверти, в противном случае точка не лежит в 1-й четверти».

Решим другую задачу: определить, лежит ли точка в области, заштрихованной на рис. 2? В эту область попадают точки, лежащие выше оси OX , а также точки, лежащие правее оси OY . Координаты этих точек удовлетворяют условию: $x > 0$ или $y > 0$. Можно сказать и так: точка лежит в заштрихованной области, если хотя бы одна из координат положительна, т. е. или первая координата положительна, вторая отрицательна, или вторая координата положительна, а первая отрицательна, или обе координаты положительны.

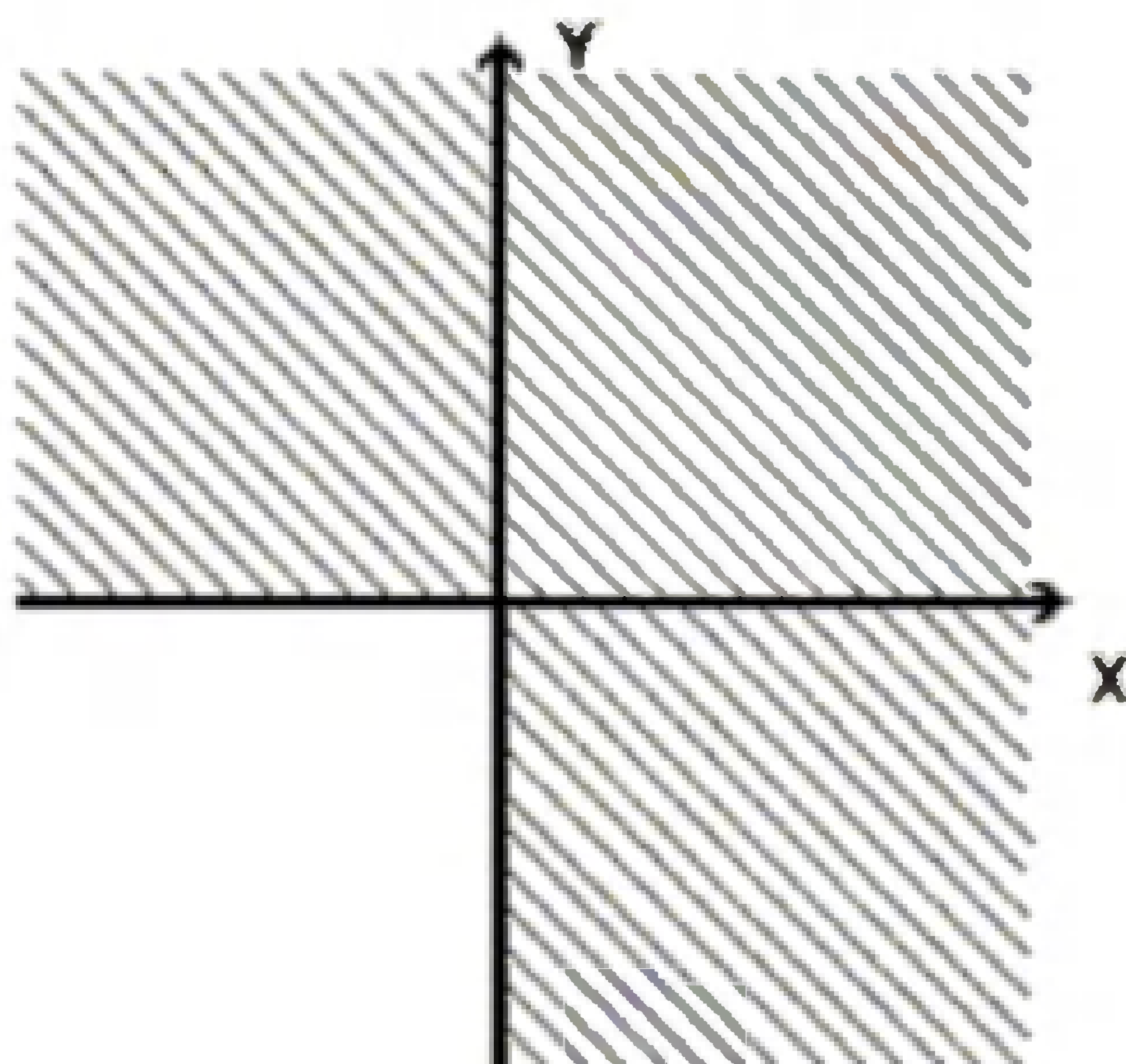


Рис. 2

У нас опять получилось составное условие, построенное из двух простых. Но простые условия соединены союзом «или». Этому союзу соответствует логическая операция, называемая «логическое сложение». На языке QBASIC эта операция обозначается словом OR.

Теперь можно записать программу, решающую нашу задачу:

```
CLS
INPUT «Введите координаты точки»; x, y
```

```
IF X > 0 OR Y > 0 THEN PRINT «Точка лежит в  
заштрихованной области» ELSE PRINT «Точка не лежит в  
заштрихованной области»
```

Новое составное условие « $x > 0$ or $y > 0$ » выполняется в том случае, если выполняется хотя бы одно из простых условий. Словосочетание «хотя бы» здесь означает, что выполняется или первое простое условие « $x > 0$ », или второе простое условие « $y > 0$ », или оба простых условия выполняются одновременно.

Подведем некоторые итоги. У нас появились новые термины: «простое условие», «составное условие». Дадим им определения.

Определение.

Условие, содержащее логические связки, называется составным условием.

Условие, не содержащее логических связок, называется простым, или элементарным условием.

Мы пока имеем дело только с двумя логическими связками — «И» и «ИЛИ» и соответственно с двумя логическими операциями — логическим умножением и логическим сложением. В дальнейшем мы познакомимся с другими логическими связками и логическими операциями. Отметим пока, что логическое умножение называют также операцией «И» и конъюнкцией, а логическое сложение — операцией «ИЛИ» и дизъюнкцией.

В одном составном условии можно объединять не только два, но и любое количество простых условий. Если простые условия объединены логическим умножением, то составное условие выполняется только в том случае, если выполняются все простые условия одновременно. Если же простые условия объединены операцией логического сложения, то для выполнения составного условия достаточно выполнения хотя бы одного простого условия.

§ 5. Циклы

Решим следующую задачу: вывести на экран квадраты натуральных чисел от 1 до 5. Решить задачу можно при помощи следующей программы:

```
CLS  
A = 1  
B = A^2  
PRINT A;B  
A = 2  
B = A^2  
PRINT A;B  
A = 3  
B = A^2  
PRINT A;B  
A = 4  
B = A^2  
PRINT A;B
```

```
A = 5  
B = A^2  
PRINT A;B
```

Мы взяли всего 5 чисел, а программа получилась уже довольно длинная. А если взять не 5 чисел, а 15? А если понадобится вычислить квадраты 150 чисел?

В программе повторяется один и тот же набор операций: переменной *A* присваивается некоторое значение, затем квадрат этого значения присваивается переменной *B* и значения обеих переменных выводятся на экран. Процесс вычисления квадратов пяти чисел можно сокращенно описать следующим образом:

1. Присваивать переменной *A* значения от 1 до 5.
2. Возвести каждое значение переменной *A* в квадрат и результат записать в переменную *B*.
3. Вывести на экран значения переменных *A* и *B*.

Вместо 15 строчек программы мы получили всего 3 строчки. Из контекста ясно, что эти строчки нужно выполнять пять раз. Такое повторение одних и тех же операций с разными значениями переменных в программировании называют циклом.

Определение. Многократное выполнение одних и тех же операций называется циклом.

Язык QBASIC имеет специальный оператор, позволяющий записать такой цикл (программисты говорят: организовать цикл). Называется этот оператор оператором цикла. Запишем программу, вычисляющую квадраты первых пяти чисел, при помощи оператора цикла:

```
CLS  
FOR A = 1 TO 5  
  B = A ^ 2  
  PRINT A; B  
NEXT A.
```

В переводе на русский язык это звучит примерно так:

1. Для каждого значения *A* от 1 до 5 выполнить.
2. Переменной *B* присвоить значение A^2 .
3. Вывести на экран значения *A* и *B*.
4. Перейти к следующему значению *A*.

Состоит оператор цикла из трех частей: заголовка, тела цикла и признака окончания цикла. Заголовком цикла является строчка, которая начинается служебным словом **FOR**. Признаком конца цикла является слово **NEXT**. Все, что стоит между заголовком и словом **NEXT**, называется телом цикла. Тело цикла — это и есть те самые операторы, которые многократно выполняются в программе. Конечно, при каждом выполнении тела цикла значения переменных *A* и *B* будут разными.

Переменная *A*, стоящая в заголовке цикла, называется параметром цикла. По этой причине сам этот цикл называется циклом с параметром. В программировании рассматриваются и другие виды циклов, с которыми мы познакомимся позднее. Начальное и конечное значения параметра

цикла определяют, сколько раз будет выполняться цикл. Каждое выполнение цикла называется итерацией.

Мы написали самый простой вариант оператора цикла, когда при каждой итерации параметр цикла увеличивается на 1. QBASIC позволяет изменять параметр цикла на любую величину. Величина, на которую изменяется параметр цикла при каждой итерации, называется шагом изменения параметра цикла. Шаг указывается в заголовке цикла после служебного слова STEP. Если мы хотим вычислить квадраты чисел от 1 до 5 с шагом 0.1, то нужно написать:

```
CLS
FOR A = 1 TO 5 STEP 0.1
  B = A ^ 2
  PRINT A; B
NEXT A
```

Здесь выражение STEP 0.1 указывает, что при каждой итерации параметр увеличивается на 0.1 (STEP — по-английски «шаг»). При этом изменится количество итераций (посчитайте, сколько итераций сделает этот цикл).

Если указать шаг отрицательный, то параметр цикла будет уменьшаться. При этом начальное значение параметра цикла должно быть больше конечного. Например, программа

```
CLS
FOR A = 5 TO 4.5 STEP -0.1
  B = A ^ 2
  PRINT A; B
NEXT A
```

выведет на экран следующие числа:

5	25
4.9	24.09
4.9	23.04
4.7	22.09
4.6	21.16
4.5	20.25

Теперь можно записать формат оператора цикла с параметром:

```
FOR парам = нз TO кз [STEP шаг]  
тело  
NEXT [парам].
```

Здесь **парам** — идентификатор параметра цикла; **нз** — начальное значение параметра цикла; **кз** — конечное значение параметра цикла; **шаг** — шаг изменения параметра цикла.

В современных версиях Бейсика имя параметра цикла после слова NEXT можно не писать (как и слово LET в операторе присваивания). Поэтому здесь имя параметра написано в квадратных скобках. Можно также не указывать шаг, если он равен 1 (в этом случае говорят, что шаг по умол-

чанию принимается равным 1). Поэтому слова «STEP шаг» в заголовке цикла написаны в квадратных скобках.

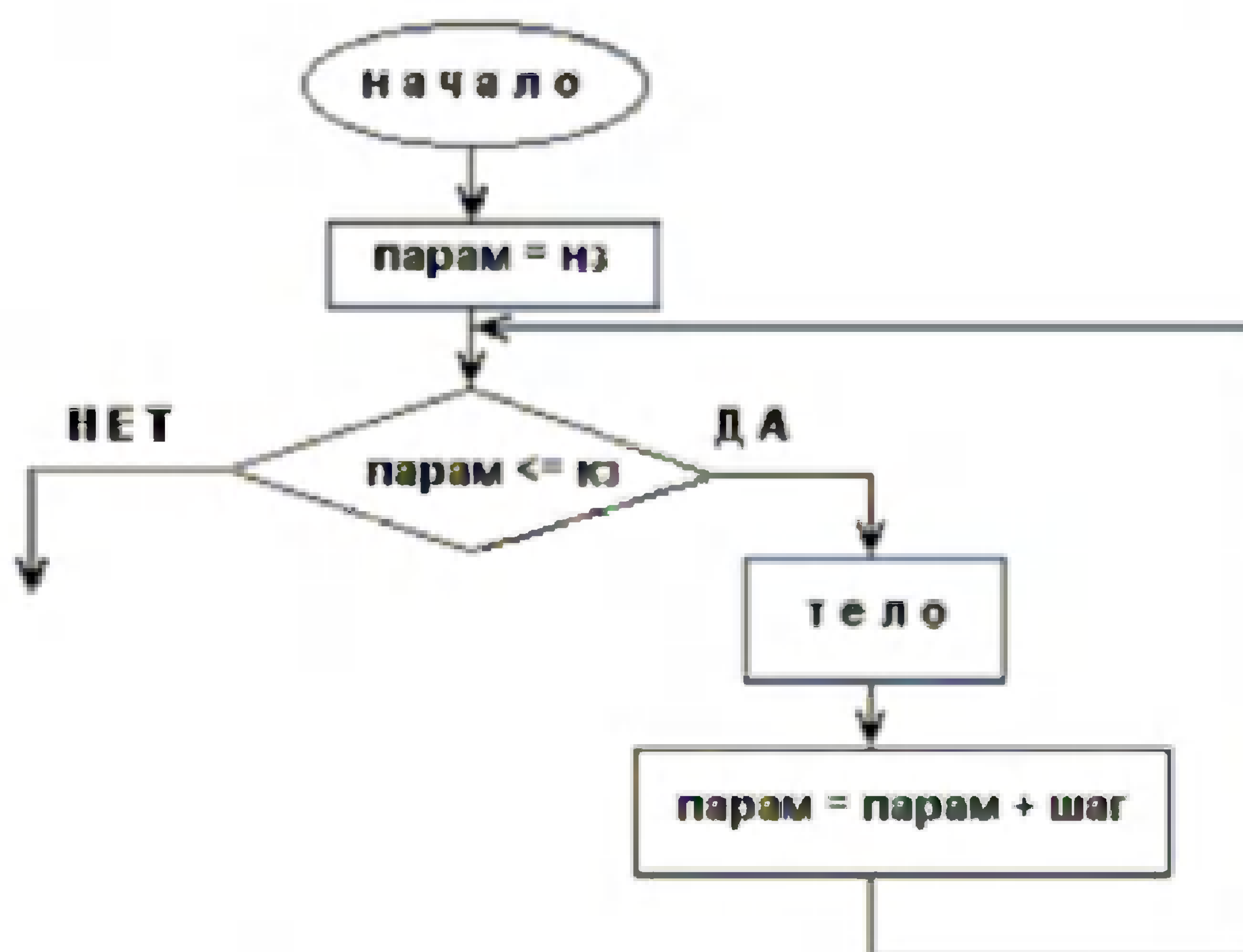
Начальное и конечное значения параметра цикла на обязательно должны быть числами. В качестве параметров «нз» и «кз» можно писать арифметические выражения. Например, можно написать следующий заголовок цикла:

```
FOR I = PI/2 TO 3*PI/2 STEP PI/10.
```

В заключение можно сказать, что оператор цикла с параметром сокращенно называют оператором FOR ... NEXT.

§ 6. Блок-схема оператора цикла

Работу оператора цикла можно представить в виде следующей схемы:



Когда заголовок цикла выполняется в первый раз, параметру цикла присваивается начальное значение. Потом проверяется условие $\text{парам} \leq \text{кз}$. Если это условие выполняется, то выполняются операторы, составляющие тело цикла. После выполнения тела цикла параметр цикла увеличивается на величину шага. Эта операция производится в строчке NEXT. Затем происходит возвращение на проверку условия $\text{парам} \leq \text{кз}$.

Первый прямоугольник и ромбик схемы содержатся в заголовке цикла. Увеличение параметра цикла происходит в строчке NEXT.

Цикл «крутится», пока выполняется условие $\text{парам} \leq \text{кз}$. Как только это условие перестает выполняться, цикл прекращает работу и выполняются операторы, стоящие после слова NEXT. Параметр цикла при этом имеет значение $\text{кз} + \text{шаг}$. Это можно проверить, поставив после слова NEXT оператор вывода:

```
CLS
FOR I = 1 TO 5
```

```
PRINT A; A^2
NEXT
PRINT I
```

Последний оператор выведет на экран значение параметра цикла, равное 6.

Если поставить в заголовке цикла $n_z > k_z$, то после проверки условия мы сразу пойдем по ветке «нет» и цикл не будет выполняться ни разу. Параметр при этом сохранит начальное значение.

Так работает оператор цикла, в котором указан положительный шаг. Если поставить шаг отрицательный, то изменится проверяемое условие, и блок-схема примет следующий вид:



У такого цикла начальное значение должно быть больше конечного.

§ 7. Массивы

Напишем программу, которая поможет учителю выявить уровень успеваемости в классе. Допустим, класс написал контрольную работу. По результатам этой работы всем ученикам выставлены оценки. Составим программу, которая посчитает, сколько получено пятерок, четверок, троек, двоек.

Здесь мы сразу столкнемся с такой проблемой: каким образом занести наши оценки в память компьютера, чтобы их было легко обрабатывать? Сразу договоримся, что фамилии учеников заносить в память не будем. Учеников мы пронумеруем (как в классном журнале) и будем различать их по номерам. Если ученик № 1 получил оценку 5, то эту оценку можно занести в память при помощи следующего оператора присваивания:

$A1 = 5.$

Если ученик № 2 получил оценку 3, то в программе нужно написать:

$A2 = 3$

и т. д. Если в классе 25 учеников, то придется написать 25 операторов присваивания.

Теперь посчитаем количество оценок каждого типа. Начнем с пятерок. Количество пятерок будем считать при помощи переменной P , которая будет играть роль счетчика. Сначала примем $P = 0$. Затем будем проверять каждую оценку (проверять значение каждой переменной A), и если это значение равно 5, то значение переменной P увеличим на 1. Проверка производится при помощи условного оператора:

```
P = 0  
IF A1 = 5 THEN P = P+1  
IF A2 = 5 THEN P = P+1
```

и т. д.

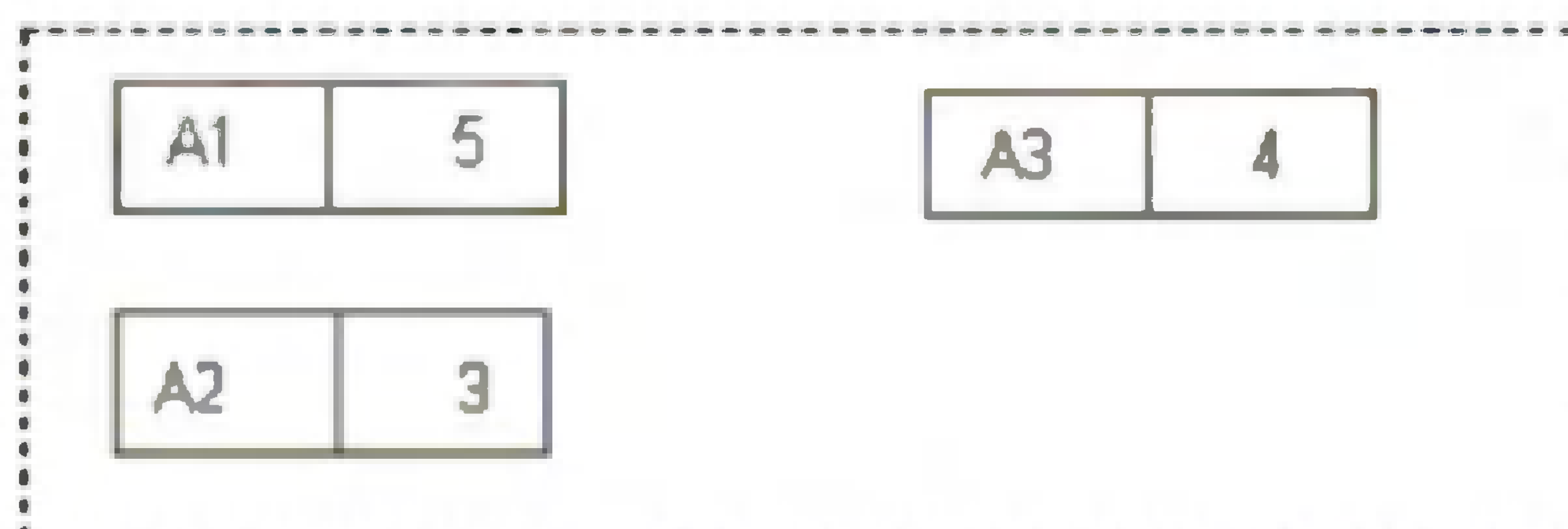
Для подсчета количества пятерок нужно написать 25 условных операторов. Для подсчета количества четверок нужно написать еще 25 операторов, и так для каждого типа оценок. Всего программа будет содержать 100 условных операторов. Сколько же операторов нужно написать, чтобы посчитать оценки по всей школе?

Очевидно, здесь нужно использовать оператор цикла, поскольку программа состоит из одинаковых операторов, проверяющих значения переменных $A_1, A_2, A_3 \dots$. Нужно только каким-то образом перебрать в цикле значения всех переменных. Делается это следующим образом: оценки нумеруются (так же, как и фамилии) и записываются в памяти под одним именем. Номер оценки указывается в скобках рядом с именем переменной: $A(1)$, $A(2)$, $A(3)$ и т. д. Если 1-й ученик получил пять, 2-й ученик три, а 3-й — четыре, то в программе можно написать: $A(1) = 5$, $A(2) = 3$, $A(3) = 4$.

Номер переменной в программировании называется индексом. Переменная, имеющая номер, называется «переменная с индексом»; переменная без номера называется «простая переменная». В памяти простые переменные и переменные с индексом записываются по-разному. Если в программе написаны операторы присваивания:

```
A1 = 5  
A2 = 3  
A3 = 4,
```

то в памяти компьютера будут образованы переменные с именами A_1, A_2, A_3 и в них будут записаны соответствующие значения. Переменные эти будут размещаться в разных участках памяти и никак не будут связаны друг с другом. Это можно схематически изобразить следующим образом:

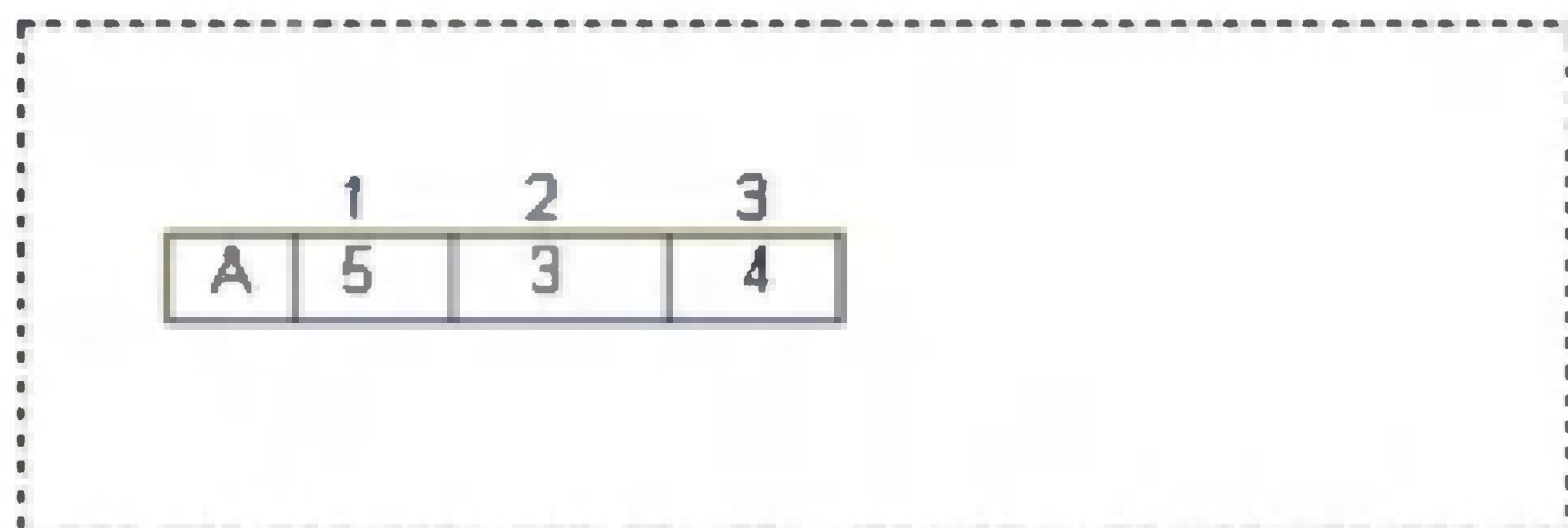


ПАМЯТЬ

Здесь пунктиром обозначена внутренняя память ЭВМ.
Другая ситуация возникнет, если мы напишем.

$A(1) = 5$
 $A(2) = 3$
 $A(3) = 4$.

В этом случае всем трем переменным будет присвоено одно и то же имя. Располагаться значения всех трех переменных будут в соседних ячейках памяти, последовательно друг за другом согласно номерам:



ПАМЯТЬ

Такой набор пронумерованных переменных, имеющих одно имя, в программировании называется массивом. Каждая переменная называется элементом массива. Номер элемента массива называется его индексом.

Определение. Группа пронумерованных элементов, имеющих одно имя, называется массивом.

В качестве индекса элемента массива можно использовать не только число, но и переменную. Если ученик номер N получил 5, то можно написать

$A(N) = 5$.

Естественно, до того как этот оператор будет выполняться, переменная N должна быть определена, т. е. ей нужно присвоить какое-либо значение. Тогда машина, выполняя этот оператор, сначала найдет в памяти переменную N , прочитает ее значение и затем запишет число 5 в переменную A с соответствующим номером. Если мы хотим занести число 5 в переменную $A(1)$, то можно написать

$N = 1$
 $A(N) = 5$.

В качестве индекса можно использовать любую переменную, которая принимает целочисленное значение. Если мы в качестве индекса используем параметр цикла, то сможем написать небольшую программу, подсчитывающую количество пятерок, полученных учениками:

```
P = 0
FOR I = 1 TO 25
  IF A(I) = 5 THEN P = P+1
NEXT.
```

Эта же программа посчитает количество пятерок, полученных любым количеством учеников (например, учениками всех восьмых классов или учениками всей школы). Надо только в качестве конечного значения параметра цикла поставить другое число.

В этот же цикл можно поставить еще три условных оператора, подсчитывающих количество четверок, троек и двоек. Сделайте это самостоятельно. Не забудьте завести для каждого типа оценок свой счетчик.

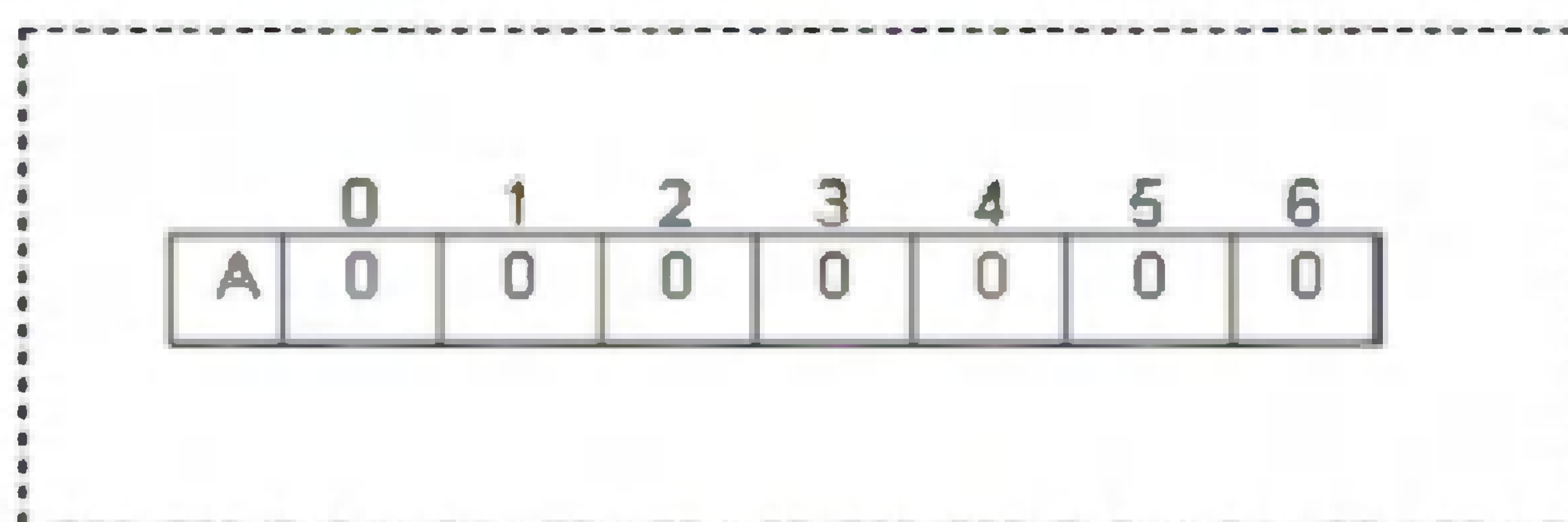
§ 8. Заполнение массива

Работа с массивом начинается с описания массива. Описать массив — это значит сообщить машине имя массива и количество элементов в массиве. Все это нужно для того, чтобы машина заранее выделила в памяти место под этот массив (зарезервировала участок памяти).

Оператор описания массива начинается с ключевого слова DIM (от английского слова *dimension* — размер). Затем идет имя массива, после которого в скобках ставится максимальное значение индекса. Например, если мы хотим описать массив A, в котором будет 6 элементов, то нужно написать

```
DIM A(6).
```

Выполняя этот оператор, машина выделит в памяти 7 клеточек, расположенных рядом, пронумерует их и присвоит им всем одно и то же имя. В Бейсике нумерация элементов массива начинается с нуля, поэтому клеточек получится на одну больше, чем написано в скобках. Все клеточки будут заполнены нулями. В памяти возникнет такая «картинка»:



ПАМЯТЬ

Теперь нам нужно заполнить массив, т. е. записать в клеточки числа, с которыми мы будем работать. Заполнить массив можно при помощи оператора присваивания, перечислив последовательно все элементы (как мы это делали в предыдущем параграфе). Но если элементов массива много, то этот способ не рационален, поскольку программа получается очень громоздкой.

Для сокращения текста программы используется оператор цикла. В операторе цикла можно поставить оператор INPUT:

```
DIM A (6)
FOR I = 1 TO 6
PRINT «Введите»; I; «-е число»
```

```
INPUT A (I)  
NEXT
```

Для заполнения массива можно использовать оператор чтения данных. Сформируем для примера массив оценок, которые получили 25 учеников:

```
DIM A (25)  
FOR I = 1 TO 25  
READ A (I)  
NEXT  
DATA 2,3,5,4,3,2,2,3,4,4,4,5,3,3  
DATA 4,4,5,5,5,3,2,4,3,3, 4
```

В строчке DATA перечисляются оценки в том порядке, в котором пронумерованы ученики.

Глава IV

Графика

§ 1. Графический режим QBASIC

До сих пор для вывода на экран информации мы использовали только оператор PRINT. Этот оператор работает подобно пишущей машинке — он позволяет выводить только отдельные значки (символы): буквы, цифры, скобки, знаки препинания и т. п. Такой режим работы называется символьным режимом.

Рисунки на экране компьютера выполняются при помощи особых операторов вывода, которые называются графическими операторами. Если оператор PRINT выводит на экран символьную информацию, то графические операторы выводят графическую информацию.

Графические операторы не работают в символьном режиме. Для вывода рисунков на экран необходимо установить графический режим. Этот режим в QBASIC устанавливается при помощи следующего оператора:

`SCREEN n,`

где **n** — номер режима.

Номер **n** может меняться от 0 до 13. При **n** = 0 устанавливается обычный символьный режим. Остальные режимы — графические.

Режим SCREEN 1 — черно-белый. Режим SCREEN 2 — цветной режим низкого разрешения. Эти режимы остались от прежних версий Бейсика, которые предназначались для работы с компьютерами, обладающими небольшой видеопамятью.

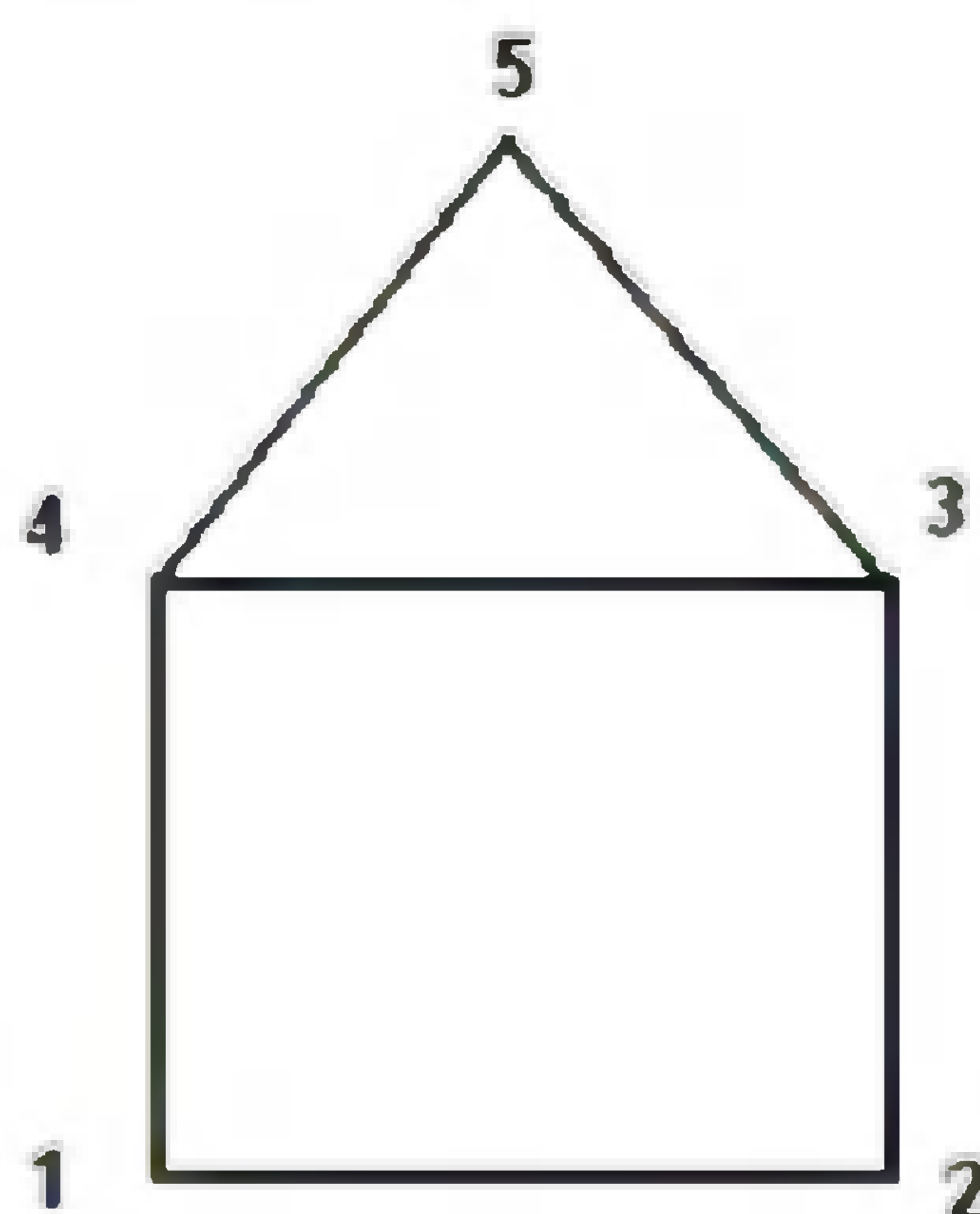
При работе с QBASIC используют режимы от 7 до 13. Режим SCREEN 7 имеет самое низкое разрешение (изображение на экране получается грубое). Для решения несложных графических задач удобнее всего пользоваться режимом SCREEN 12.

Графические операторы QBASIC позволяют использовать 16 цветов. Цвет задается номером, который называется параметром цвета. Как параметр цвета используется в графических операторах, мы увидим позднее, а пока перечислим цвета и их номера:

0 — черный	5 — малиновый	10 — светло-зеленый
1 — голубой	6 — коричневый	11 — светло-бирюзовый
2 — зеленый	7 — белый	12 — светло-красный
3 — бирюзовый	8 — серый	13 — светло-малиновый
4 — красный	9 — светло-голубой	14 — желтый
		15 — ярко-белый

§ 2. Установка системы координат

После установки режима экрана можно приступать к созданию рисунка. Давайте нарисуем на экране домик.



Наш домик состоит из отдельных отрезков. Чтобы этот домик появился на экране, нужно пронумеровать концы этих отрезков и дать машине команды:

1. Провести линию из точки 1 в точку 2.
2. Провести линию из точки 2 в точку 3.
3. Провести линию из точки 3 в точку 4.
4. Провести линию из точки 4 в точку 1.
5. Провести линию из точки 4 в точку 5.
6. Провести линию из точки 5 в точку 3.

Но как указать положения точек? Как вы знаете из курса математики, положение точки на плоскости задается при помощи двух чисел, которые называются координатами точки. Чтобы определить эти координаты, необходимо задать систему координат, т. е. указать, где находится начало координат, выбрать направления осей, выбрать единицу измерения.

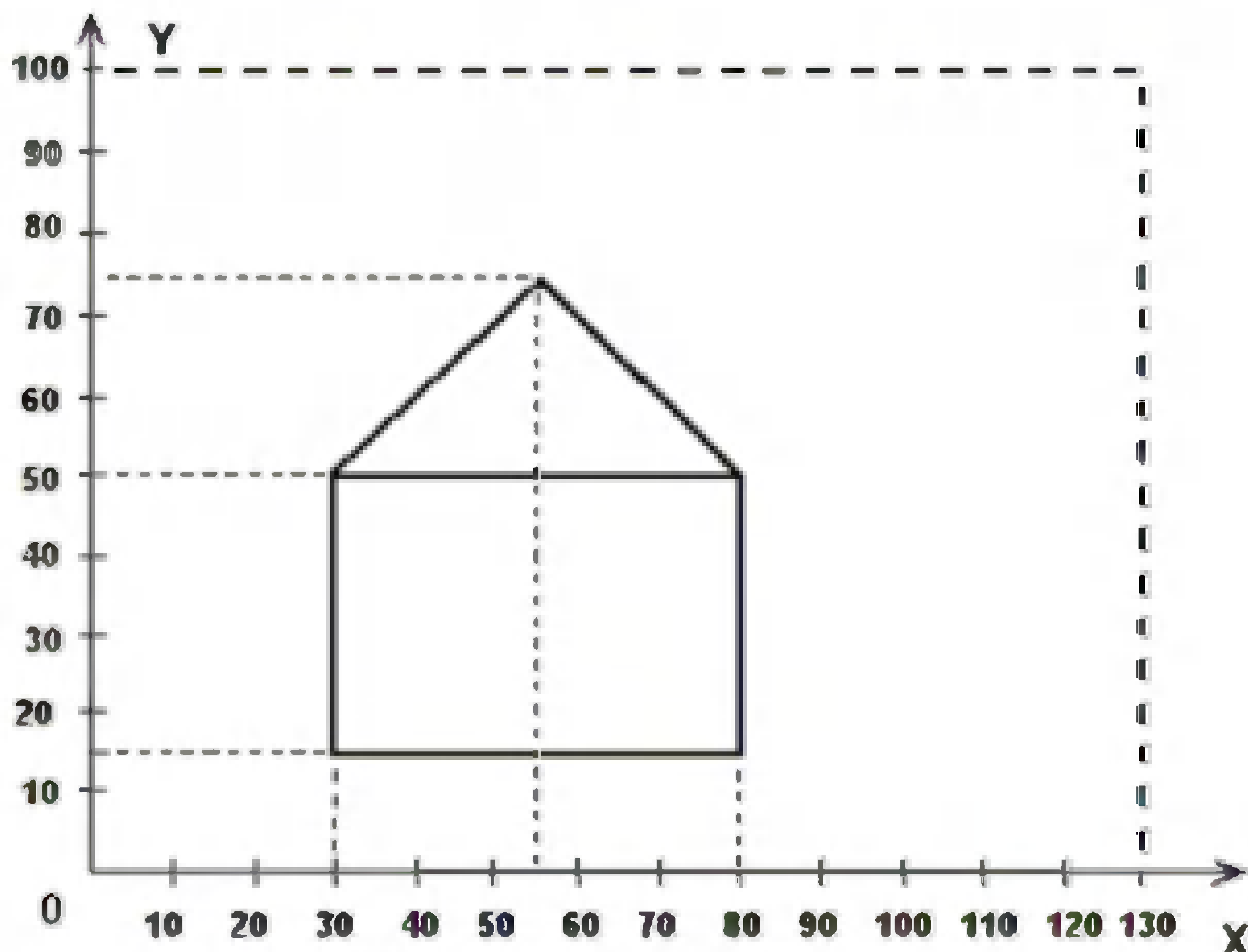
Для установки системы координат на экране служит оператор WINDOW:

WINDOW (X1, Y1) - (X2, Y2),

где (X1, Y1) — координаты левого нижнего угла экрана; (X2, Y2) — координаты правого верхнего угла экрана.

Другими словами, координаты по оси абсцисс могут меняться от X1 до X2, а координаты по оси ординат могут меняться от Y1 до Y2.

Прежде чем нарисовать рисунок на экране, его нужно сначала нарисовать на бумаге, выбрать удобную систему координат и рассчитать координаты точек рисунка. Для нашего домика удобно выбрать следующую систему координат:



Здесь пунктиром условно изображен экран дисплея. Такому размеру экрана соответствует оператор масштаба `WINDOW (0, 0) — (130, 100)`. Теперь можно выписать координаты нужных нам точек: точка 1 — (30, 15); точка 2 — (80, 15); точка 3 — (80, 50); точка 4 — (30, 50); точка 5 — (55, 75). Для получения рисунка на экране остается только соединить эти точки линиями так, как написано в начале параграфа.

§ 3. Точки и линии

Современные языки программирования содержат операторы, позволяющие выводить на экран некоторые геометрические объекты — точки, линии, окружности, дуги, прямоугольники и др., из которых строится рисунок. Эти объекты называются графическими примитивами языка.

Простейший графический примитив — точка высвечивается на экране при помощи оператора

```
PSET (X, Y), C.
```

Этот оператор дает машине команду: «высветить на экране точку с координатами (X, Y) цветом C». Здесь C — параметр цвета, который может принимать значения от 0 до 15, как написано в § 1. Если параметр цвета не указан, то точка будет иметь белый цвет.

Прямую линию чертит оператор:

LINE (X1, Y1) — (X2, Y2), C,

где C — уже знакомый нам параметр цвета. Этот оператор дает машине команду: «провести прямую линию из точки с координатами (X1, Y1) в точку с координатами (X2, Y2)

цветом C». Если цвет не указать, линия получится белого цвета.

Теперь мы можем написать программу, которая нарисует на экране домик:

```
SCREEN 12
WINDOW (0, 0) — (130, 100)
LINE (30, 15) — (80, 15)
LINE (80, 15) — (80, 50)
LINE (80, 50) — (30, 50)
LINE (30, 50) — (30, 15)
LINE (30, 50) — (55, 75)
LINE (55, 75) — (80, 50)
```

Когда будет выполнен оператор LINE (30, 15) — (80, 15), луч остановится в точке с координатами (80, 15). Следующую линию надо провести из этой точки в точку с координатами (80, 50). Если линия проводится из той точки, где находится луч в данный момент, то координаты этой точки можно не указывать. Вторую линию можно провести при помощи оператора LINE — (80, 50). Этот оператор дает машине команду «провести линию из точки, где находится луч, в точку с координатами (80, 50)».

Когда мы рисуем домик на бумаге, то можем одной непрерывной линией соединить точки 1 — 2 — 3 — 4 — 1, затем одной непрерывной линией построить крышу 4 — 5 — 3. Непрерывную линию на экране можно получить при помощи «укороченных» операторов LINE — (X, Y):

```
SCREEN 12
WINDOW (0, 0) — (130, 100)
LINE (30, 15) — (80, 15)
LINE — (80, 50)
LINE — (30, 50)
LINE — (30, 15)
LINE (30, 50) — (55, 75)
LINE — (80, 50)
```

§ 4. Прямоугольники

В нашей программе первые 4 оператора LINE строят прямоугольник — стену домика. Этот же прямоугольник можно построить при помощи одного оператора.

LINE (30, 15) — (80, 50), C, B.

Этот оператор даст машине команду: «построить прямоугольник с диагональю (30, 15) — (80, 50) цветом C». Буква B здесь означает, что строится

именно прямоугольник, а не линия, причем диагональ прямоугольника рисоваться не будет.

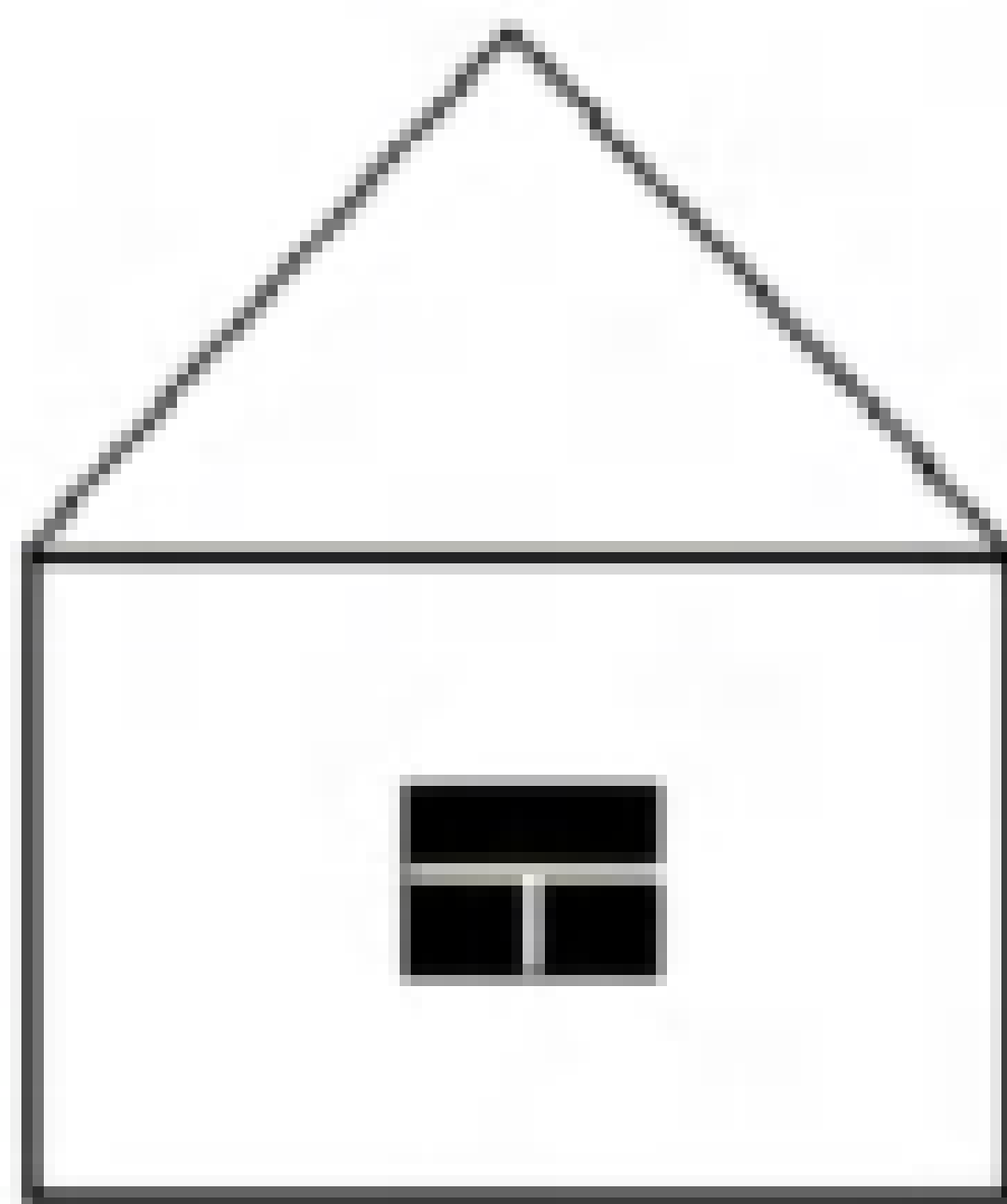
В этом операторе можно не указывать параметр цвета, но тогда необходимо перед параметром В ставить две запятые: `LINE (30, 15) — (80, 50),, В`. Запятые отмечают место, предназначенное для параметра цвета. Если параметр цвета не указан, то говорят, что рисунок рисуется текущим цветом.

Новый оператор позволит значительно сократить нашу программу:

```
SCREEN 12
WINDOW (0, 0) — (130, 100)
LINE (30, 15) — (80, 50),, В
LINE (30, 50) — (55, 75)
LINE — (80, 50)
```

Вместо диагонали (30, 15) — (80, 50) можно указать другую диагональ: (30, 50) — (80, 15); получится тот же прямоугольник.

Если вместо параметра В поставить ВF, то получится закрашенный прямоугольник. В качестве упражнения нарисуйте у нашего домика окошко, составленное из трех закрашенных прямоугольников:



§ 5. Форматы оператора LINE. Установка текущего цвета

Перечислим те форматы оператора LINE, с которыми мы познакомились.

`LINE (X1, Y1) — (X2, Y2)` — провести линию из точки с координатами (X1, Y1) в точку с координатами (X2, Y2) текущим цветом.

`LINE (X1, Y1) — (X2, Y2), С` — провести линию из точки с координатами (X1, Y1) в точку с координатами (X2, Y2) цветом С.

`LINE — (X, Y)` — провести линию из той точки, где находится курсор, в точку с координатами (X, Y) текущим цветом.

`LINE — (X, Y), С` — провести линию из той точки, где находится курсор, в точку с координатами (X, Y) цветом С.

`LINE (X1, Y1) — (X2, Y2), С, В` — построить прямоугольник с диагональю (X1, Y1) — (X2, Y2) цветом С.

`LINE (X1, Y1) — (X2, Y2), С, ВF` — построить закрашенный прямоугольник с диагональю (X1, Y1) — (X2, Y2) цветом С.

`LINE (X1, Y1) — (X2, Y2),, В` — построить прямоугольник с диагональю (X1, Y1) — (X2, Y2) текущим цветом.

LINE (X1, Y1) — (X2, Y2),, BF — построить закрашенный прямоугольник с диагональю (X1, Y1) — (X2, Y2) текущим цветом.

Обычно текущим цветом является белый (программисты говорят — устанавливается по умолчанию). Однако текущий цвет можно установить при помощи специального оператора:

COLOR C.

Если поставить этот оператор в начале программы, то текущим цветом будет цвет C, т. е. в случае отсутствия параметра цвета в операторе LINE линии будут проводиться не белым цветом, а тем, который указан в операторе COLOR.

§ 6. Окружности и дуги

Оператор

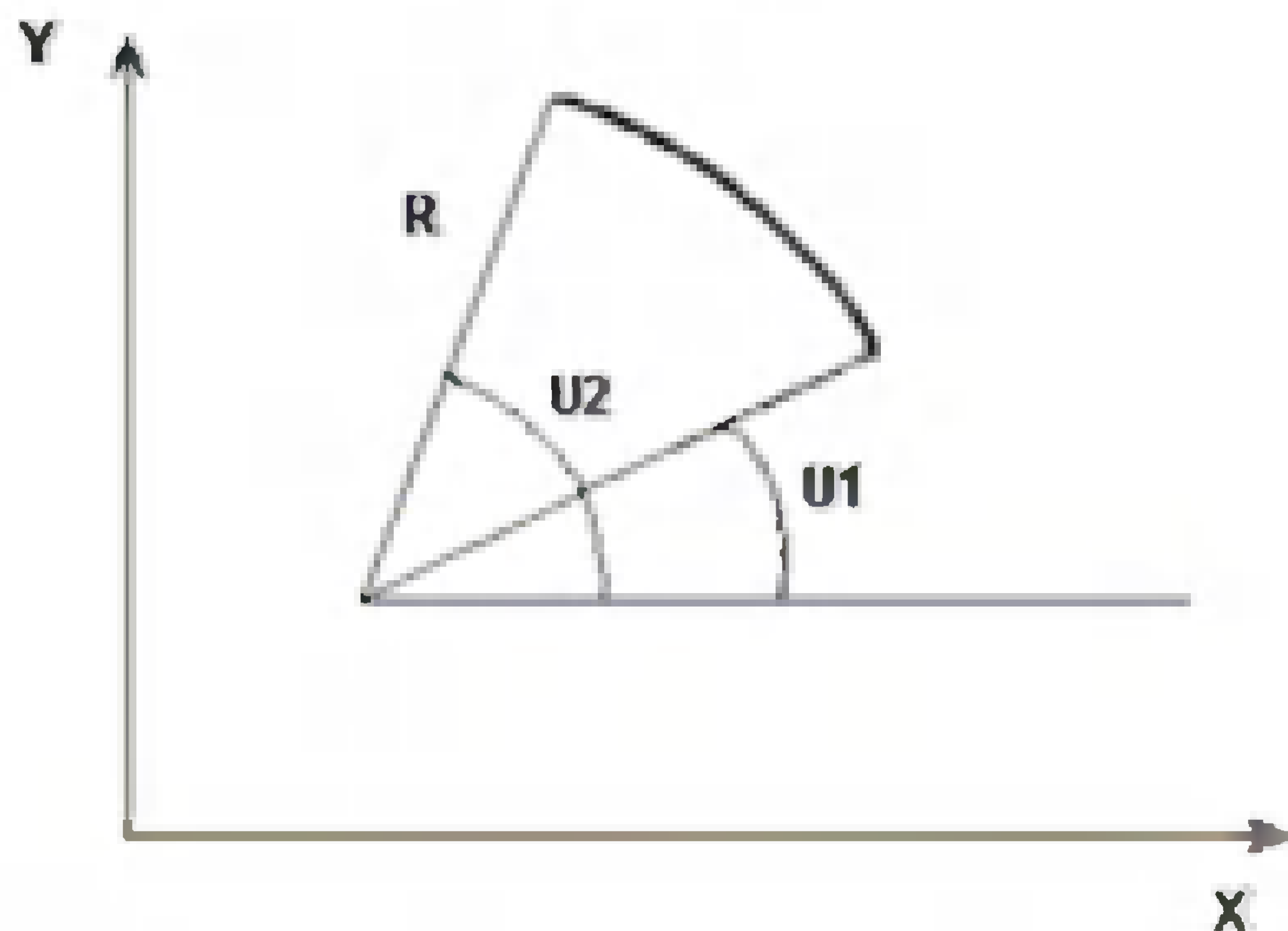
CIRCLE (X, Y), R, C

рисует окружность радиусом R с центром в точке (X, Y) цветом C. Как и прежде, если параметр C не указан, то окружность будет нарисована текущим цветом.

Этот оператор позволяет также нарисовать дугу (часть окружности). Для этого после оператора цвета нужно указать еще два параметра — начальный и конечный углы дуги:

CIRCLE (X, Y), R, C, U1, U2,

где U1 — начальный угол дуги; U2 — конечный угол.



Углы здесь указываются в радианах. Радиан — это такой угол, длина дуги которого равна радиусу. Из геометрии известно, что длина окружности в 2π раз больше радиуса, где буквой π обозначено число 3,1416... Поскольку длина окружности равна 2π длин радиусов, то полный угол 360° равен 2π радиан. Отсюда следует, что для перевода величины угла, выраженного в градусах, в радианную меру, необходимо количество градусов разделить на 360 и умножить на 2π :

$$UR = \frac{UG}{360} 2\pi,$$

где UG — угол, выраженный в градусах, UR — угол, выраженный в радианах.

Приведем перевод некоторых углов из градусной меры в радианную:

$$\begin{aligned} 1^\circ &= 0,0175, & 60^\circ &= \frac{\pi}{3}, & 120^\circ &= \frac{2}{3}\pi, & 180^\circ &= \pi, \\ 30^\circ &= \frac{\pi}{6}, & 90^\circ &= \frac{\pi}{2}, & 150^\circ &= \frac{5}{6}\pi, & 270^\circ &= \frac{3}{2}\pi. \end{aligned}$$

В качестве упражнения составим программу, которая нарисует парашют.

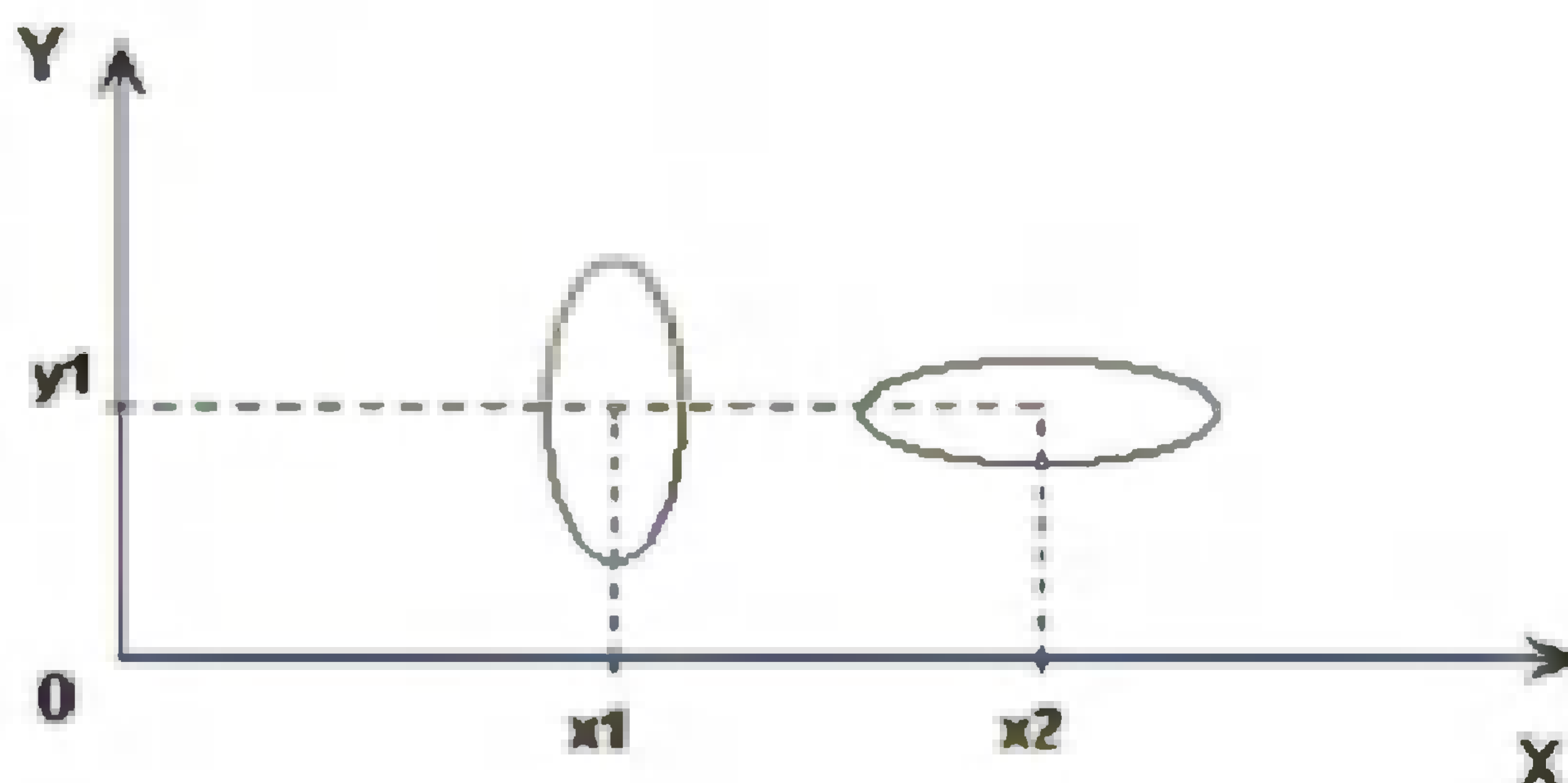


Купол парашюта и крайние стропы получатся при выполнении операторов:

```
SCREEN 12
WINDOW (0,0) — (130, 100)
CIRCLE (50, 50), 20, 15, 0, 3.142
LINE (70, 50) — (50, 20)
LINE (30, 50) — (50, 20)
```

Самостоятельно напишите операторы, которые закончат рисунок.

После параметров «начальный угол» и «конечный угол» можно указать еще один параметр, который называется «аспект». Этот параметр позволяет нарисовать эллипс (вытянутую окружность). Аспект — это отношение размеров эллипса в вертикальном направлении к размерам эллипса в горизонтальном направлении. Если $\text{аспект} > 1$, то эллипс будет вытянут в вертикальном направлении, если $\text{аспект} < 1$, то эллипс будет вытянут в горизонтальном направлении. На приведенном ниже рисунке левый эллипс нарисован оператором `CIRCLE (X1, Y1), R, , , 2`, правый — оператором `CIRCLE (X2, Y1), R, , , , 0.5`.



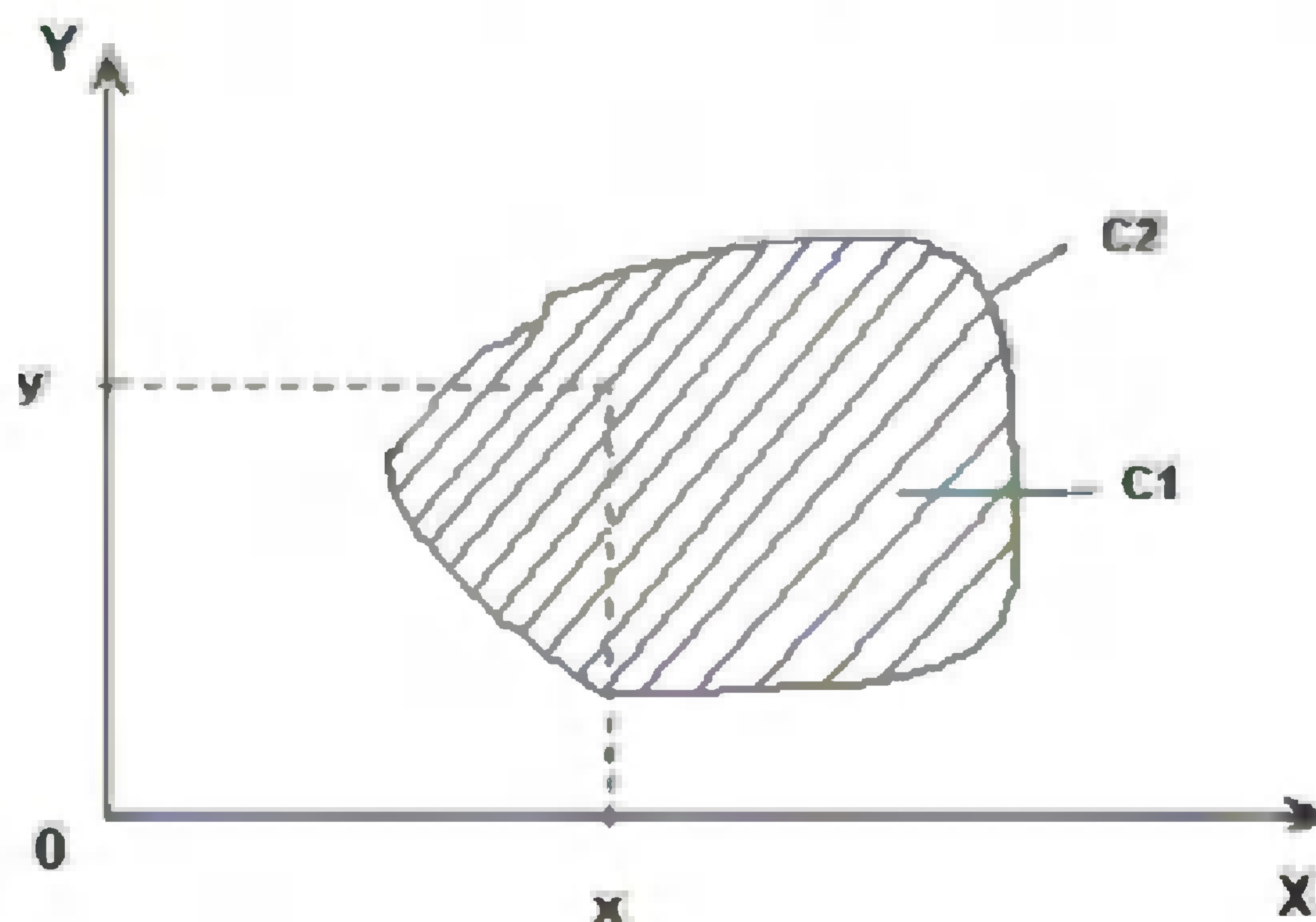
§ 7. Раскрашивание рисунка

Раскрашивание рисунков производится при помощи оператора

`PAINT (X, Y), C1, C2,`

где $C1$ — цвет закрашки; $C2$ — цвет границы.

Этот оператор заполняет цветом $C1$ участок экрана, ограниченный замкнутой линией. $C2$ — цвет, которым нарисована эта линия (цвет границы), а (X, Y) — координаты любой точки, лежащей внутри этого участка.



Оператор `PAINT (X, Y), C1, C2` как бы опрокидывает ведро с краской в точку с координатами (X, Y) . Краска «растекается» по экрану до тех пор, пока не наткнется на линию цвета $C2$. На этой линии краска останавливается.

Линия, ограничивающая закрашиваемый участок экрана, обязательно должна быть замкнутой (граница не должна иметь «дырок»). Если в границе имеется дырочка, то краска «проливается» через нее и закрашивает весь экран.

Такой же эффект получится, если неправильно указан цвет границы. Краска «разливается» по экрану до тех пор, пока не будет достигнута линия, нарисованная цветом $C2$. Если цвет границы не совпадает с цветом, указанным параметром $C2$, то краска заполнит весь экран.

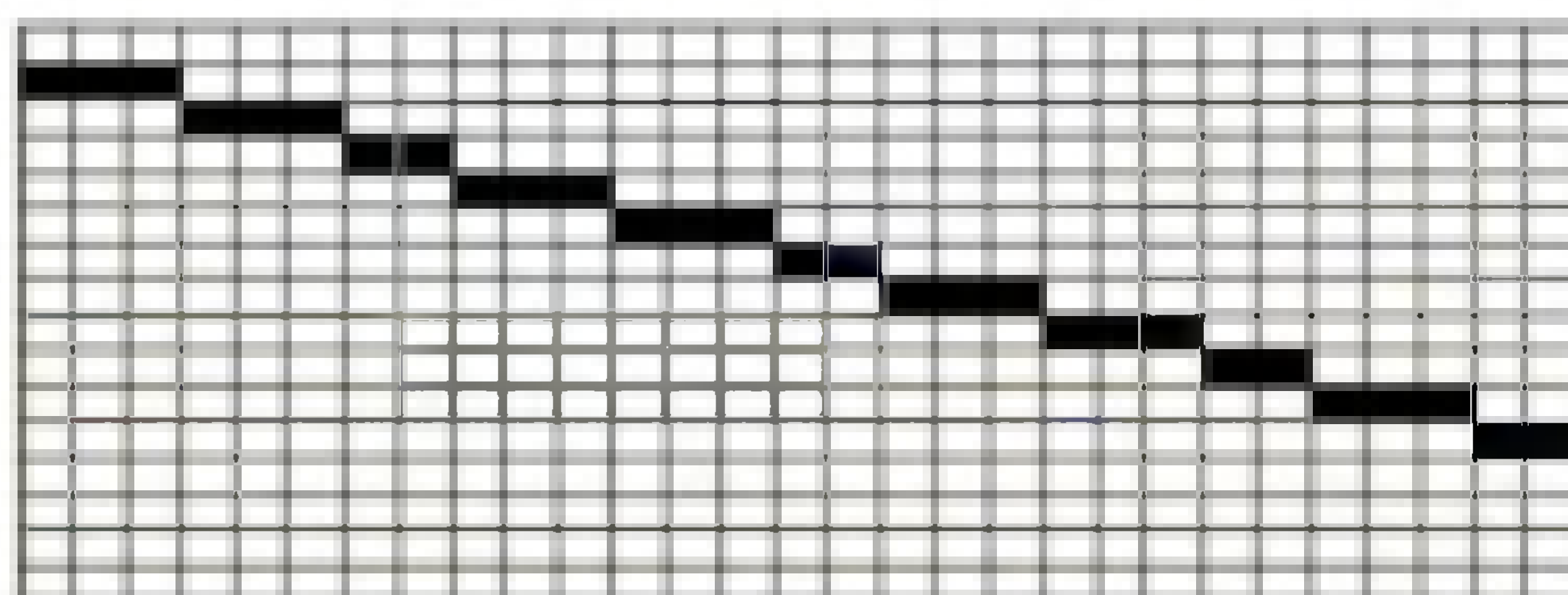
Еще одна распространенная ошибка — указываются координаты точки, лежащей на границе. Тогда краска вообще не «разливается» по экрану и рисунок не закрашивается. Чтобы избежать этой ошибки, можно при помощи оператора `PSET (X, Y)` «посмотреть», правильно ли мы указали координаты точки, в которую будем «выливать» краску. Если точка выбрана правильно, оператор `PSET (X, Y)` меняется на оператор `PAINT (X, Y), C1, C2`.

§ 8. Экранная система координат. Окно просмотра

До сих пор мы пользовались удобной нам системой координат, которую устанавливали при помощи оператора `WINDOW`. Но компьютер имеет свою собственную систему координат, которая устанавливается оператором `SCREEN`.

Изображение на экране компьютера не является непрерывным. Компьютер строит изображение из отдельных светящихся прямоугольников. Эти прямоугольники называются пикселями. Пиксели очень маленькие и неразличимы невооруженным глазом, поэтому изображение на экране кажется сплошным.

На самом деле все линии на экране — и прямые, и кривые — являются ступенчатыми. Прямая линия, построенная из пикселей, выглядит примерно следующим образом.



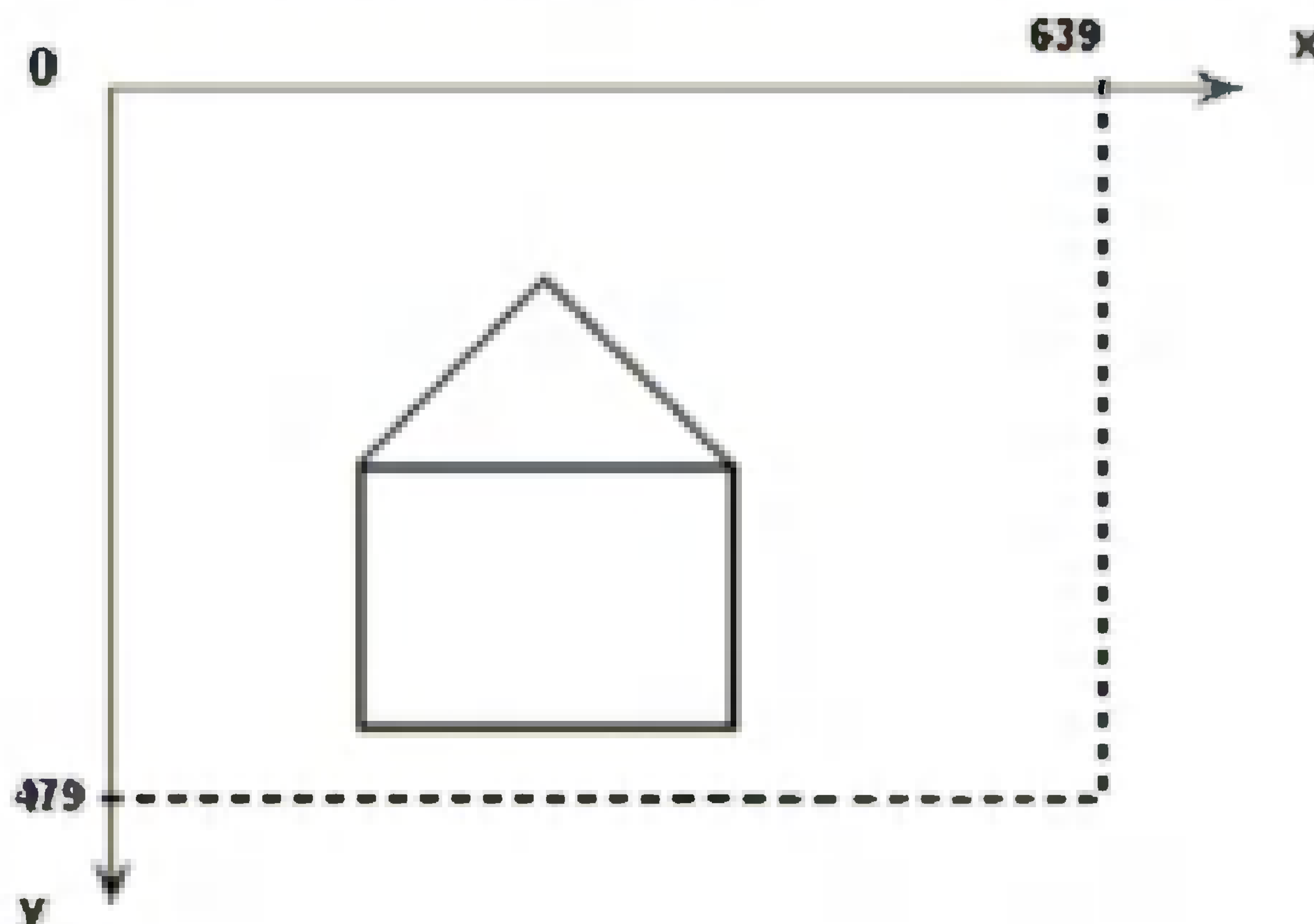
Поэтому ровными являются лишь горизонтальные и вертикальные прямые линии. На наклонных линиях даже невооруженным глазом видны ступеньки. Особенно отчетливо эти ступеньки проявляются на овальных линиях.

Количество пикселей по горизонтали и вертикали называется разрешением экрана. В режиме SCREEN 12 по горизонтали располагаются 640 пикселей, по вертикали — 480. Разрешение экрана — 640×480 .

Другие режимы устанавливают другое разрешение экрана. В режиме SCREEN 7 разрешение равно 320×200 , т. е. прямоугольники-пиксели являются более крупными, а изображение более грубым, чем в режиме SCREEN 12.

Оператор SCREEN автоматически устанавливает систему координат, обусловленную разрешением экрана. Начало координат располагается в левом верхнем углу экрана, ось X направлена слева направо, ось Y — сверху вниз. Пиксели нумеруются с 0, поэтому диапазон изменения координат в режиме SCREEN 12 следующий: $0 \leq x \leq 639$, $0 \leq y \leq 479$.

Координаты угловых точек нашего домика в этой системе координат: 1 (157, 408), 2 (393, 408), 3 (393, 240), 4 (157, 240), 5 (271, 120).



Координаты в экранной системе координат называются абсолютными, координаты, в которых выполнен рисунок, называются мировыми. Роль оператора WINDOW заключается в том, чтобы пересчитать мировые координаты в абсолютные.

Иногда возникает необходимость занять рисунком не весь экран, а только часть его. Часть экрана, занятая рисунком, называется окном просмотра. До сих пор окно просмотра занимало у нас весь экран. В QBASIC имеется оператор, который позволяет установить нужное окно просмотра на экране. Формат оператора

```
VIEW (X1, Y1) — (X2, Y2),
```

где (X1, Y1) — координаты левого верхнего угла окна просмотра; (X2, Y2) — координаты правого нижнего угла окна просмотра.

Если мы напишем в программе этот оператор, то оператор WINDOW будет работать только внутри выделенного прямоугольника. Например, если мы хотим нарисовать домик в правом верхнем углу экрана, то нужно в начале программы написать:

```
SCREEN 12  
VIEW (320, 0) — (639, 240)  
WINDOW (0, 0) — (130, 100)
```

У оператора VIEW есть и другая модификация:

```
VIEW SCREEN (X1, Y1) — (X2, Y2) .
```

В этом случае оператор WINDOW устанавливает систему координат на весь экран, но высвечивается только та часть рисунка, которая попадает в выделенное окно. Если мы в начале программы напишем

```
SCREEN 12  
VIEW SCREEN (320, 0) — (639, 240)  
WINDOW (0, 0) — (130, 100),
```

то «проявится» только та часть домика, которая попадет в верхнюю правую четверть экрана.

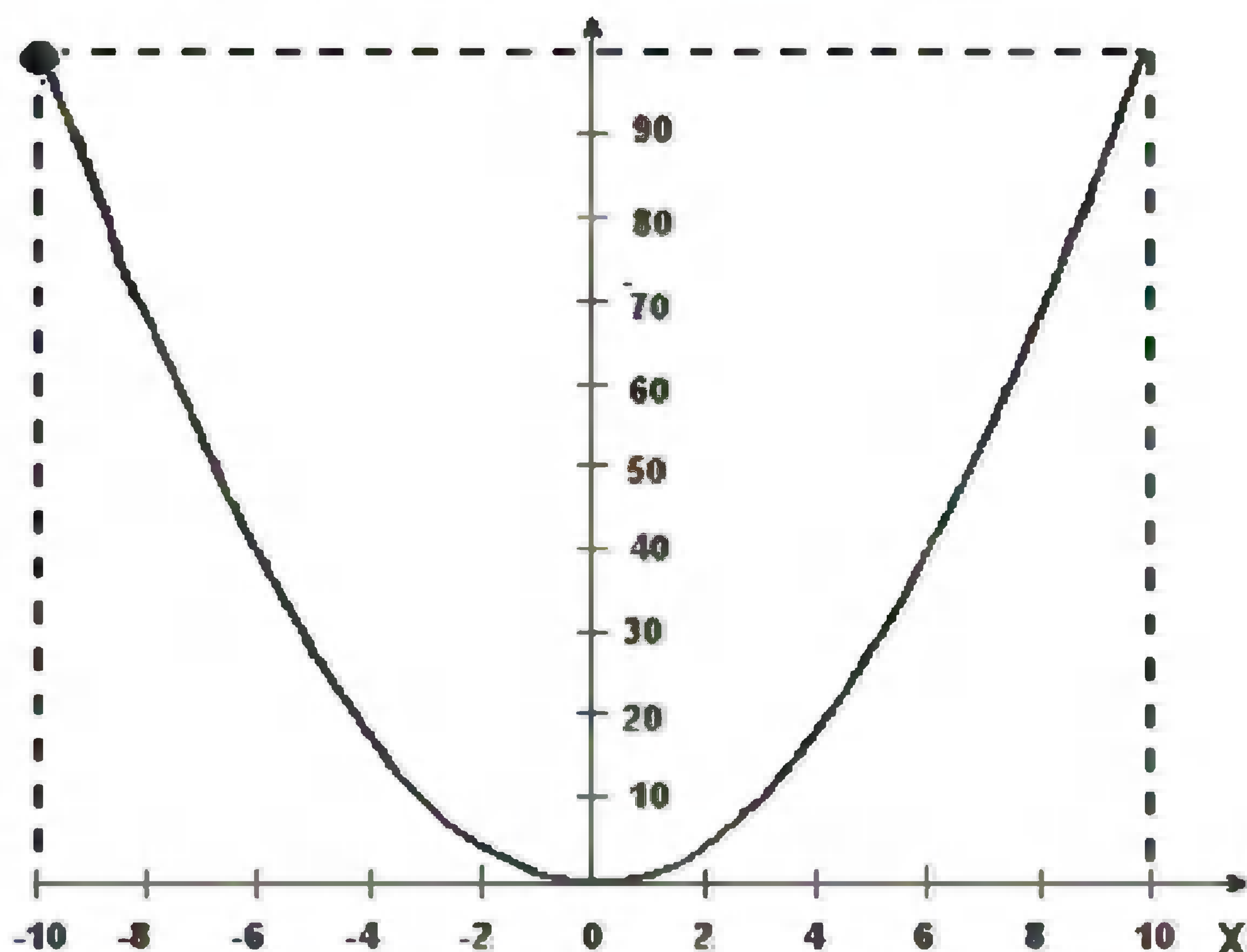
§ 9. Алгоритм построения графиков функций

Прежде чем строить график функции на экране компьютера, нужно его построить на бумаге. Рисунок поможет нам правильно выбрать масштаб. Иначе можно ошибиться.

Рассмотрим процесс построения графика функции $y = x^2$. Построим график этой функции на отрезке $[-10, 10]$. При этом значения функции будут изменяться в пределах $0 \leq y \leq 100$. Эти предельные значения функции и следует использовать в качестве параметров в операторе масштаба

```
WINDOW (-10, 0) — (10, 100) .
```

Тогда график у нас займет весь экран:



Когда мы строим график на бумаге, то после установки системы координат мы определяем положение точки, с которой начинаем вычерчивать график, и ставим в эту точку карандаш. То же самое нужно сделать на экране. В программе необходимо вычислить координаты начальной точки и дать машине команду: «поставить луч в начальную точку».

На рисунке начальная точка обозначена кружком. Очевидно, ее координаты $(-10, 100)$. Команда «поставить луч в точку с координатами (X, Y) » дается оператором `PSET (X, Y)`.

Теперь нужно «вести» луч по экрану так, чтобы получился график функции $y = x^2$. Очевидно, нужно вычислить координаты нескольких точек и провести через них линию. Вычисляются координаты нужных нам точек при помощи оператора цикла. В качестве параметра цикла используется аргумент, а значение функции вычисляется в теле цикла. Затем оператором `LINE — (X, Y)` луч перемещается в нужную точку. В результате получается такая программа:

```
SCREEN 12
WINDOW (-10, 0) — (10, 100)
X = -10: Y = 100
PSET (X, Y)
FOR X = -10 TO 10
Y = X ^ 2
LINE — (X, Y)
NEXT
```

Оператор `LINE — (X, Y)` проводит отрезок прямой. Поэтому наш график будет представлять собой ломаную линию, состоящую из 20 отрезков.

Чтобы график на экране выглядел более «гладким», необходимо укоротить отрезки, из которых он построен. Для обычных «школьных» графиков достаточно увеличить количество отрезков до 100. Это означает, что параметр X в цикле нужно изменять с шагом, равным $1/100$ от диапазона изменения аргумента. Обозначим начальное значение аргумента X_1 , конечное значение X_2 и преобразуем программу следующим образом:

```
SCREEN 12
WINDOW (-10, 0) - (10,100)
X1 = -10: X2 = 10
H = (X2 - X1) / 100
Y1 = X1 ^ 2
PSET (X1, Y1)
FOR X = X1 TO X2 STEP H
  Y = X ^ 2
  LINE - (X, Y)
NEXT
```

Теперь сформулируем задачу в общем виде: построить график функции $y = f(x)$ на отрезке $[x_1, x_2]$. Алгоритм построения графика следующий:

1. Определить область значений функции на этом отрезке.
2. Исходя из диапазона значений функции и аргумента установить масштаб экрана.
3. Вычислить шаг изменения аргумента $H = (X_2 - X_1) / 100$.
4. Вычислить координаты начальной точки $X = X_1, Y = f(X_1)$.
5. Установить луч в начальную точку.
6. Организовать цикл с аргументом в качестве параметра. Параметр цикла изменяется от X_1 до X_2 с шагом H .
7. В цикле вычислять значения функции $y = f(x)$ и перемещать луч в точку с координатами (x, y) .

§ 10. Функция, определяемая пользователем

В алгоритме построения графиков функций, написанном в предыдущем параграфе, используется общее обозначение функции $y = f(x)$. Этот алгоритм пригоден для построения графиков любых функций, а не только рассмотренной нами функции $y = x^2$. Однако в программе построения графика стоит именно эта функция. Чтобы построить график другой функции, например $y = x^3$, нужно во всех строках программы выражение $y = x ^ 2$ поменять на $y = x ^ 3$. В нашей программе функция записана всего два раза, и сделать это не трудно. Но бывают громоздкие программы, в которых одна и та же функция встречается много раз. В языке QBASIC имеется оператор, который позволяет дать функции некоторое имя (определить функцию) и затем вместо самой функции ставить ее имя. Нашей функции $y = x^2$ можно присвоить имя A следующим образом:

```
DEF FNA (X) = X ^ 2.
```

Теперь в программе вместо $Y = X^2$ можно писать $Y = \text{FNA}(X)$:

```
DEF FNA (X) = X ^ 2
SCREEN 12
WINDOW (-10, 0) - (10, 100)
X1 = -10: X2 = 10
H = (X2 - X1) / 100
Y1 = FNA (X1)
PSET (X1, Y1)
FOR X = X1 TO X2 STEP H
Y = FNA (X)
LINE - (X, Y)
NEXT
```

При помощи этой программы можно построить график любой непрерывной функции. Нужно только в оператор описания функции поставить соответствующую формулу, определить диапазон значений функции на заданном отрезке и установить соответствующий масштаб.

В одной и той же программе можно описать несколько функций, присвоив им различные имена:

```
DEF FNA (X) = X ^ 2
DEF FNB (X) = X ^ 3.
```

Здесь функция $y = x^2$ будет иметь имя А; функция $y = x^3$ будет иметь имя В.

§ 11. Построение графика функции с особыми точками

График функции $y = \frac{1}{x}$ состоит из двух ветвей. В точке $x = 0$ функция не определена. Эта точка для данной функции называется особой точкой, или точкой разрыва.

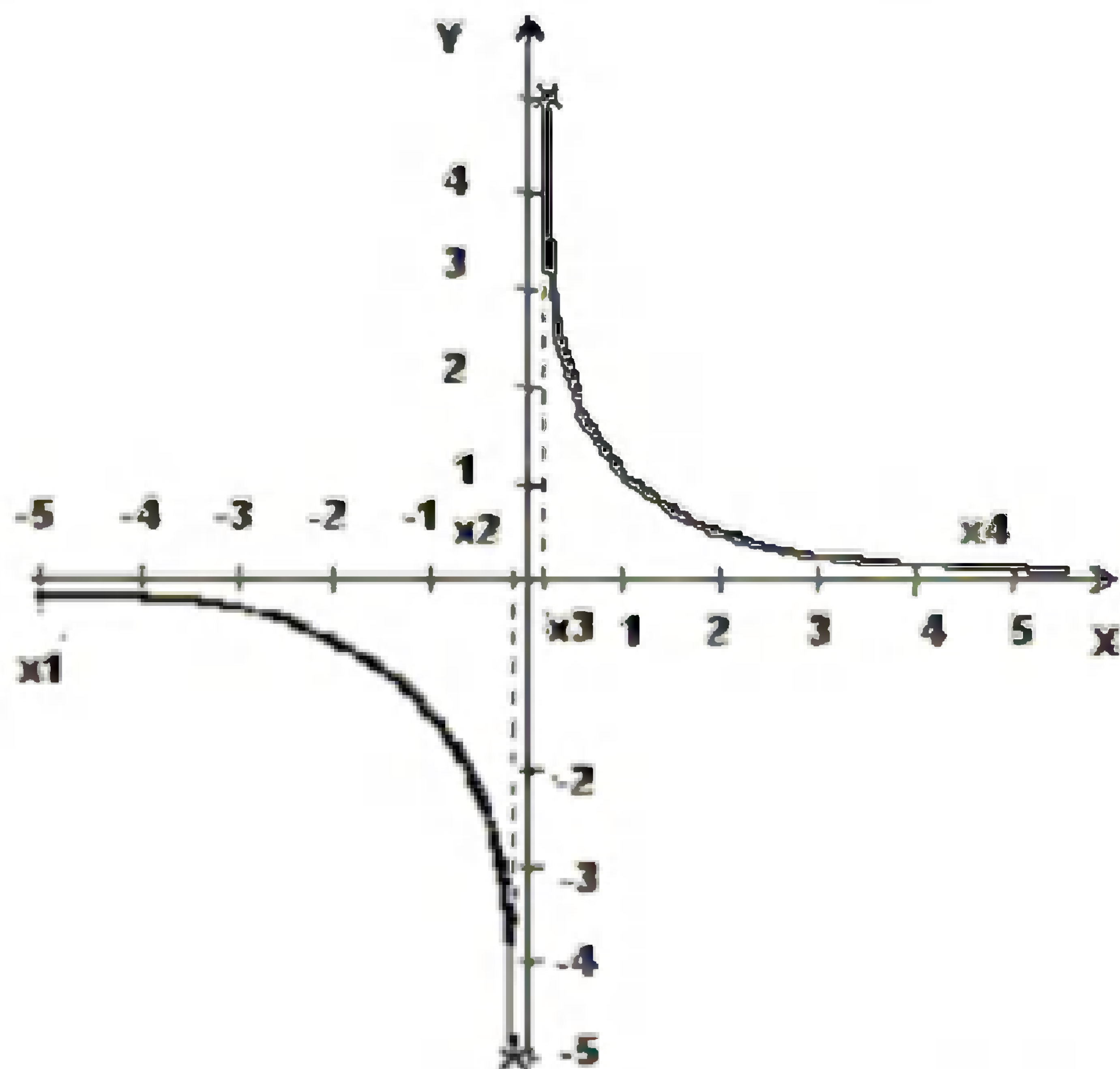
Наличие точки разрыва на графике осложняет алгоритм его построения. Во-первых, каждую ветвь придется строить при помощи отдельных операторов цикла, следовательно, программа значительно удлинится. Во-вторых, придется повозиться с начальными и конечными значениями параметра цикла. Теперь уже нельзя просто поставить в оператор цикла граничные значения отрезка, на котором строится функция. Требуется аккуратно обойти точку разрыва.

Построим график функции $y = \frac{1}{x}$ на отрезке $[-5, 5]$. Сначала строим левую ветвь.

Положение начальной точки рассчитываем, как и ранее: $x1 = -5$, $y1 = \frac{1}{x1}$. Начальное значение параметра цикла определяется наименьшим значением аргумента. А вот конечное значение параметра цикла нужно выбрать таким, чтобы не выйти за пределы экрана, поскольку при приближе-

нии аргумента к 0 функция уходит в бесконечность. Процесс вычерчивания гиперболы следует прервать в точке, лежащей на краю экрана (или области просмотра). На рисунке эта точка отмечена крестиком. Обозначим ее координаты (x_2, y_2) .

Координата y_2 — это, очевидно, наименьшее значение функции на отрезке построения. Но поскольку у нас теперь нет наименьшего значения функции, то нам придется самим выбрать тот интервал по оси ординат, на котором мы будем строить график, и, исходя из этого интервала, определить x_2 .



Возьмем интервал по оси ординат такой же, как по оси абсцисс: $-5 \leq y \leq 5$. Тогда график будет иметь наиболее привычный нам вид. Следовательно, $y_2 = -5$.

Координату x_2 определяем из уравнения $y_2 = \frac{1}{x_2}$; $x_2 = \frac{1}{-5} = -0,2$. Теперь можно написать программу, которая построит левую ветвь параболы:

```
DEF FNA (X) = 1 / X
SCREEN 12
WINDOW (-5,-5) - (5, 5,)
X1=-5: Y1 = FNA (X1)
X2 = -0.2
H = (X2 - X1) / 100
PSET (X1, Y1)
FOR X = X1 TO X2 STEP H
Y = FNA (X)
LINE - (X, Y)
NEXT
```

Для построения правой ветви графика необходимо рассчитать координаты начальной точки этой ветви. На рисунке эта точка также обозначена крестиком. Обозначим ее координаты (x_3, y_3) . Очевидно, $y_3 = 5$. Отсюда $x_3 = \frac{1}{y_3} = 0,2$. Теперь можно написать программу, которая нарисует обе ветви графика:

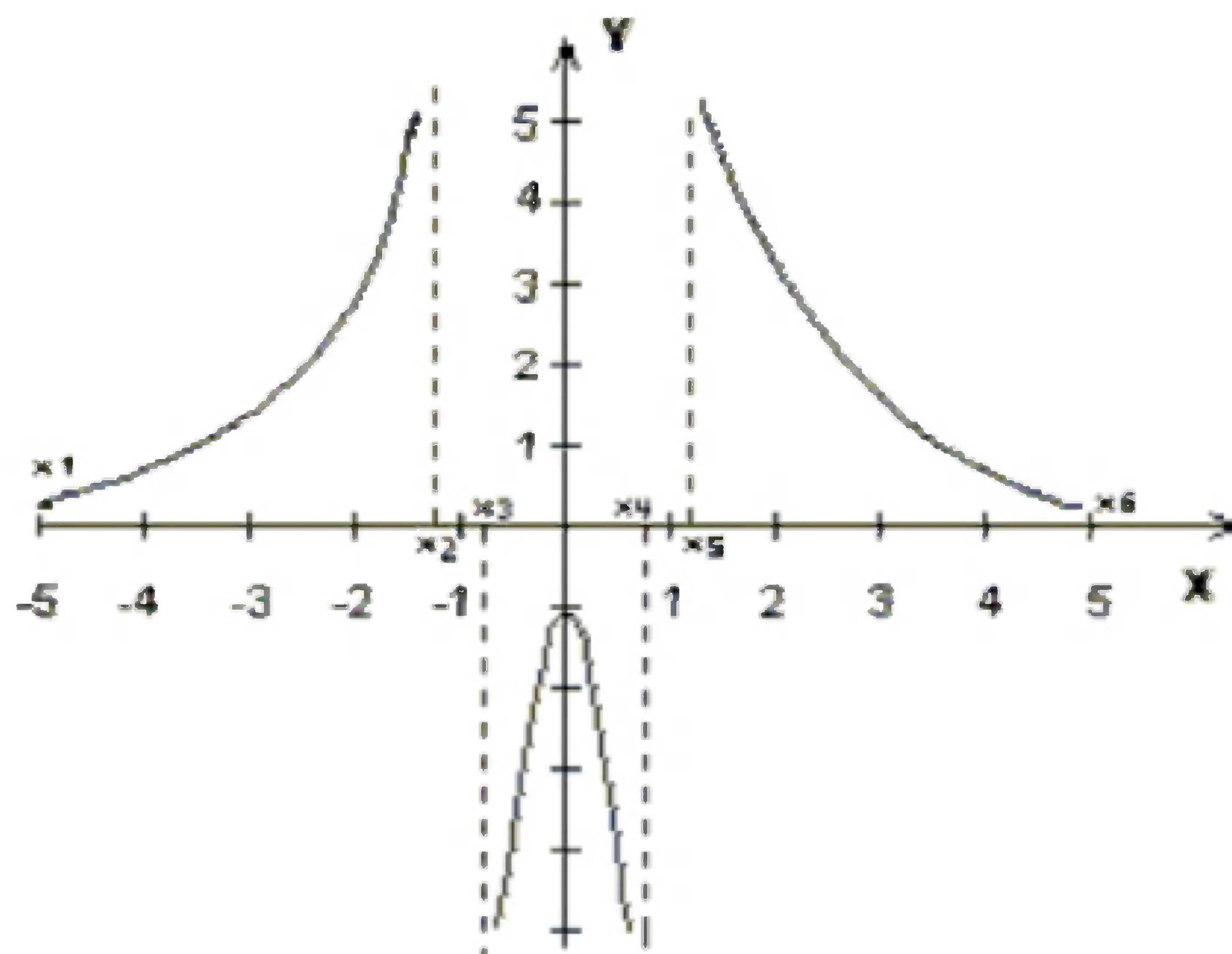
```
DEF FNA (X) = 1 / X
SCREEN 12
WINDOW (-5,-5) - (5, 5,)
X1=-5: Y1 = FNA (X1)
X2 = -0.2
X3 = 0.2: Y3 = 0.2
X4 = 5
H = (X2 - X1) / 100
PSET (X1, Y1)
FOR X = X1 TO X2 STEP H
Y = FNA (X)
LINE - (X, Y)
NEXT
H = (X4 - X3) / 100
PSET (X3, Y3)
FOR X = X3 TO X4 STEP H
Y = FNA (X)
LINE - (X, Y)
NEXT
```

Обычные «школьные» функции с особыми точками — это дробные функции со знаменателем, обращающимся в 0 при некоторых значениях аргумента. Если такая функция имеет одну особую точку, то график будет иметь две ветви, если функция имеет две особые точки, то график будет иметь три ветви и т. д.

Рассмотрим функцию $y = \frac{1}{|x| - 1}$. Знаменатель этой функции обращается в 0 при двух значениях аргумента: $x = 1$ и $x = -1$. Следовательно, график будет иметь три ветви. Для определения масштаба нарисуем примерный вид графика. Если расположить начало координат в середине экрана, то можно взять следующие значения аргумента и функции: $-5 \leq x \leq 5$; $-5 \leq y \leq -5$.

Обозначим границы отрезков, на которых строятся ветви графика, через $x_1, x_2, x_3, x_4, x_5, x_6$. Очевидно, $x_1 = -5, x_6 = 5$. Для значений аргумента $x = x_2$ и $x = x_5$ функция равна 5. Значения x_2 и x_5 находим из уравнения:

$$5 = \frac{1}{|x| - 1}.$$



$$\text{I. } 5 = \frac{1}{-x_2 - 1},$$

$$x_2 = -1,2$$

$$\text{II. } 5 = \frac{1}{x_5 - 1},$$

$$x_5 = 1,2$$

Для значений аргумента x_3 и x_4 функция равна -5 . Значения x_3 и x_4 находим из уравнения:

$$5 = \frac{1}{|x| - 1}.$$

$$\text{I. } 5 = \frac{1}{-x_3 - 1},$$

$$x_3 = -0,8$$

$$\text{II. } 5 = \frac{1}{x_4 - 1},$$

$$x_4 = 0,8$$

Теперь можно написать программу, которая нарисует график функции $y = \frac{1}{|x| - 1}$ на отрезке $[-5, 5]$:

```
DEF FNA (X) = 1 / (ABS (X) - 1)
SCREEN 12
WINDOW (-5, -5) - (5, 5)
X1 = -5: X2 = -1.2: X3 = -0.8
X4 = 0.8: X5 = 1.2: X6 = 5
H = (X6 - X1) / 100
Y1 = FNA(X1)
PSET (X1, Y1)
FOR X = X1 TO X2 STEP H
Y = FNA (X)
```

```
LINE — (X, Y)
NEXT
Y3 = FNA(X3)
PSET (X3, Y3)
FOR X = X3 TO X4 STEP H
Y = FNA (X)
LINE — (X, Y)
NEXT
Y5 = FNA(X5)
PSET (X5, Y5)
FOR X = X5 TO X6 STEP H
Y = FNA (X)
LINE — (X, Y)
NEXT
```

Глава V

Вычисления по рекуррентным формулам

§ 1. Арифметическая прогрессия

Проведем мысленный эксперимент. Заберемся куда-нибудь повыше и бросим вниз камень. Если мы бросаем камень без начальной скорости (т. е. просто отпускаем его), то за первую секунду камень пролетит 4,9 м, за вторую — 14,7, за третью — 24,5 м и т. д. За каждую последующую секунду камень будет пролетать расстояние на 9,8 м большее, чем за предыдущую. Если мы выпишем расстояния, которые камень пролетит за 1-ю, 2-ю и последующие секунды:

$$4,9; 14,7; 24,5; 34,3; 44,1...$$

то получим ряд чисел, который называется арифметической прогрессией. В этом ряду каждое последующее число на 9,8 м больше, чем предыдущее.

Числа, образующие арифметическую прогрессию, называются членами прогрессии. Все члены прогрессии пронумерованы. Если мы обозначим член прогрессии с номером n через a_n , то выражение «каждый последующий член на 9,8 больше, чем предыдущий», можно записать в виде формулы:

$$a_n = a_{n-1} + 9,8.$$

Эта формула носит название рекуррентной формулы (от лат. слова *recurro* — возвращаться).

При задании арифметической прогрессии рекуррентной формулой задается ее первый член a_1 и разность d . В нашем случае

$$a_1 = 4,9, d = 9,8,$$

$$a_n = a_{n-1} + d.$$

Чтобы вычислить n -й член арифметической прогрессии при помощи рекуррентной формулы, необходимо вычислить сначала 2-й член, затем 3-й и так до n -го. Конечно, в случае арифметической прогрессии удобнее

воспользоваться известной формулой, позволяющей сразу вычислить n -й член прогрессии через 1-й член и разность. Однако в научных и инженерных расчетах чаще всего приходится иметь дело с такими последовательностями, для которых невозможно указать формулу общего члена и которые задаются только рекуррентной формулой. В этом случае у нас просто нет другого выхода, кроме как вычислять одну за другой все члены последовательности вплоть до n -го.

В «докомпьютерную эру» такие расчеты представляли собой трудную задачу. В XIX в. английский математик Вильям Шенкс вычислил 707 знаков числа π , используя так называемую формулу удвоения, которая сводится к вычислениям по рекуррентным формулам. Он потратил на эту работу более 20 лет. Впоследствии оказалось, что В. Шенкс допустил ошибку в 520-м знаке. В настоящее время с помощью ЭВМ вычислено более 500 000 знаков числа π . Современные компьютеры делают эту работу за несколько часов, причем не допускают ошибок.

Но рекуррентные формулы часто применяются при вычислениях таких последовательностей, для которых можно указать формулу общего члена. На это есть много причин. Использование рекуррентных формул позволяет, как говорят программисты, экономить ресурсы компьютера. Это означает, что при использовании рекуррентных формул вычисления производятся быстрее, требуют меньше внутренней памяти и т. д. Бывают такие последовательности, которые компьютер по формуле общего члена просто не сможет вычислить (например, может возникнуть переполнение), а по рекуррентной формуле прекрасно вычисляет. С такими последовательностями мы скоро познакомимся.

Умение использовать рекуррентные формулы при работе на компьютере так же важно, как умение использовать таблицу умножения при решении математических задач. Поэтому сейчас мы научимся решать известные вам задачи из учебника алгебры, используя рекуррентные формулы, а затем перейдем к более сложным задачам, которые без компьютера не решишь.

Составим программу, которая при помощи рекуррентной формулы вычислит расстояние, которое наш камень пролетит за 5-ю секунду. Для этого нам нужно запрограммировать выполнение следующих операций:

$$a_1 = 4,9; d = 9,8$$

$$a_2 = a_1 + d$$

$$a_3 = a_2 + d$$

$$a_4 = a_3 + d$$

$$a_5 = a_4 + d$$

Очевидно, при составлении программы следует использовать оператор цикла. Цикл должен «крутиться» четыре раза, а в теле цикла должно стоять выражение

$$a_{\text{новое}} = a_{\text{старое}} + d.$$

Как вы, наверное, уже догадались, этой формуле соответствует оператор присваивания $A = A + D$. Программа будет выглядеть следующим образом:

```
A = 4.9: D = 9.8
FOR I = 1 TO 4
A = A + D
NEXT
PRINT A
```

Самое главное в этом алгоритме — «превращение» рекуррентной формулы $a_n = a_{n+1} + d$ в оператор присваивания $A = A + D$. Этот оператор означает команду: «увеличить значение переменной A на величину D и сохранить новое значение в той же переменной». Посмотрим, как работает наша программа.

При выполнении операторов, стоящих в первой строчке программы, в памяти компьютера появятся переменные

A	4.9
D	9.8

Когда цикл проработает первый раз, в памяти появится еще одна переменная и изменится значение переменной A:

A	14.7
D	9.8
I	2

После второй итерации значения переменных A и I изменятся:

A	24.5
D	9.8
I	3

После третьей итерации:

A	34.3
D	9.8
I	4

После четвертой итерации:

A	44.1
D	9.8
I	5

Отметим, что после первой итерации у нас появляется второй член прогрессии, так как первый член прогрессии мы записываем в первой строчке программы. Поэтому цикл делает четыре итерации, и параметр цикла имеет значение на 1 меньше номера вычисляемого члена прогрессии. Если начальное и конечное значение параметра цикла увеличить на 1, то значение параметра цикла будет равно номеру вычисляемого члена прогрессии. Тогда в тело цикла можно поставить оператор PRINT, который будет выводить на экран очередной член прогрессии вместе с его номером:

```
A = 4.9: D = 9.8
FOR I = 2 TO 5
A = A + D
PRINT I, A
NEXT
```

§ 2. Геометрическая прогрессия

Геометрическая прогрессия отличается от арифметической тем, что в рекуррентной формуле вместо операции сложения стоит операция умножения:

$$a_n = a_{n-1} * d.$$

Сомножитель d называется знаменателем прогрессии. В школьных учебниках для геометрической прогрессии используются другие обозначения, но это несущественно. Существенно то, что «внешний вид» алгоритма, вычисляющего n -й член геометрической прогрессии, такой же, что и у алгоритма, вычисляющего n -й член арифметической прогрессии. Этот алгоритм представляет собой цикл, в котором стоит рекуррентная формула. Разница лишь в том, что в случае арифметической прогрессии в рекуррентной формуле стоит операция сложения, а в случае геометрической прогрессии — операция умножения.

При решении задач на геометрические прогрессии сложность обычно представляет вычисление знаменателя прогрессии. В качестве примера рассмотрим две задачи.

Задача 1. Срочный вклад, положенный в банк, ежегодно увеличивается на 3 %. Каким станет вклад через пять лет, если вначале он был равен 8000 рублей?

Казалось бы, если сказано, что вклад увеличивается на какую-то величину, то здесь должна быть арифметическая прогрессия. Но величина, на которую увеличивается вклад ежегодно, не постоянна. Она равна 3 % от значения вклада за предыдущий год. Выразим величину вклада в n -ом году a_n через величину вклада в предыдущем году a_{n-1} :

$$a_n = a_{n-1} + 0,03 * a_{n-1} = a_{n-1} * (1 + 0,03) = 1,03 * a_{n-1}.$$

Таким образом, процесс роста вклада по годам описывается геометрической прогрессией со знаменателем 1,03. Здесь начальной величине вклада удобно присвоить номер 0, тогда сумма вклада через год будет иметь номер 1, сумма вклада через два года будет иметь номер 2 и т. д. Параметр цикла будет изменяться от 1 до 5:

```
A = 8000: D = 1.03
FOR I = 1 TO 5
A = A * D
NEXT
PRINT A
```

Задача 2. При каждом движении поршня разряжающего насоса из сосуда удаляется 20 % находящегося там воздуха. Вычислить, какое давление будет в сосуде через 6 движений поршня, если начальное давление равно 760 мм рт. ст.?

Фраза «из сосуда удаляется 20 % находящегося там воздуха» на математическом языке выражается следующей рекуррентной формулой:

$$a_n = a_{n-1} - 0,2 * a_{n-1} = a_{n-1} * (1 - 0,2) = 0,8 * a_{n-1}.$$

Процесс изменения давления в сосуде описывается геометрической прогрессией со знаменателем 0,8. Как и в предыдущей задаче, первому члену прогрессии удобно приписать номер 0. Тогда получим следующую программу, решающую задачу:

```
A = 760: D = 0.8
FOR I = 1 TO 6
A = A * D
NEXT
PRINT A
```

§ 3. Алгоритм вычисления суммы

Вернемся к задаче о падающем камне и посчитаем, сколько метров он пролетит за 5 секунд. Очевидно, для этого нужно посчитать сумму первых пяти членов арифметической прогрессии с первым членом 4,9 и разностью 9,8. Обозначим эту сумму буквой S .

За 1-ю секунду камень пролетит 4,9 м. Это расстояние примем за первое значение суммы S_1 . Чтобы вычислить, сколько метров камень пролетит за 2 секунды, нужно к S_1 прибавить 2-й член арифметической прогрессии: $a_2 = 14,7$; $S_2 = S_1 + a_2$. Чтобы вычислить, какое расстояние пролетит камень за 3 секунды, нужно к прежнему значению S прибавить a_3 : $S_3 = S_2 + a_3$, и т. д. Получаем последовательность, задаваемую рекуррентной формулой: $S_n = S_{n-1} + a_n$. Эта формула выражает сумму n членов арифметической прогрессии через сумму $n-1$ членов; здесь a_n — n -й член арифметической прогрессии. Чтобы решить задачу, нужно подставить полученную нами формулу в уже готовый цикл из § 1 (естественно, нужно задать начальное значение S):

```
A = 4.9: D = 9.8: S = A
FOR I = 2 TO 5
  A = A + D
  S = S + A
NEXT
PRINT S.
```

Посмотрим, как работает эта программа. После выполнения операторов, стоящих в первой строчке, в памяти компьютера появятся переменные:

A	4.9
D	9.8
S	4.9

При первой итерации значение переменной A будет равно второму члену арифметической прогрессии: $a_2 = 14,7$, а значение переменной S увеличится на эту величину: $S = 4,9 + 14,7 = 19,6$:

A	14.7
D	9.8
S	19.6

При второй итерации значение переменной A будет равно 3-му члену арифметической прогрессии $a_3 = 24,5$, а значение переменной S увеличится на эту величину $S = 19,6 + 24,5 = 44,1$:

A	24.5
D	9.8
S	44.1

После двух итераций в переменной S накопилась сумма первых трех членов арифметической прогрессии: $S = a_1 + a_2 + a_3$. Аналогичным образом после трех итераций значение переменной S будет равно сумме первых четырех членов прогрессии, и после четырех итераций в переменной S накопится сумма пяти членов арифметической прогрессии.

В заключение приведем общую схему вычисления суммы членов числовой последовательности.

Пусть у нас имеются n членов числовой последовательности $a_1, a_2, a_3, \dots, a_n$. Требуется вычислить их сумму. Входными данными будут являться 1-й член последовательности и количество слагаемых. Алгоритм вычисления суммы следующий.

1. Присвоить переменной S значение 1-го члена последовательности.
2. Организовать цикл с числом повторений n . В теле цикла:
 - а) вычислить очередное слагаемое;
 - б) увеличить S на величину очередного слагаемого.
3. Вывести значение S .

§ 4. Циклы типа «До» и типа «Пока»

До сих пор мы пользовались только одним оператором цикла, который носит название «Цикл с параметром». Этот цикл используется в тех случаях, когда из условия задачи можно сразу определить, сколько итераций нужно сделать, чтобы решить задачу. Но не всегда это удастся сделать. Рассмотрим задачу 2 из § 2, несколько изменив условие.

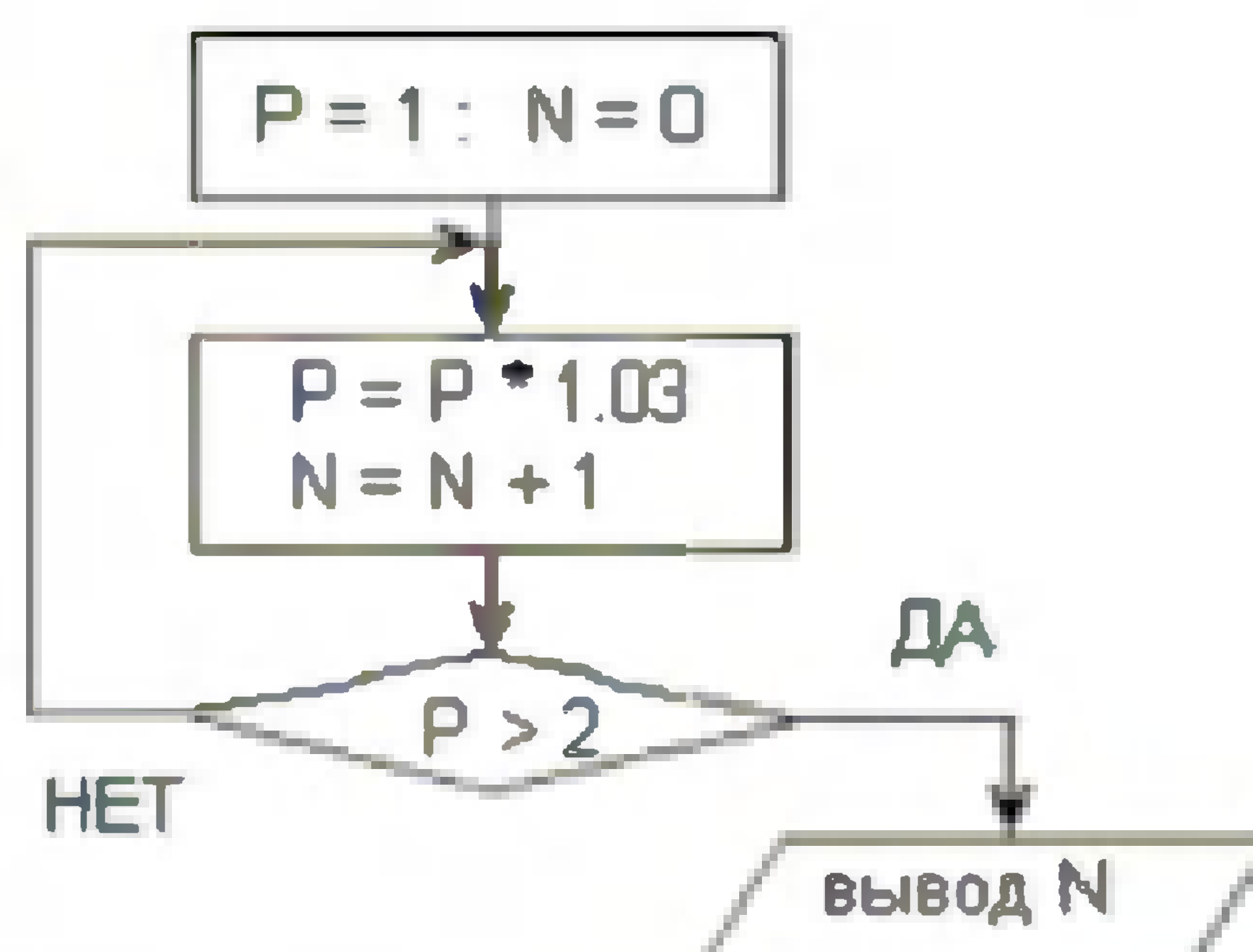
Срочный вклад, положенный в банк, ежегодно увеличивается на 3 %. Через сколько лет вклад удвоится?

Здесь мы уже не сможем воспользоваться циклом с параметром, поскольку мы не знаем, сколько лет должно пройти, чтобы вклад увеличился в два раза. Очевидно, после каждого увеличения вклада (т. е. после каждой итерации) необходимо проверять, во сколько раз увеличилась сумма вклада по сравнению с первоначальным значением, и остановиться тогда, когда условие задачи будет выполнено.

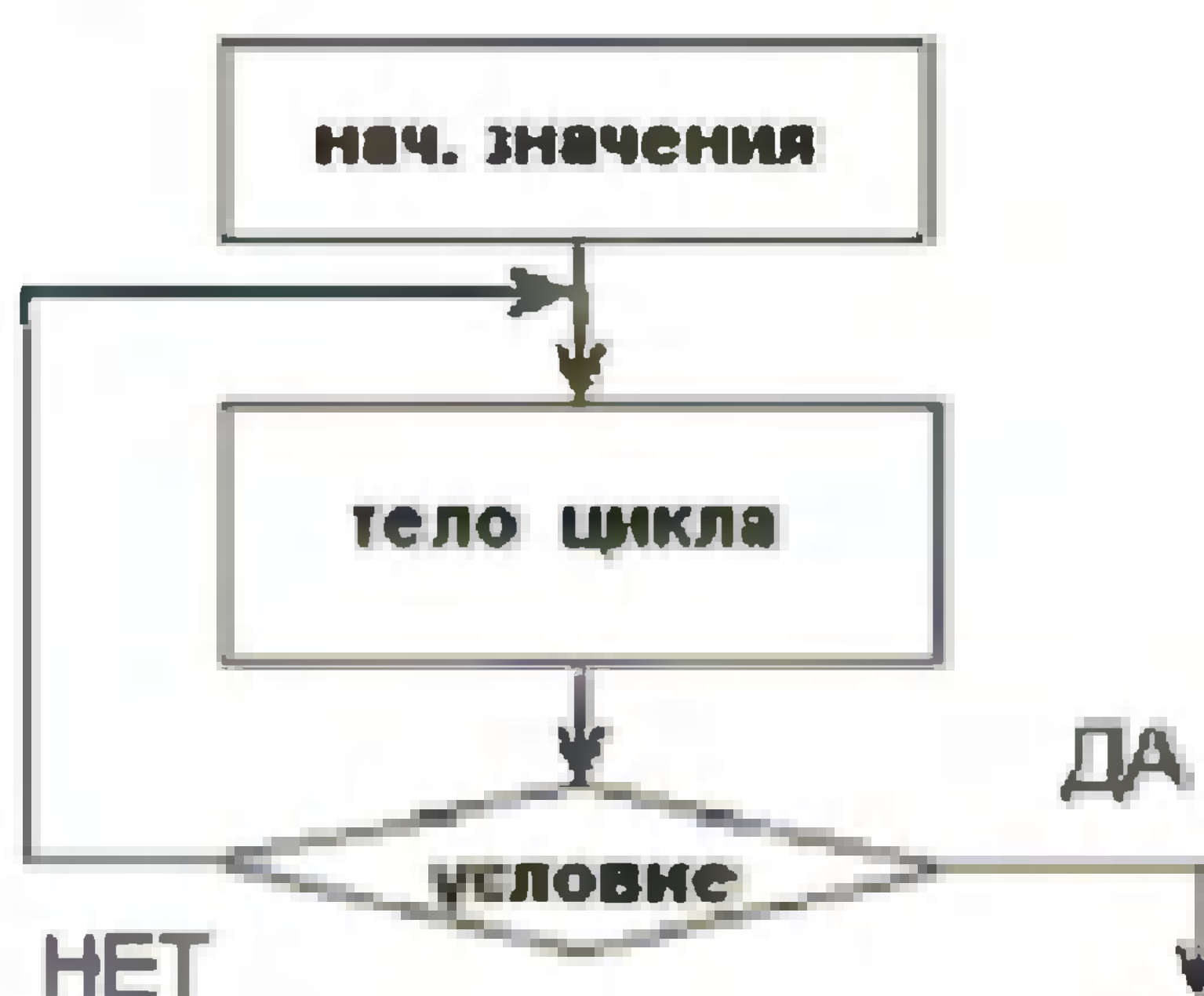
Обозначим вклад буквой P . Первоначальная величина вклада здесь неважна, поскольку любой вклад увеличится вдвое за одно и то же количество лет.

во лет. Поэтому начальное значение P можно принять за 1. Для подсчета количества лет введем переменную N . Начальное значение $N = 0$. При каждой итерации N будет увеличиваться на 1.

Значения вклада по годам образуют геометрическую прогрессию со знаменателем 1,03. Будем вычислять члены этой прогрессии до тех пор, пока не найдем первый член, больший 2. Получается следующий алгоритм:



Очевидно, эта блок-схема изображает циклический процесс, который работает до выполнения условия $P > 2$. Как только достигается выполнение этого условия, цикл прекращает работу. Такой цикл носит название цикла типа «До». Цикл работает до выполнения некоторого условия. Когда условие выполнено, цикл прекращает работу. Блок-схема цикла типа «До»:



Для реализации такого цикла в языке QBASIC существует особый оператор:

```

DO
    тело цикла
LOOP UNTIL условие.
  
```

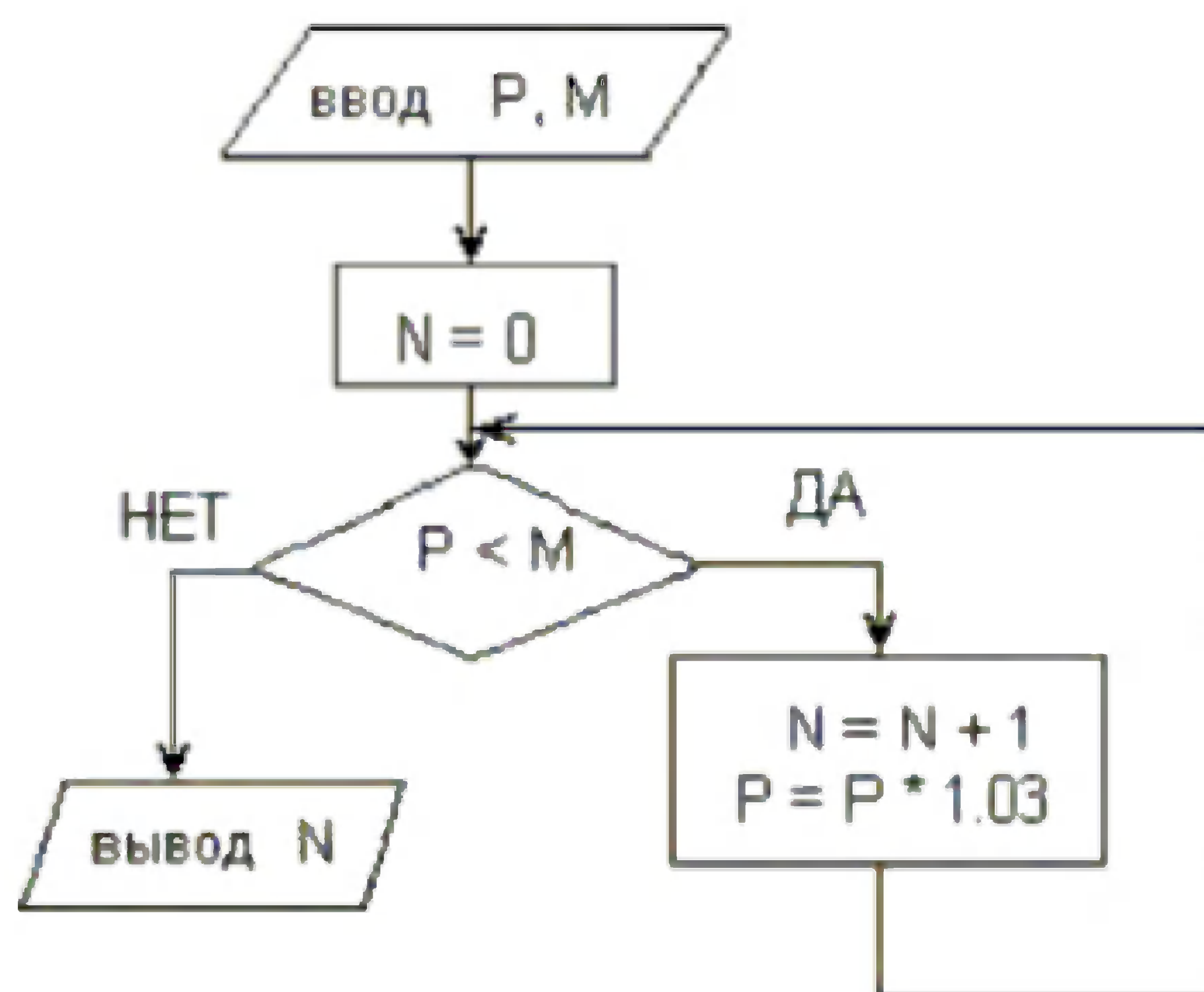
Здесь слово DO — заголовок цикла. Тело цикла пишется со следующей строки. Признаком окончания цикла является слово LOOP (по-английски — петля). После слова UNTIL (по-английски — до тех пор, пока) стоит условие, при выполнении которого цикл завершает работу. Программа, реализующая написанный выше алгоритм, выглядит следующим образом:

```

P = 1: N = 0
DO
  P = P * 1.03
  N = N + 1
LOOP UNTIL P > 2
PRINT N

```

Еще раз изменим условие задачи. Поставим вопрос так: через сколько лет вклад превысит величину M , если первоначальная сумма равна P ? Поскольку величины P и M нам заранее неизвестны, то нужно проверить, удовлетворяют ли введенные значения условию $P < M$. Если это условие не выполняется, то проценты наращивать не нужно. Получается следующий алгоритм решения задачи:



Получившийся здесь цикл называется циклом типа «Пока». Цикл работает, пока выполняется условие $P < M$. Как только это условие перестает выполняться, происходит выход из цикла. Блок-схема цикла типа «Пока»:



Оператор QBASIC, реализующий цикл типа «Пока», выглядит следующим образом:

```
DO WHILE условие
    тело цикла
LOOP.
```

В заголовке оператора цикла после слова DO стоит слово WHILE, которое по-английски означает «пока». Заголовок оператора цикла можно перевести на русский язык как: «Делать, пока выполняется условие». Признаком окончания цикла служит слово LOOP. Операторы, стоящие между строчками DO и LOOP составляют тело цикла.

Программа, решающая задачу с новым условием:

```
INPUT P, M
N = 0
DO WHILE P < M
    N = N + 1
    P = P * 1.03
LOOP
PRINT N
```

Отметим одну важную особенность цикла типа «Пока». В этом цикле условие проверяется перед входом в цикл. Если мы введем такие данные, что условие сразу не будет выполняться, то тело цикла не будет выполнено ни разу. Например, если мы введем в нашу программу значения $P = 150$, $M = 80$, то получим $N = 0$. В цикле типа «До» условие проверяется после выполнения тела цикла, поэтому цикл типа «До» всегда выполняется хотя бы один раз.

В заключение перечислим главные особенности двух новых циклов:

1. Операторы циклов типа «До» и типа «Пока» служат для организации таких циклов, в которых заранее неизвестно число повторений.
2. Цикл типа «До» выполняется до выполнения некоторого условия; цикл типа «Пока» выполняется, пока выполняется некоторое условие.
3. У цикла типа «До» условие выхода из цикла проверяется после выполнения тела цикла; у цикла типа «Пока» условие проверяется перед выполнением тела цикла.
4. Цикл типа «До» обязательно выполняется хотя бы один раз, цикл типа «Пока» может не выполняться ни разу.

§ 5. Числовые последовательности

Арифметическая и геометрическая прогрессии, с которыми мы до сих пор имели дело, являются числовыми последовательностями. Сейчас мы начнем работать с другими типами числовых последовательностей, поэтому познакомимся с этим математическим понятием поближе.

Определение. Числовой последовательностью называется функция натурального аргумента.

Что такое функция, вы знаете из курса алгебры. Функция — это правило, согласно которому какому-либо числу ставится в соответствие дру-

гое число. Например, функция $y = x^2$ числу 1 ставит в соответствие 1, числу 1,5 ставит в соответствие 2,25 и т. д. Число x называется аргументом, y — функцией. Здесь аргумент меняется от $-\infty$ до $+\infty$ и является непрерывной переменной, то есть может принимать любые действительные значения.

Если мы для функции $y = x^2$ будем брать только натуральные значения аргумента, то получим последовательность 1, 4, 9, 16, ... Здесь аргумент меняется дискретно (прерывно). Натуральные числа в математике принято обозначать буквой n , поэтому при работе с последовательностями аргумент обозначают не буквой x , а буквой n , чтобы подчеркнуть дискретный характер аргумента (иногда применяют букву i).

Числа, образующие последовательность, называются членами последовательности. Обозначают члены последовательности строчными буквами латинского алфавита, приписывая аргумент внизу в качестве индекса. Функция $y = x^2$ при преобразовании ее в последовательность записывается $a_n = n^2$ или $a_i = i^2$. Значения аргумента называют также номером члена последовательности, а саму числовую последовательность определяют как группу пронумерованных чисел.

Натуральный ряд чисел бесконечен. Поэтому числовая последовательность, по определению, бесконечна. Натуральный ряд чисел начинается с единицы, и члены числовой последовательности обычно нумеруют с единицы. Но иногда для удобства расчетов первому члену последовательности присваивается номер 0, хотя 0 не является натуральным числом.

Задать последовательность — значит указать правило, по которому можно вычислить любой член последовательности по его номеру. На уроках алгебры арифметическую и геометрическую прогрессии задают при помощи формулы общего члена:

$$a_n = a_1 + (n - 1) * d,$$

$$b_n = b_1 * q^{n-1}.$$

В вычислительной математике чаще всего задают последовательность при помощи рекуррентной формулы (если это возможно). Рекуррентная формула позволяет вычислить каждый последующий член последовательности через предыдущие. При задании числовой последовательности при помощи рекуррентной формулы задаются первые несколько членов последовательности, а затем через них вычисляются остальные. В случае арифметической и геометрической прогрессий необходимо задать только первый член, например, арифметическая прогрессия:

$$a_1 = 5, a_n = a_{n-1} + 2,$$

геометрическая прогрессия:

$$b_1 = 1, b_n = 2 * b_{n-1}.$$

Пример последовательности, в которой каждый последующий член задается через 2 предыдущих:

$$a_1 = 1, a_2 = 1, a_n = a_{n-1} + a_{n-2}.$$

Получаем последовательность чисел

$$1, 1, 2, 3, 5, 8, 13, \dots$$

Эта последовательность носит название чисел Фибоначчи.

§ 6. Факториал

В дальнейшем нам понадобится функция натурального аргумента, которая носит название «факториал». Факториалом натурального числа n называется произведение натуральных чисел от 1 до n . Обозначается факториал восклицательным знаком, который ставится справа от числа:

$$n! = 1 * 2 * 3 * 4 * \dots * n.$$

Факториал числа — это тоже число:

$$5! = 1 * 2 * 3 * 4 * 5 = 120;$$

$$6! = 1 * 2 * 3 * 4 * 5 * 6 = 720.$$

Факториал нуля принимается за единицу:

$$0! = 1.$$

Рассмотрим числовую последовательность с общим членом $a_n = n!$:

$$1, 2, 6, 24, 120, 720, \dots$$

Эту последовательность можно задать рекуррентной формулой $a_n = a_{n-1} * n$, или $n! = (n-1)! * n$. Если в последней формуле восклицательный знак «сдвигать влево», то получится следующая цепочка равенств:

$$n! = (n-1)! * n = (n-2)! * (n-1) * n = (n-3)! * (n-2) * (n-1) * n \\ \text{и т. д.}$$

Например:

$$6! = 5! * 6 = 4! * 5 * 6 = 3! * 4 * 5 * 6 = 2! * 3 * 4 * 5 * 6 = \\ = 1! * 2 * 3 * 4 * 5 * 6.$$

Двумя восклицательными знаками обозначается произведение только четных или только нечетных чисел. Если мы поставим справа от четного числа два восклицательных знака, то это будет означать произведение четных чисел от 2 до данного числа:

$$(2k)!! = 2 * 4 * 6 * 8 * \dots * (2k).$$

Если же два восклицательных знака стоят справа от нечетного числа, то это означает произведение нечетных чисел:

$$(2k+1)!! = 1 * 3 * 5 * 7 * \dots * (2k+1).$$

Здесь соседние сомножители отличаются друг от друга на 2, поэтому рекуррентные формулы для таких «двойных факториалов» будут выглядеть следующим образом:

$$(2k)!! = (2k - 2)!! * (2k),$$

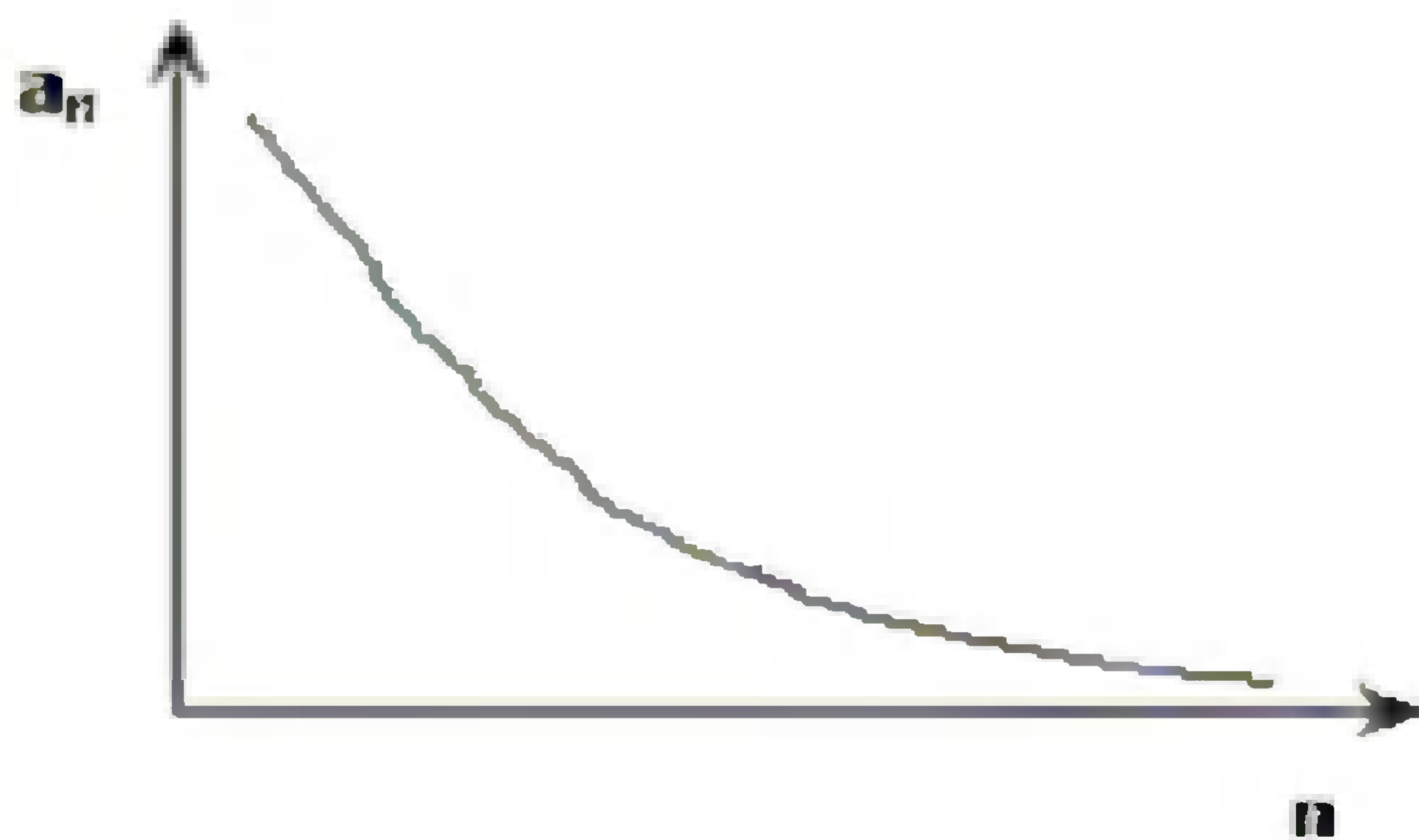
$$(2k + 1)!! = (2k - 1)!! * (2k + 1).$$

Вычисляется факториал при помощи следующей программы:

```
P = 1
INPUT «Введите число»; N
FOR I = 2 TO N
P = P * I
NEXT
PRINT «Факториал числа»; N; «Равен»; P
```

§ 7. Сходящиеся числовые последовательности. Предел числовой последовательности

Рассмотрим числовую последовательность с первым членом $\frac{1}{2}$ и знаменателем $\frac{1}{2}$: $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \dots$. Если нарисовать график зависимости a_n от n , то получится



При увеличении n члены прогрессии уменьшаются, неограниченно приближаясь к нулю, но никогда не достигая его. Слова «неограниченно приближаются» означают, что, какое бы маленькое положительное число ϵ мы ни задумали, всегда найдется такой номер n , начиная с которого все члены последовательности a_n будут меньше этого числа (ϵ — буква греческого алфавита, читается эпсилон, применяется в математике для обозначения очень маленьких величин). Например, возьмем $\epsilon = \frac{1}{1000}$. Тогда

$a_{10} = \frac{1}{1024} < \frac{1}{1000}$, и все члены последовательности с номерами большими 10 будут меньше $\frac{1}{1000}$. Если взять то $\epsilon = \frac{1}{1000000}$, то

$a_{20} = \frac{1}{1048576} < \frac{1}{1000000}$, и все члены последовательности с номерами большими 20 будут меньше $\frac{1}{1000000}$.

Теперь рассмотрим последовательность, образованную суммами двух, трех, четырех и т. д. членов прогрессии a_n . Сумму n членов прогрессии будем обозначать S_n . Если взять в качестве 1-го члена этой последовательности взять «сумму», состоящую из одного члена a_1 , то получим следующую последовательность:

$$S_1 = \frac{1}{2};$$

$$S_2 = \frac{1}{2} + \frac{1}{4} = \frac{3}{4};$$

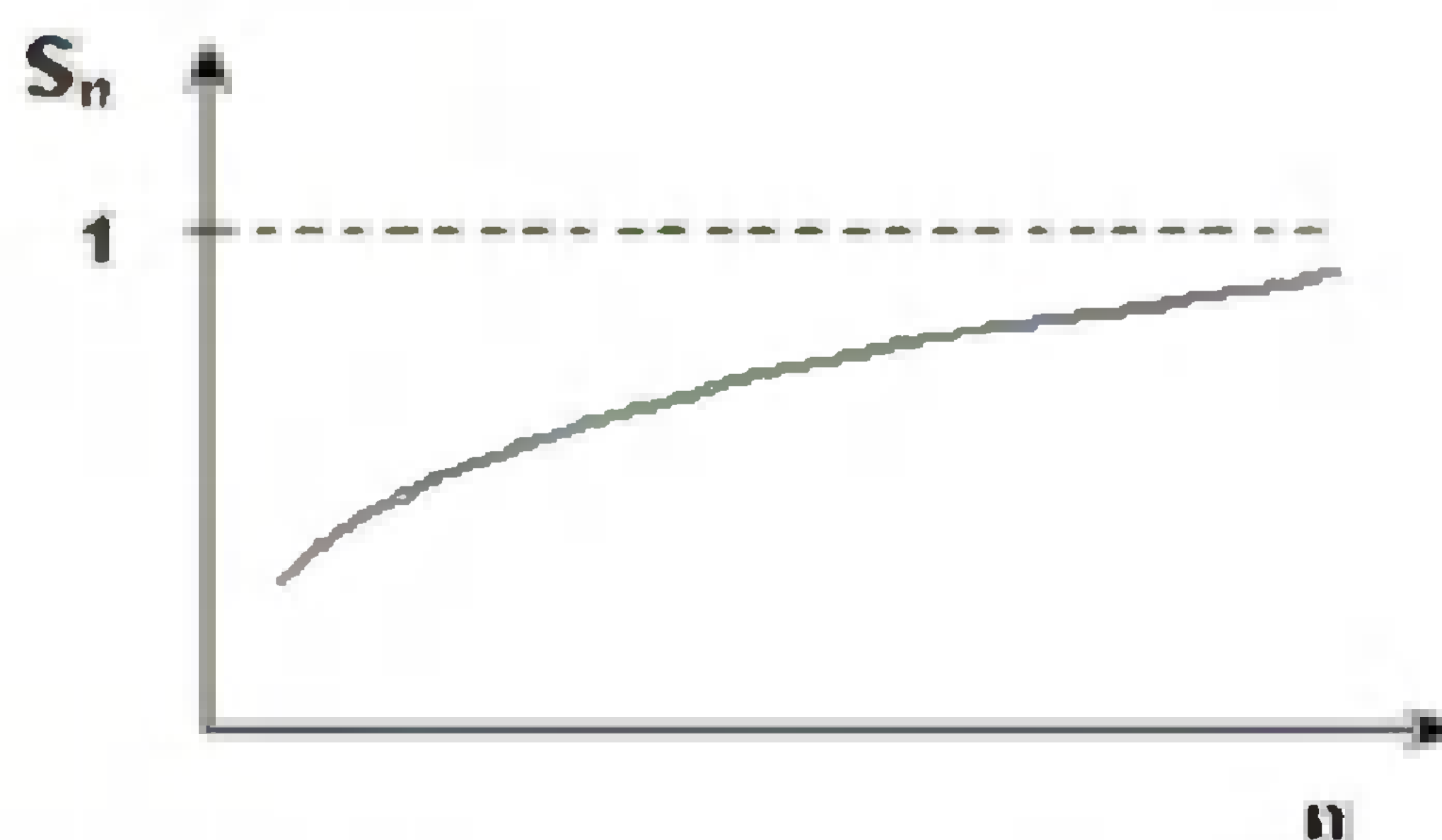
$$S_3 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = \frac{7}{8};$$

$$S_4 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} = \frac{15}{16};$$

Как видно, члены последовательности S_n увеличиваются, оставаясь в то же время меньше 1.

Из курса алгебры известно, что сумму членов бесконечной геометрической прогрессии со знаменателем $|q| < 1$ можно вычислить по формуле: $S = \frac{a_1}{1 - q}$. Для нашей прогрессии эта формула дает значение $S = 1$. Это оз-

начает, что члены последовательности S_n неограниченно приближаются к 1, никогда не достигая ее:



Про последовательность S_n можно сказать, что, какое бы маленькое число ϵ мы ни взяли, всегда найдется номер n такой, начиная с которого все члены последовательности S_n будут отличаться от 1 меньше, чем на ϵ . В этом случае говорят, что последовательность S_n сходится к 1. Последовательность a_n соответственно сходится к 0. Последовательность, которая сходится к какому-либо числу, называется сходящейся числовой последовательностью.

Для числа, к которому сходится числовая последовательность, существует специальное название: это число называется пределом числовой последовательности.

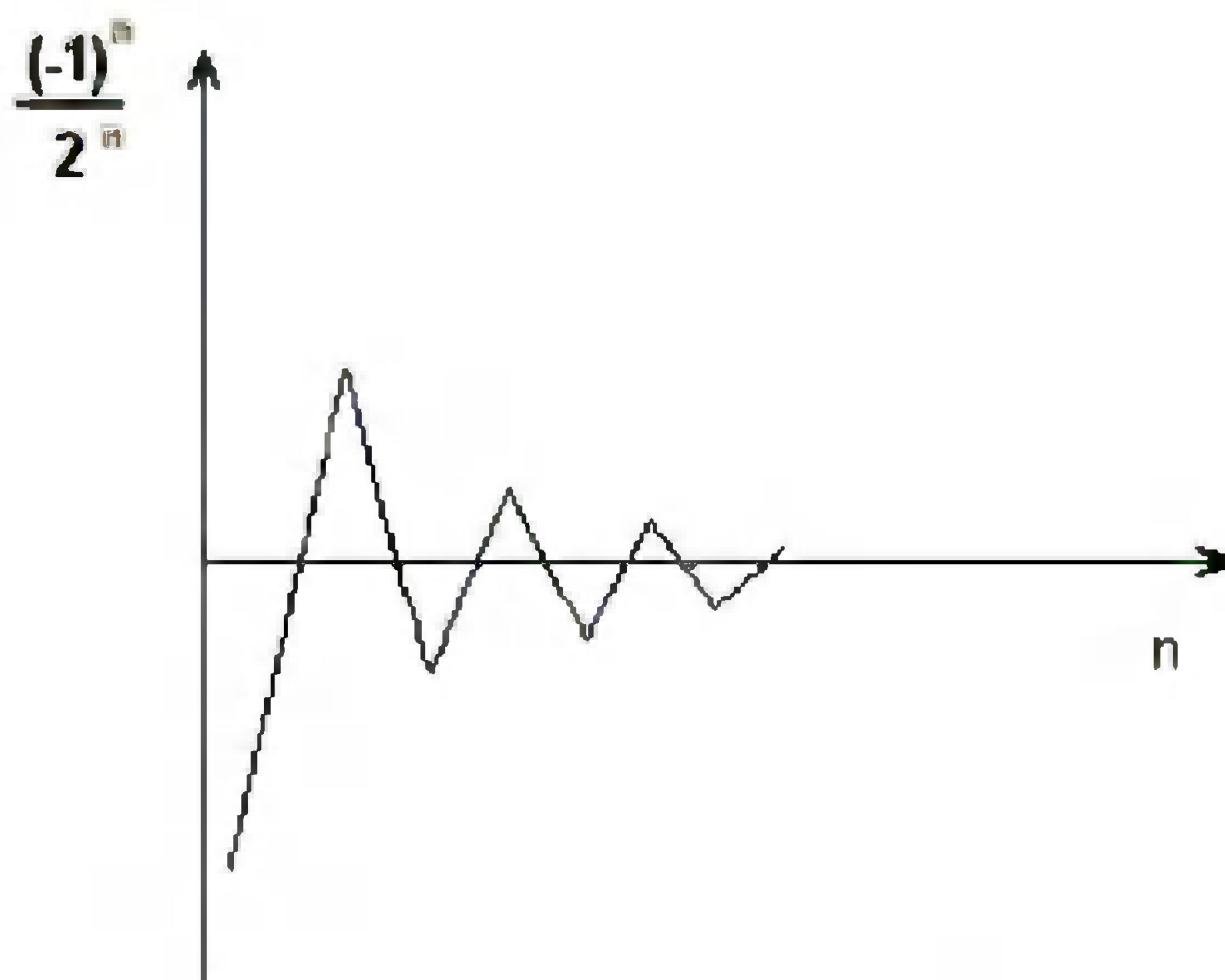
Определение. Число x называется пределом последовательности $a_1 \dots a_n$, если для любого сколь угодно малого ϵ можно указать такое достаточно большое положительное число N , что для всех $n > N$ выполняется неравенство $|a_n - x| < \epsilon$.

Обозначение предела: $\lim_{n \rightarrow \infty} = x$. Это обозначение происходит от латинского слова *limit* (предел).

Обратите внимание, что в определении предела стоит абсолютная величина разности числа x и члена последовательности a_n . Знак абсолютной величины стоит потому, что разность $a_n - x$ может быть и отрицательной. Например, члены последовательности

$$-\frac{1}{2}, +\frac{1}{4}, -\frac{1}{8}, +\frac{1}{16} \dots \frac{(-1)^n}{2^n}, \dots$$

принимают как положительные, так и отрицательные значения. Тем не менее при увеличении n члены последовательности приближаются к нулю, и 0 является пределом этой последовательности:



Таким образом, члены последовательности не обязательно должны приближаться к своему предельному значению с одной стороны. Последовательность может «колебаться» вокруг предела. Важно лишь, чтобы величина $|a_n - x|$ уменьшалась при увеличении n .

В школьном курсе алгебры изучается только один тип последовательностей, имеющих предел: геометрическая прогрессия со знаменателем $|q| < 1$. Приведем еще два примера сходящихся последовательностей:

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4} \dots \frac{1}{n}.$$

Эта последовательность называется гармоническим рядом чисел. Она также сходится к 0. Следующая последовательность сходится к 1:

$$\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \dots, \frac{n}{n+1}$$

Натуральный ряд чисел $1, 2, 3, 4, 5, \dots, n$ не сходится ни к какому числу. Эта последовательность неограниченно возрастает. Про такие последовательности говорят, что они стремятся к бесконечности (имеют бесконечный предел).

Не имеют предела последовательности:

$$\begin{aligned} &-1, 1, -1, 1 \dots (-1)^n, \\ &-2, 4, -8, 16 \dots (-2)^n. \end{aligned}$$

Заметьте, что выражения «последовательность не имеет предела» и «последовательность имеет бесконечный предел» обозначают разные свойства последовательностей.

§ 8. Сумма членов бесконечной числовой последовательности. Числовой ряд

В предыдущем параграфе мы использовали формулу для вычисления суммы членов бесконечной геометрической прогрессии со знаменателем $|q| < 1$. Эта формула выводится в курсе алгебры. Однако реально мы не можем сложить бесконечное количество чисел. Что же понимается под суммой бесконечного количества слагаемых?

В математике для суммы бесконечного количества слагаемых существует специальный термин. Сумма бесконечного числа слагаемых называется рядом. Суммы, образованные одним, двумя, тремя и т. д. слагаемыми, называются частными суммами ряда. Частные суммы ряда обозначаются обычно S_1, S_2, S_3 и т. д. (мы уже использовали эти обозначения в предыдущем параграфе). За сумму бесконечного количества слагаемых принимается предел, к которому стремится последовательность, образованная суммами $S_1, S_2, S_3 \dots S_n$ при неограниченном возрастании n . Записывается это так:

$$S = \lim_{n \rightarrow \infty} S_n.$$

Если последовательность, образованная частными суммами ряда, имеет предел, то ряд называется сходящимся. В § 7 мы рассматривали последовательность, образованную суммами одного, двух, трех и т. д. членов геометрической прогрессии со знаменателем $q = \frac{1}{2}$. Было сказано, что эта последовательность сходится к 1. Следовательно, сумма членов этой прогрессии является сходящимся рядом. Поскольку существует формула для вычисления суммы членов бесконечной геометрической прогрессии со знаменателем $|q| < 1$, то сумма членов любой геометрической прогрессии со знаменателем $|q| < 1$ является сходящимся рядом.

Не всякий ряд является сходящимся. Сумма членов гармонического ряда

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4} \dots \frac{1}{n}$$

не сходится, поскольку последовательность, образованная частными суммами

$$\begin{aligned} S_1 &= 1; \\ S_2 &= 1 + \frac{1}{2} = 1,5; \\ S_3 &= 1 + \frac{1}{2} + \frac{1}{3} = 1,833; \\ &\dots\dots\dots \end{aligned}$$

бесконечно увеличивается (частные суммы гармонического ряда расходятся).

До сих пор мы использовали так называемую развернутую запись операции суммирования членов последовательности, когда все слагаемые выписываются в строчку. Например, запись суммы пяти членов последовательности выглядит так:

$$S = a_1 + a_2 + a_3 + a_4 + a_5.$$

Если количество слагаемых увеличивается, запись удлиняется. Если нам надо записать сумму произвольного, но конечного числа слагаемых, мы обозначаем их количество буквой n и записываем:

$$S = a_1 + a_2 + a_3 + \dots + a_n.$$

Если необходимо записать сумму бесконечного количества слагаемых, то в конце выражения ставится многоточие:

$$S = a_1 + a_2 + a_3 \dots$$

или

$$S = a_1 + a_2 + a_3 + \dots + a_n \dots$$

Такая форма записи неудобна для работы. При работе с последовательностями используют более удобную сокращенную форму записи. При такой записи операцию суммирования обозначают греческой буквой Σ (сигма). Справа от буквы Σ пишут выражение общего члена последовательности, внизу — номер члена последовательности, с которого начинается суммирование, вверху — номер последнего слагаемого. Запись суммы первых пяти членов числовой последовательности в сокращенной записи выглядит так:

$$S = \sum_{i=1}^5 a_i.$$

Сокращенная запись суммы первых n членов последовательности:

$$S = \sum_{i=1}^n a_i.$$

Здесь номер члена последовательности обозначен буквой i , буквой n обозначено количество слагаемых. Обратите внимание, что в этих формулах i — переменная величина (аргумент), а n — какое-то постоянное число.

Запишем в качестве примера сумму первых десяти членов гармонического ряда:

$$S = \sum_{i=1}^{10} \frac{1}{i}.$$

Чтобы получить развернутую запись, нужно в формуле общего члена $\frac{1}{i}$ изменять значение параметра i в тех пределах, которые указаны снизу и сверху значка Σ и соединять полученные числа знаком «+»:

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10}.$$

Суммирование необязательно нужно начинать с первого члена. Мы можем записать выражение для суммы членов последовательности с номера m по номер n (здесь предполагается, что $m < n$):

$$S = \sum_{i=m}^n a_i.$$

Поскольку ряд — сумма бесконечного количества слагаемых, то при записи рядов в качестве верхнего предела суммирования пишут значок ∞ (бесконечность).

Сумма членов геометрической прогрессии $a_i = \frac{1}{2}$, $q = \frac{1}{2}$ запишется как

$$S = \sum_{i=1}^{\infty} \frac{1}{2^i}.$$

Значок Σ удобен тем, что он переводится на язык QBASIC оператором цикла. Если вычисляется сумма конечного числа слагаемых, то в программе пишется оператор цикла с параметром. Нижний предел суммирования дает начальное значение параметра цикла, верхний предел — конечное значение. Для примера напомним программу, вычисляющую сумму:

$$S = \sum_{i=5}^{10} \frac{1}{i} = \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10}.$$

Последовательность задана формулой общего члена $a_i = \frac{1}{i}$. В программе члены последовательности обозначим буквой A , сумму — буквой S . Начальное значение суммы примем $S = 0$.

```
S = 0
FOR I = 5 TO 10
```

```

A=1 / I
S = S + A
NEXT
PRINT S.

```

Если последовательность можно задать рекуррентной формулой, то программа немного изменится. Вычислим сумму:

$$S = \sum_{i=1}^{10} \frac{1}{2^i}.$$

Здесь последовательность $a_i = \frac{1}{2^i}$ — это геометрическая прогрессия $a_1 = \frac{1}{2}$, $q = \frac{1}{2}$. Ее удобно задать рекуррентной формулой $a_i = \frac{a_{i-1}}{2}$. Тогда перед циклом записываем значение первого члена прогрессии, в качестве начального значения суммы берем a_1 , начальное значение параметра цикла будет не 1, а 2, поскольку первое слагаемое мы уже «просуммировали». Получится следующая программа:

```

A = 0.5: S = A
FOR I = 2 TO 10
A = A / 2
S = S + A
NEXT
PRINT S.

```

Разберемся теперь, как при помощи компьютера вычисляются ряды. Ряд — сумма бесконечного количества слагаемых. Сложить бесконечное количество слагаемых мы не сможем даже на компьютере. Поэтому при вычислении рядов берется конечное число слагаемых (т.е. вычисляется частная сумма ряда). Необходимо решить, сколько же слагаемых нужно взять?

Для ответа на этот вопрос необходимо вспомнить, что многие числа мы не можем записать абсолютно точно (например, число π или $\sqrt{2}$). В школе у таких чисел пишут обычно два знака после запятой ($\pi = 3,14$; $\sqrt{2} = 1,41$). В этом случае говорят, что число записано с точностью 0,01. Это означает, что записанное число отличается от истинного значения не более чем на 0,01. Если мы возьмем шесть знаков после запятой ($\pi = 3,141593$; $\sqrt{2} = 1,414214$), то это будет запись с точностью 0,000001; в этом случае записанное число отличается от истинного значения не более чем на 0,000001.

Вспомним теперь, что у сходящегося ряда слагаемые все время уменьшаются по абсолютной величине. Если мы возьмем достаточно много слагаемых, то последние слагаемые будут настолько маленькими, что мы их уже «не заметим», т. е. частные суммы ряда практически не будут отличаться друг от друга. Этот факт и используют при вычислении рядов. Сначала решают, с какой точностью мы хотим вычислить ряд, и затем складывают члены последовательности до тех пор, пока очередные слагаемые не станут меньше заданной точности.

Будем обозначать точность буквой E. Для наших расчетов достаточно взять $E = 0,000001$. Сумма по-прежнему вычисляется при помощи оператора цикла, но поскольку мы не знаем, сколько итераций нужно сделать, то используем не цикл с параметром, а цикл типа «До». Условие выхода из цикла: очередное слагаемое меньше заданной точности. Напишем программу, вычисляющую ряд $S = \sum_{i=1}^{\infty} \frac{1}{2^i}$:

```
A = 0.5: S = A: E = 0.000001
DO
  A = A / 2
  S = S + A
LOOP UNTIL ABS (A) < E
PRINT S.
```

Сравним ее с программой вычисления суммы $S = \sum_{i=1}^{10} \frac{1}{2^i}$, приведенной выше. Как видим, замена верхнего предела интегрирования значком ∞ привела к тому, что цикл с параметром был заменен циклом типа «До» и появилась новая переменная E.

В разобранных нами примерах нижний предел суммирования равнялся 1. При записи рядов часто ставится нижний предел суммирования $i = 0$. В этом случае начальное значение A будет равно числу, которое получится, если в формулу общего члена поставить $i = 0$. Например, при вычислении ряда

$$S = \sum_{i=0}^{\infty} \frac{1}{2^i}$$

начальное значение $A = 1$, и программа примет следующий вид:

```
A = 1: S = A: E = 0.000001
DO
  A = A / 2
  S = S + A
LOOP UNTIL ABS (A) < E
PRINT S.
```

§ 9. Вывод рекуррентной формулы из формулы общего члена числовой последовательности

Числовые последовательности часто задаются формулой общего члена. В частности, в сокращенной записи суммы членов числовой последовательности

$$S = \sum_{i=0}^n \frac{1}{2^i}$$

под знаком суммы стоит формула общего члена. Однако при вычислениях на компьютерах, как уже говорилось, чаще всего применяется рекуррентная формула. Поэтому очень важно уметь из формулы общего члена выводить рекуррентную формулу. Рассмотрим несколько типичных примеров.

Часто встречаются последовательности, которые включают в себя степенные функции (т. е. в формуле стоит степень какого-либо числа). Знакомая нам геометрическая прогрессия — это степенная функция аргумента

$$a_n = a_1 * q^{n-1}.$$

Чтобы из формулы общего члена геометрической прогрессии получить рекуррентную формулу, необходимо вычислить знаменатель q . Процедура вычисления следующая: записываем формулы для n -го и $(n - 1)$ -го членов прогрессии, затем выражение для n -го члена делим на выражение для $(n - 1)$ -го; результатом будет, очевидно, знаменатель прогрессии. Рассмотрим для примера известную нам прогрессию с общим членом

$$a_n = \frac{1}{2^n}.$$

Выражение для $(n - 1)$ -го члена:

$$a_{n-1} = \frac{1}{2^{n-1}}.$$

Для вычисления знаменателя делим a_n на a_{n-1} :

$$q = \frac{a_n}{a_{n-1}} = \frac{\frac{1}{2^n}}{\frac{1}{2^{n-1}}} = \frac{2^{n-1}}{2^n} = \frac{1}{2}.$$

Получаем следующую рекуррентную формулу:

$$a_n = \frac{1}{2} a_{n-1}.$$

Рассмотрим теперь ряд

$$S = \sum_{n=0}^{\infty} \frac{(-1)^n}{2^n}.$$

Этот ряд сходится к числу $\frac{2}{3}$. Под знаком суммы стоит степенная функция. Получим рекуррентную формулу:

$$a_n = \frac{(-1)^n}{2^n}; \quad a_{n-1} = \frac{(-1)^{n-1}}{2^{n-1}}.$$

$$\frac{a_n}{a_{n-1}} = \frac{(-1)^n}{2^n} : \frac{(-1)^{n-1}}{2^{n-1}} = \frac{(-1)^n \cdot 2^{n-1}}{2^n \cdot (-1)^{n-1}} = -1 \cdot \frac{1}{2} = -\frac{1}{2}.$$

$$a_n = -\frac{1}{2}a_{n-1}.$$

Как видно, степень $(-1)^n$ дает знак «минус» в рекуррентной формуле. Теперь можно написать программу, вычисляющую этот ряд. Поскольку нижний предел суммирования равен 0, то начальное значение $A = \frac{(-1)^0}{2^0} = 1$:

```
A = 1: S = A: E = 0.000001
DO
  A = - A / 2
  S = S + A
LOOP UNTIL ABS(A) < E
PRINT S.
```

Во многих числовых последовательностях коэффициент пропорциональности между a_n и a_{n-1} не является постоянным числом, а зависит от n . Такова последовательность

$$a_n = \frac{1}{n!}.$$

Вычислим коэффициент пропорциональности между a_n и a_{n-1} для этой последовательности:

$$\begin{aligned} a_{n-1} &= \frac{1}{(n-1)!}; \\ \frac{a_n}{a_{n-1}} &= \frac{(n-1)!}{n!} = \frac{1}{n}; \\ a_n &= \frac{a_{n-1}}{n}. \end{aligned}$$

Сумма членов этой последовательности, начиная с нулевого

$$\sum_{n=0}^{\infty} \frac{1}{n!},$$

сходится к числу, которое играет большую роль в математике. Обозначается это число буквой e и примерно равно 2,71828... Как и число π , число e является трансцендентным, т. е. не может быть записано в виде рациональной дроби, а представляется бесконечной непериодической десятичной дробью.

Это число можно легко вычислить на компьютере (конечно, с определенной точностью), если использовать полученную нами рекуррентную формулу:

```
A = 1: S = A: E = 0.000001
I = 0
DO
  I = I + 1
```

```

    A = A / I
    S = S + A
LOOP UNTIL ABS (A) < E
PRINT S.

```

Обратите внимание, что ряд начинается со значения $i = 0$. Поскольку $0! = 1$, то начальное значение $a_0 = 1$.

Можно составить программу, в которой каждое слагаемое будет вычисляться непосредственно по формуле общего члена, а не по рекуррентной формуле:

```

E = 0.000001
S = 1: I = 0
DO
I = I + 1
P = 1
FOR J = 1 TO I
P = P * J
NEXT
A = 1 / P
S = S + A
LOOP UNTIL ABS (A) < E
PRINT S.

```

Эта программа будет работать гораздо медленнее предыдущей, поскольку вместо одного оператора присваивания $A = A / I$ у нас появился цикл. Рекуррентная формула позволила нам заменить «медленную» операцию вычисления факториала «более быстрой» — делением. Такой же выигрыш получается при использовании рекуррентной формулы в процессе вычисления суммы членов геометрической прогрессии $a_n = \frac{1}{2^n}$: рекуррентная формула позволяет заменить «медленную» операцию возведения в степень «более быстрой» операцией деления.

Однако не всегда использование рекуррентной формулы ускоряет работу программы. Рассмотрим бесконечную сумму:

$$\sum_{i=1}^{\infty} \frac{1}{i(i+1)}.$$

Этот ряд сходится к 1. Последовательность $a_i = \frac{1}{i(i+1)}$ можно задать при помощи рекуррентной формулы:

$$a_1 = 1, \quad a_i = \frac{i-1}{i+1} a_{i-1}.$$

Но эта рекуррентная формула не даст выигрыша в скорости вычислений, поскольку в ней остались те же самые операции умножения и деления, которые стоят в формуле общего члена.

При работе с числовыми последовательностями формулу общего члена имеет смысл заменять рекуррентной формулой в тех случаях, когда при такой замене вместо «медленных» операций появляются «быстрые». Например, операции умножения и деления выполняются быстрее, чем вычисление факториала и возведение в степень. По сравнению с операциями умножения и деления «более быстрыми» являются операции сложения и вычитания.

Мы уже видели, что при замене формулы общего члена геометрической прогрессии $a_n = a_1 * q^{n-1}$ рекуррентной формулой $a_n = a_{n-1} * q$ исчезает операция возведения в степень. Точно такой же эффект достигается, если формулу общего члена арифметической $a_n = a_1 + (n - 1) * d$ прогрессии заменить рекуррентной формулой $a_n = a_{n-1} + d$: здесь исчезает операция умножения.

Рассмотрим более сложные случаи. Пусть требуется вычислить сумму

$$S = \sum_{i=1}^n \frac{x^i}{y + (i - 1)d},$$

где x, y, d, n — заданные числа.

Здесь в числителе стоит геометрическая прогрессия, а в знаменателе — арифметическая прогрессия. В этом случае следует отдельно вычислять числитель и знаменатель, а затем производить деление:

```
INPUT N, X, Y, D
A = X: B = Y
S = A / B
FOR I = 2 TO N
  A = A * X
  B = B + D
  S = S + A / B
NEXT
PRINT S.
```

Рассмотрим еще одну сумму:

$$S = \sum_{i=0}^n \left(1 + \frac{(-1)^i}{(i + 1)!} \right).$$

Здесь каждый член последовательности разбивается на два слагаемых. Одно слагаемое постоянно (единица), другое представляется в виде рекуррентной формулы $a_1 = 1, a_i = -a_{i-1} / i$. Соответствующая программа:

```
INPUT N
A = 1
S = 1 + A
FOR I = 1 TO N
  A = -A / I
  B = 1 + A
  S = S + B
NEXT
```

PRINT 5

§ 10. «Золотое сечение». Цепные дроби

Разделим отрезок произвольной длины на две неравные части таким об-

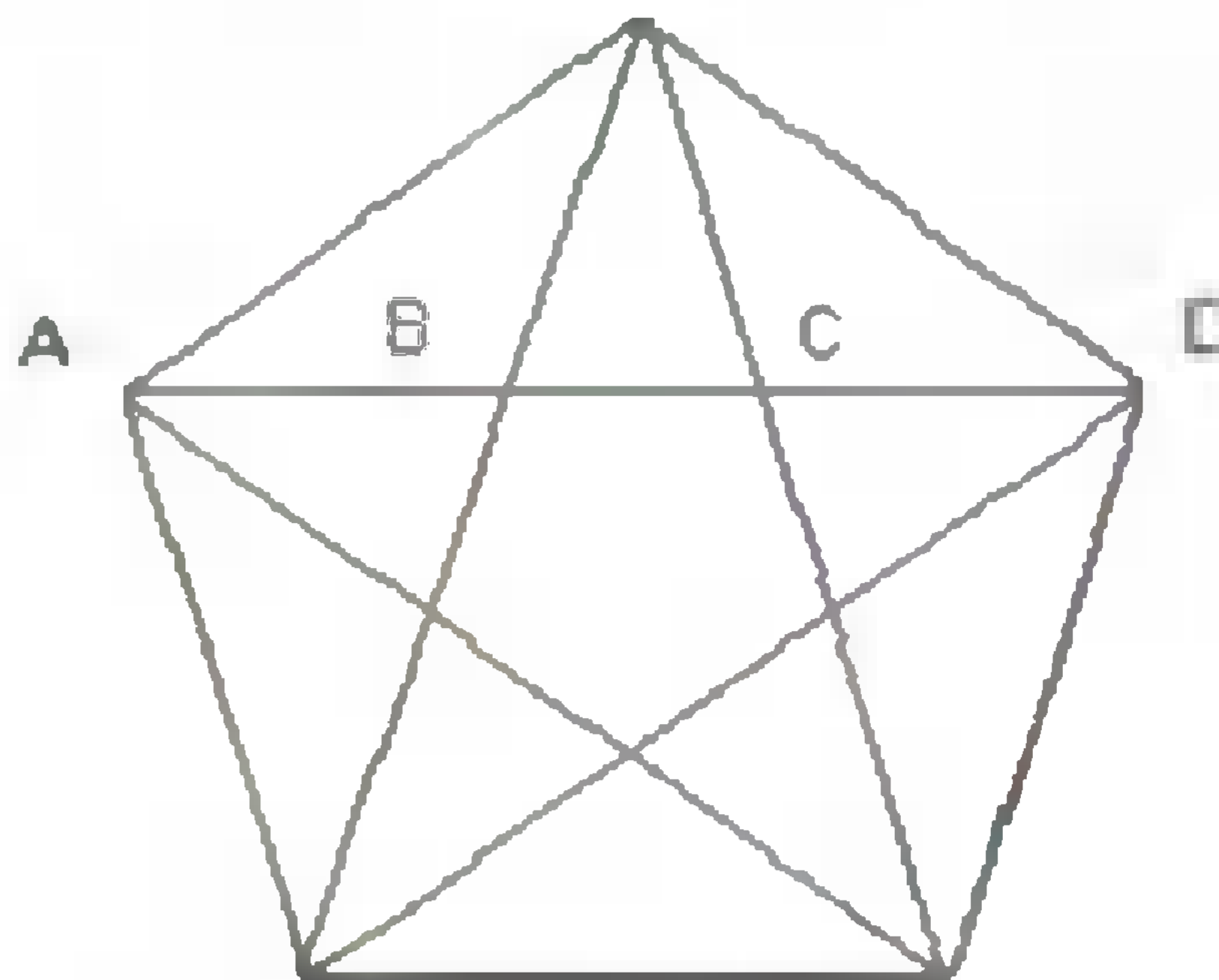


разом, чтобы длина всего отрезка относилась к длине большей части так же, как длина большей части относится к длине меньшей части. Если длину отрезка принять равной 1, а длину большей части обозначить x то получим пропорцию

$$\frac{1}{x} = \frac{x}{1-x}.$$

Отношение $\frac{1}{x}$ называется «золотым сечением». Если решить пропорцию относительно x , то получим $x = 0,618034\dots$, $\frac{1}{x} = 1,6180339\dots$

«Золотое сечение» обладает многими замечательными свойствами. Например, прямоугольник, отношение сторон которого равно x , выглядит наиболее эстетично. Если от такого прямоугольника отрезать квадрат наибольшей площади, то отношение сторон полученного прямоугольника сно-



ва будет равно x .

Если в правильном пятиугольнике провести все диагонали, то получится пятиугольная звезда.

Оказывается, при этом диагонали будут делить друг друга «золотым сечением»:

$$\frac{AD}{AC} = \frac{AC}{CD}; \quad \frac{AD}{BD} = \frac{BD}{AB}.$$

Эта особенность пятиконечной звезды была известна еще пифагорейцам — последователям древнегреческого философа и математика Пифаго-

ра; вероятно, поэтому пифагорейцы избрали пятиконечную звезду своим символом.

Для нас «золотое сечение» интересно тем, что выражается необычной формулой:

$$\frac{1}{x} = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{\dots 1 + \frac{1}{2}}}}$$

Называется это выражение цепной дробью. Цепные дроби часто встречаются в математике, и они эффективно вычисляются при помощи оператора цикла.

Цепные дроби бывают как конечные, так и бесконечные. Выражение для «золотого сечения» — бесконечная цепная дробь. Но мы учтем ограниченное количество дробей в знаменателе (достаточно будет взять 10—15 дробей), чтобы использовать оператор цикла с параметром.

Вычисляются такие «этажерки» снизу (с «хвоста»). Составим последовательность из дробей, двигаясь «снизу вверх» и добавляя на каждом шаге одну дробную черту:

$$\begin{aligned} a_1 &= 2; \\ a_2 &= 1 + \frac{1}{2} = 1 + \frac{1}{a_1}; \\ a_3 &= 1 + \frac{1}{1 + \frac{1}{2}} = 1 + \frac{1}{a_2}; \\ a_4 &= 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{2}}} = 1 + \frac{1}{a_3}; \\ &\dots \end{aligned}$$

В результате получается последовательность, заданная рекуррентной формулой:

$$a_1 = 2, \quad a_n = 1 + \frac{1}{a_{n-1}}.$$

Эта последовательность быстро сходится к числу 1,618... Соответствующая программа

```
A = 2
FOR I = 1 TO 10
  A = 1 + 1 / A
NEXT
```

```
PRINT A.
```

Другое выражение для «золотого сечения»:

$$\frac{1}{x} = \sqrt{1 + \sqrt{1 + \sqrt{1 + \dots + \sqrt{2}}}}$$

Методика вычисления по этой формуле такая же, как и в случае с цепной дробью. Нарастивая радикалы «с хвоста», получаем следующую последовательность:

$$\begin{aligned} a_1 &= \sqrt{2}, \\ a_2 &= \sqrt{1 + \sqrt{2}} = \sqrt{1 + a_1}, \\ a_3 &= \sqrt{1 + \sqrt{1 + \sqrt{2}}} = \sqrt{1 + a_2}, \\ &\dots\dots\dots \\ a_n &= \sqrt{1 + a_{n-1}}. \end{aligned}$$

Соответствующая программа

```
A = SQR(2)
FOR I = 1 TO 10
  A = SQR(1 + A)
NEXT
PRINT A
```

Глава VI

Задачи на проверку условий

§ 1. Логические выражения

До сих пор мы использовали только вычислительные возможности компьютера — умение быстро выполнять арифметические операции над числами. Мы проводили громоздкие вычисления по сложным формулам и в результате получали число. Но не всегда от компьютера требуется числовой результат. Вспомним задачи, которые решаются при помощи условного оператора, например, такую: даны два числа, вывести на экран наибольшее из них. Задача решается при помощи следующей программы:

```
INPUT A, B  
IF A > B THEN PRINT A ELSE PRINT B
```

При решении этой задачи не делается никаких вычислений, а только проверяется условие $A > B$. Условие может выполняться или не выполняться в зависимости от значений переменных A и B . Если $A = 5$, $B = 3$, то мы говорим, что условие выполняется; если $A = 5$, $B = 8$, то условие не выполняется.

В качестве условия у нас фигурирует выражение, в котором сравниваются два числа. Что же получается в результате? Если мы над числами совершаем арифметическую операцию, то в качестве результата получаем число ($5 + 3 = 8$; $8 * 5 = 40$). Если мы сравниваем два числа, то получаем некоторый логический вывод: «верно» — «неверно» ($5 > 3 = \text{«верно»}$; $5 > 8 = \text{«неверно»}$).

Выражения, которые включали в себя арифметические операции над числами и в результате давали число, мы называем арифметическими выражениями (гл. II, § 3). Выражения, которые своим результатом имеют логический вывод, называются логическими выражениями. В программировании логическими выражениями являются «условия», которые ставятся в условном операторе после слова IF. Здесь они получаются в результате применения к числам операций отношения (гл. III, § 1).

Термин «логическое выражение» пришел в математику из так называемой формальной логики, которая появилась в античном мире и впервые была систематически изложена в трудах Аристотеля (384—322 до н. э.). Формальная, или Аристотелева логика занимается изучением законов че-

ловеческого мышления. Основным объектом, с которым работает формальная логика, является логическое выражение — повествовательное предложение, о котором можно сказать, истинно оно или ложно. Например:

1. Волга впадает в Каспийское море (истинно).
2. 5 — положительное число (истинно).
3. 8 — отрицательное число (ложно).
4. Земля не имеет естественных спутников (ложно).

Не являются логическими выражениями следующие предложения:

1. Который час?
2. Пейте томатный сок!
3. Как прекрасен этот мир!

Логические выражения называют также высказываниями.

Дальнейшим развитием формальной логики явилась математическая логика, основы которой заложил английский математик Джордж Буль (1815—1864) в своей работе «Исследование законов мысли» (1854). В математической логике также рассматриваются логические выражения (высказывания), но здесь высказывания обязательно являются предложениями обычного «человеческого» языка. В математической логике высказывания — это величины, которые могут принимать только два значения. Эти значения обозначают словами «истина» — «ложь» или символами «1» — «0», «И» — «Л», TRUE — «FALSE» и т. д. При этом символы «1» и «0» означают не числа 1 и 0, с которыми мы имеем дело в обычной школьной математике, это просто условные обозначения значений логических величин. Слова «истина» и «ложь» также являются условным обозначением значений логических величин, а не истиной или ложью в обычном смысле.

§ 2. Логические переменные

Логические величины — это новый для нас тип математических величин, который не встречается в «обычной» школьной математике. В школе мы привыкли иметь дело с числами и компьютер до сих пор использовал для операций над числами. Однако в современной математике широко используются логические величины и операции над ними. Поэтому в современных языках программирования предусмотрены средства работы с логическими величинами.

Логическую величину можно записать в память, как и число. Любая величина (логическая или числовая) во внутреннюю память компьютера записывается при помощи переменной. На языке QBASIC можно записать выражение

$X = 5 > 3.$

Тогда в памяти появится логическая переменная

X	истина
---	--------

Значением этой переменной будет не число, а логическая величина «истина». Если в программе написать

$$A = 5 < 0,$$

то в памяти появится другая логическая переменная

A	ложь
---	------

В языке QBASIC логическое значение «истина» кодируется числом «-1», а логическое значение «ложь» кодируется числом «0». Если мы введем в память приведенные выше логические переменные, а затем напишем

```
PRINT X  
PRINT A,
```

то на экране появятся числа

```
-1  
0
```

Логические переменные широко используются в программировании наряду с привычными нам числовыми переменными. Наиболее простой пример — использование в условном операторе вместо конкретного логического выражения его идентификатора (имени). В программе из предыдущего параграфа можно записать следующим образом:

```
INPUT A, B  
X = A > B  
IF X THEN PRINT A ELSE PRINT B
```

Теперь перевод условного оператора на русский язык будет звучать, как: «Если переменная X имеет значение «истина», вывести на экран значение переменной A, в противном случае вывести на экран значение переменной B».

В дальнейшем мы будем решать задачи, в которых будет необходимо проверять истинность сложных логических выражений. Эти логические выражения будут записываться в виде длинных формул, которые просто неудобно ставить в условный оператор. Здесь нам помогут логические переменные. Мы будем обозначать полученные формулы идентификаторами и затем использовать эти идентификаторы в качестве «условий» в условных операторах.

Например, решая задачу о принадлежности точки с координатами (x, y) 1-й четверти (гл. III, § 4), мы можем составное логическое выражение « $X > 0$ AND $Y > 0$ » обозначить каким-нибудь идентификатором и поставить этот идентификатор в условный оператор:

```
CLS
INPUT «Введите координаты точки»; x, y
A = X > 0 AND Y > 0
IF A THEN PRINT «Точка лежит в 1-й четверти» ELSE PRINT
«Точка не лежит в 1-й четверти»
```

§ 3. Логические операции

Мы уже познакомились с некоторыми логическими операциями при изучении условного оператора (гл. III, § 4). Логической операцией мы называли процесс объединения нескольких простых условий в одно составное. Точно так же теперь мы будем понимать под логической операцией процедуру построения нового логического выражения из уже имеющихся. Логическое выражение, получившееся из нескольких более простых логических выражений, будем называть составным логическим выражением. Главный признак составного логического выражения — наличие знаков логических операций в формулах. Логические выражения, не содержащие знаки логических операций, будем называть простыми, или элементарными логическими выражениями.

Определение. Логической операцией называется процедура построения нового логического выражения из уже имеющихся.

Определение. Логическое выражение, не содержащее знаков логических операций, называется простым (элементарным).

Определение. Логическое выражение, содержащее знаки логических операций, называется составным.

В математической логике значения логических операций задаются при помощи так называемых таблиц истинности. В этих таблицах записывают в столбик значения исходных логических выражений, а в крайнем правом столбце записывают значение составного логического выражения, т. е. результат логической операции. Напишем таблицы истинности известных нам логических операций.

Логическое умножение (операция «И», конъюнкция) обозначается значком « $\wedge$ ». Таблица истинности:

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

Эту же таблицу можно записать в другом виде:

$$0 \wedge 0 = 0$$

$$0 \wedge 1 = 0$$

$$1 \wedge 0 = 0$$

$$1 \wedge 1 = 1$$

В этих формулах значок «1» обозначает логическое значение «истина», а значок «0» — логическое значение «ложь». Как видим, результат логического умножения двух логических выражений принимает значение «истина» только в том случае, если оба логических выражения имеют значения «истина».

Логическое сложение (операция «ИЛИ», дизъюнкция) обозначается значком « $\vee$ ». Таблица истинности:

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

Другой вид этой таблицы:

$$0 \vee 0 = 0$$

$$0 \vee 1 = 1$$

$$1 \vee 0 = 1$$

$$1 \vee 1 = 1$$

Результат логического сложения двух логических выражений принимает значение «истина» в том случае, если хотя бы одно из логических выражений имеет значение «истина».

При помощи конъюнкции и дизъюнкции новое логическое выражение получается из двух «старых». Поэтому эти операции называются бинарными (т. е. операциями над двумя объектами). В математической логике известна так же операция над одним логическим выражением (унарная операция). Эта операция изменяет значение логического выражения на противоположное. Называется она логическим отрицанием (операция «НЕ», инверсия). Обозначается логическое отрицание черточкой над именем пе-

ременной ($\bar{a}$) или значком « $\neg$ » ($\neg a$). В QBASIC логическое отрицание обозначается словом NOT.

Таблица истинности логического отрицания:

a	$\bar{a}$
0	1
1	0

§ 4. Перевод логических выражений с русского языка на язык математики

Наличие логических переменных и логических операций в языках программирования позволяет решать на компьютере задачи, в которых требуется проверить выполнение некоторого условия. Решение таких задач сводится к вычислению некоторого логического выражения, более или менее сложного. Естественно, в условии задачи логическое выражение записывается на обычном человеческом языке. Компьютер человеческого языка не понимает. Чтобы компьютер помог нам решить задачу, мы должны сформулировать условие задачи на языке математических формул. Такой перевод условия задачи с обычного человеческого языка на язык математических формул называется формализацией задачи.

Процесс формализации задачи у нас будет состоять из следующих этапов:

- выделение из текста задачи «условия», которое требуется проверить. Это «условие» будет представлять собой некоторое логическое выражение;
- разбиение полученного логического выражения на простые (элементарные);
- запись элементарных логических выражений при помощи математических формул;
- объединение формализованных логических выражений в одно составное при помощи логических операций.

Рассмотрим процесс формализации задачи на конкретных примерах.

Задача 1. Дан круг радиуса R с центром в начале координат и точка с координатами (x, y) . Определить, лежит ли точка внутри круга.

Здесь требуется вычислить значение логического выражения «Точка с координатами (x, y) лежит внутри круга радиуса R с центром в начале координат». Чтобы формализовать это выражение, его нужно переписать, используя математическую терминологию: «Расстояние от точки до начала координат меньше радиуса круга». Теперь это логическое выражение можно записать в виде математической формулы: $\sqrt{x^2 + y^2} < R$. Эта формула является «условием», которое можно поставить в условный оператор. Но мы в условный оператор поставим идентификатор логической переменной

$L = \sqrt{x^2 + y^2} < R$. Тогда программа, решающая нашу задачу, будет выглядеть следующим образом:

```
CLS
INPUT R, X, Y
L = SQR(X^2 + Y^2) < R
IF L THEN PRINT «ТОЧКА ЛЕЖИТ ВНУТРИ КРУГА» ELSE PRINT «НЕТ»
```

Решение задачи сводится к цепочке преобразований логического выражения: «Точка с координатами (x, y) лежит внутри круга радиуса R с центром в начале координат» $\rightarrow$ «Расстояние от точки до начала координат меньше радиуса круга» $\rightarrow \sqrt{x^2 + y^2} < R \rightarrow L = \sqrt{x^2 + y^2} < R$. Это логическое выражение является элементарным, т. е. не содержащим логических связок. Рассмотрим задачу, в которой вычисляется значение составного логического выражения.

Задача 2. Даны три числа. Выяснить, существует ли треугольник с такими сторонами.

Решение этой задачи сводится к проверке условия существования треугольника: «Сумма двух любых сторон больше третьей». Если мы обозначим стороны треугольника буквами a, b, c , то это условие можно сформулировать следующим образом: «Одновременно выполняются три неравенства: $a + b > c$, $a + c > b$, $b + c > a$ ». Очевидно, каждое неравенство представляет собой элементарное логическое выражение, а условие существования треугольника со сторонами a, b, c — это составное логическое выражение $(a + b > c)$ и $(a + c > b)$ и $(b + c > a)$. Если мы обозначим элементарные логические выражения идентификаторами $L1 = a + b > c$, $L2 = a + c > b$, $L3 = b + c > a$, то получим логическую формулу $L1 \wedge L2 \wedge L3$, которая соответствует фразе «Треугольник со сторонами a, b, c существует». Программа, решающая данную задачу:

```
CLS
INPUT A, B, C
L1 = A + B > C
L2 = A + C > B
L3 = B + C > A
L = L1 AND L2 AND L3
IF L THEN PRINT «ТРЕУГОЛЬНИК СО СТОРОНАМИ»; A; B; C;
«СУЩЕСТВУЕТ»
ELSE PRINT «ТРЕУГОЛЬНИК СО СТОРОНАМИ»; A; B; C; «НЕ
СУЩЕСТВУЕТ»
```

Задача 3. Определить, попадает ли точка с координатами (x, y) в заштрихованную область. Радиус круга R считать известным.

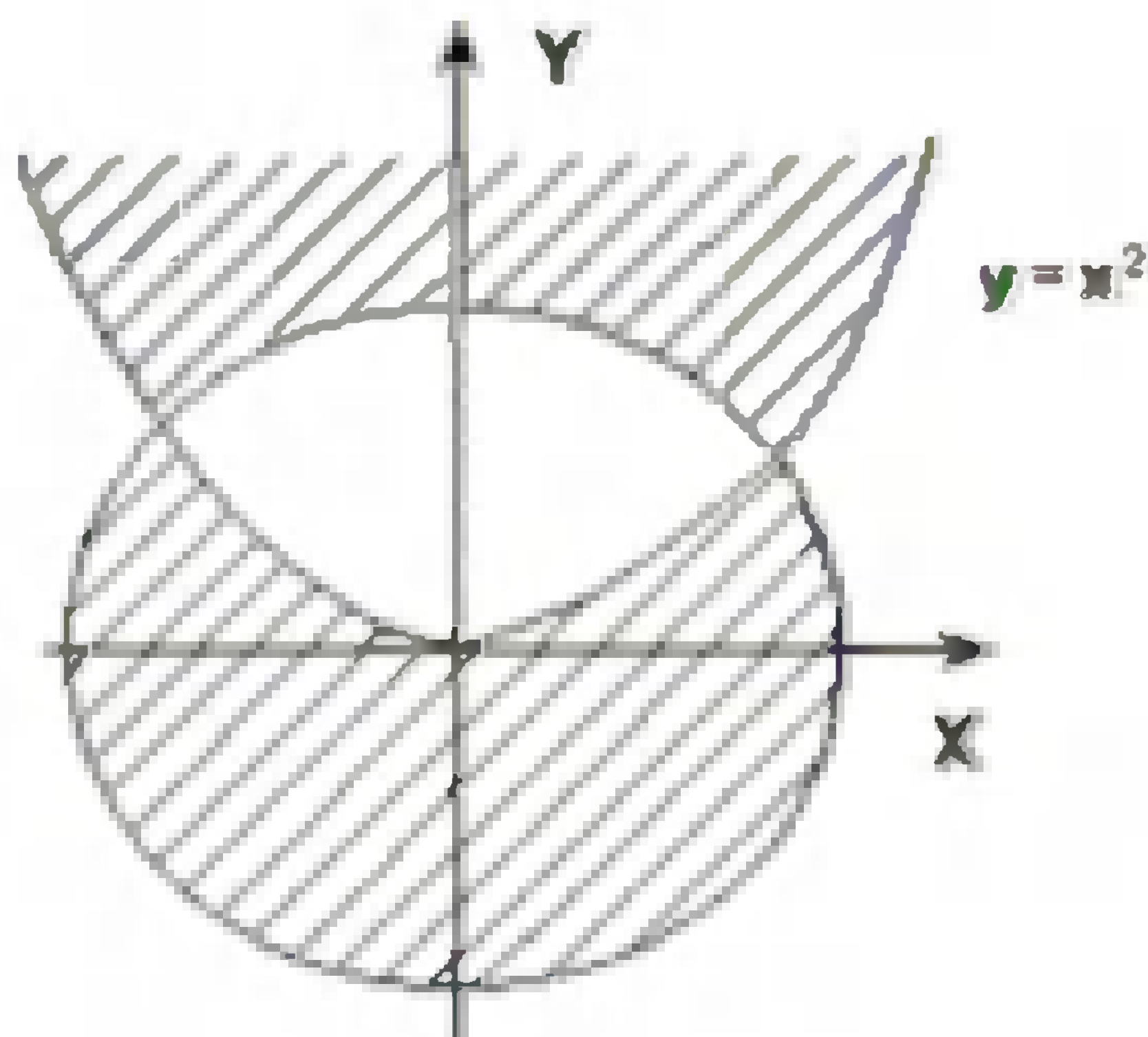
Точка попадает в заштрихованную область, если она лежит внутри круга и ниже параболы или вне круга и выше параболы. Формализуем элементарные логические выражения:

точка лежит внутри круга $= \sqrt{x^2 + y^2} \leq R$;

точка лежит вне круга $= \sqrt{x^2 + y^2} > R$ (можно взять отрицание предыдущего выражения);

точка лежит выше параболы $= y > x^2$;

точка лежит ниже параболы $= y \leq x^2$ (или отрицание предыдущего выражения).



Теперь можно составить программу:

```
CLS
INPUT X, Y, R
L1 = SQR(X^2 + Y^2) <= R
L2 = NOT L1
L3 = Y > X^2
L4 = NOT L3
L = (L1 AND L4) OR (L2 AND L3)
IF L THEN PRINT «ТОЧКА ПОПАДАЕТ В ЗАШТРИХОВАННУЮ
ОБЛАСТЬ»
ELSE PRINT «ТОЧКА НЕ ПОПАДАЕТ В ЗАШТРИХОВАННУЮ ОБЛАСТЬ»
```

§ 5. Составные высказывания и логические формулы

Как вы уже, наверное, заметили, при решении многих задач составные логические выражения формулируются таким образом, что логические связки явно не проявляются. Например, в задаче 2 из предыдущего параграфа условие существования треугольника в конце концов сводится к следующему высказыванию: «Три логических выражения истинны одновременно». При решении таких задач необходимо уметь переводить такие расплывчатые выражения на язык логических формул.

Прежде всего следует определить, что такое логическая формула. Логическая формула — это конструкция, состоящая из идентификаторов, знаков логических операций и скобок. В простейшем случае формула может состоять из единственного идентификатора какого-либо логического выражения. В математике для идентификаторов логических выражений используют обычно начальные буквы латинского алфавита: $a, b, c \dots$. Поскольку мы при написании программ эти буквы используем для обозначения числовых переменных, то во избежание путаницы логические переменные будем обозначать буквой L .

Итак, если L — идентификатор логического выражения, то запись L уже является формулой. Эта формула означает, что логическое выражение истинно. Соответственно $\bar{L}$ на русский язык можно перевести как «Логическое выражение L ложно».

Рассмотрим теперь составные логические выражения. Если два логических выражения $L1$ и $L2$ соединить знаком логического умножения (дизъюнкции), то новое логическое выражение примет значение «истина» в том случае, если элементарные логические выражения $L1$ и $L2$ истинны одновременно. Можно сказать, что формула $L1 \wedge L2$ является переводом на язык математики предложения «два логических выражения $L1$ и $L2$ истинны одновременно».

Составное логическое выражение $L1 \vee L2$ принимает значение «истина», если хотя бы одно из элементарных логических выражений истинно. Отсюда получаем: фраза «хотя бы одно из логических выражений $L1$ и $L2$ истинно» переводится на язык математической логики формулой $L1 \vee L2$.

Из двух элементарных логических выражений можно сконструировать еще два интересных для нас высказывания: «два логических выражения имеют одинаковую истинность» и «два логических выражения имеют противоположную истинность». Если два логических выражения имеют одинаковую истинность, то они или истинны одновременно, или ложны одновременно. Получаем следующую цепочку преобразований:

два логических выражения имеют одинаковую истинность $\equiv$
(первое истинно и второе истинно) или (первое ложно и второе ложно) $=$
 $(L1 \wedge L2) \vee (\bar{L1} \wedge \bar{L2})$.

Если два логических выражения имеют противоположную истинность, то или первое истинно и одновременно второе ложно, или первое ложно и одновременно второе истинно. Получаем:

два логических выражения имеют противоположную истинность $\equiv$
(первое истинно и второе ложно) или (первое ложно и второе истинно) $=$
 $(L1 \wedge \bar{L2}) \vee (\bar{L1} \wedge L2)$.

Рассмотрим теперь три логических выражения $L1$, $L2$, $L3$. Без труда можно перевести на язык формул следующие выражения:

все три логических выражения истинны одновременно $= L1 \wedge L2 \wedge L3$;

все три ложны одновременно $= \bar{L1} \wedge \bar{L2} \wedge \bar{L3}$;

хотя бы одно из них истинно $= L1 \vee L2 \vee L3$;

хотя бы одно из них ложно $= \bar{L1} \vee \bar{L2} \vee \bar{L3}$.

В качестве упражнения формализуйте следующие утверждения:

- одно из трех логических выражений истинно, два ложны;
- одно из трех логических выражений ложно, два истинны;
- три логических выражения имеют одинаковую истинность.

§ 6. Еще две логические операции

Вспомним определение логической операции: логическая операция — это процесс построения нового логического выражения из уже имеющихся. С этой точки зрения все составные логические выражения, рассмотренные в предыдущем параграфе, являются операциями. До сих пор мы упо-

минали только две логические операции — дизъюнкцию и конъюнкцию. Рассмотрим еще две операции над двумя логическими переменными.

Составному высказыванию «два логических выражения имеют одинаковую истинность» соответствует логическая операция «эквиваленция». Эквиваленция — это такая операция над двумя логическими величинами, результат которой принимает значение «истина», если истинности обоих логических выражений совпадают (т. е. оба они или истинны, или ложны). Обозначается эквиваленция значком « $\Leftrightarrow$ ».

Таблица истинности:

a	b	$a \Leftrightarrow b$
0	0	1
0	1	0
1	0	0
1	1	1

Составному высказыванию «два логических выражения имеют противоположную истинность» соответствует логическая операция «исключающее или». Исключающее или — это такая операция над двумя логическими величинами, результат которой принимает значение «истина», если истинности обоих логических выражений противоположны. Обозначается эта операция значком « $\oplus$ ».

Таблица истинности:

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Рассмотрим конкретный пример применения этих операций. Пусть дана прямая $y = k \cdot x + b$. Пусть заданы две точки (x_1, y_1) и (x_2, y_2) . Формализуем утверждение «Точки лежат по одну сторону прямой».

Выражение «Точка с координатами (x_1, y_1) лежит выше прямой $y = k \cdot x + b$ » на языке математических формул запишется как $y_1 > k \cdot x_1 + b$. Аналогично выражение «Точка с координатами (x_2, y_2) лежит выше прямой $y = k \cdot x + b$ » запишется как $y_2 > k \cdot x_2 + b$. Если обе точки лежат по одну сторону прямой, то эти два логических выражения или истинны одновременно, или ложны одновременно (т. е. обе точки лежат или выше прямой, или ниже прямой). Обозначив полученные логические выражения идентификаторами: $L_1 = y_1 > k \cdot x_1 + b$; $L_2 = y_2 > k \cdot x_2 + b$, получим логическую формулу, соответствующую выражению «Точки лежат по одну сторону прямой»: $L_1 \Leftrightarrow L_2$.

Если точки лежат по разные стороны прямой, то из двух логических выражений L_1 и L_2 одно истинно, другое ложно. Отсюда получаем формализацию логического выражения «Точки с координатами (x_1, y_1) и (x_2, y_2) лежат по разные стороны прямой $y = k \cdot x + b$ » = $L_1 \oplus L_2$.

Эти два выражения используются при решении многих задач.

В математической логике вводятся и другие операции над двумя логическими величинами, которые мы рассматривать не будем. Отметим только, что операции инверсия, дизъюнкция и конъюнкция называются булевыми операциями (по имени основателя математической логики Дж. Буля). Особого названия эти операции удостоились по той причине, что через них можно выразить все другие логические операции. Например, для операций «эквиваленция» и «исключающее или» можно записать:

$$a \Leftrightarrow b = (a \wedge b) \vee (\bar{a} \wedge \bar{b})$$

$$a \oplus b = (a \wedge \bar{b}) \vee (\bar{a} \wedge b)$$

В языке QBASIC операция эквиваленция обозначается EQV, а «исключающее или» — XOR.

§ 7. Решение логических задач методом составления характеристического уравнения

В задачах, которые мы решали до сих пор, требовалось вычислить значение составного логического выражения, если известны значения простых логических выражений. Теперь мы рассмотрим задачи, в которых требуется найти такие значения простых высказываний, при которых составное логическое выражение примет значение «истина». Другими словами, мы будем решать логические уравнения.

Рассмотрим следующую логическую задачу. Алеша, Боря и Гриша нашли в земле сосуд. Рассматривая удивительную находку, каждый высказал по два предположения:

А л е ш а. Это сосуд греческий и изготовлен в V веке.

Б о р и с. Это сосуд финикийский и изготовлен в III веке.

Г р и ш а. Это сосуд не греческий и изготовлен в IV веке.

Учитель истории сказал ребятам, что каждый из них прав только в одном из двух предположений. Где и в каком веке изготовлен сосуд?

Очевидно, высказывание каждого из мальчиков — это логическое умножение двух простых высказываний. Поскольку в каждом случае одно из высказываний истинно, а другое ложно, то все три высказывания ложны. Однако если вместо логического умножения над этими простыми высказываниями произвести операцию «исключающее или», то высказывание каждого мальчика примет значение «истина». Введем обозначения:

LG — сосуд греческий;

LF — сосуд финикийский;

L3 — сосуд изготовлен в III в.;

L4 — сосуд изготовлен в IV в.;

L5 — сосуд изготовлен в V в.

Тогда на основании высказывания Алеши получим уравнение

$$LG \oplus L5 = 1.$$

На основании высказываний Бориса и Гриши получаются еще два уравнения:

$$LF \oplus L3 = 1.$$

$$\overline{LG} \oplus L4 = 1.$$

У нас получились три уравнения для пяти неизвестных. Чтобы решить систему, необходимы еще два уравнения. Эти уравнения можно получить из следующих соображений. Во-первых, сосуд изготовлен или в III, или в IV, или в V в., т. е. из трех логических выражений L3, L4, L5 одно истинно, два ложны:

$$(L3 \wedge \overline{L4} \wedge \overline{L5}) \vee (\overline{L3} \wedge L4 \wedge \overline{L5}) \vee (\overline{L3} \wedge \overline{L4} \wedge L5) = 1.$$

Во-вторых, сосуд изготовлен в одной из двух стран — или в Греции, или в Финикии:

$$LG \oplus LF = 1.$$

Если мы теперь объединим полученные высказывания при помощи операции логического умножения, то новое составное высказывание будет иметь, очевидно, значение «истина»:

$$(LG \oplus L5) \wedge (LF \oplus L3) \wedge (\overline{LG} \oplus L4) \wedge (LG \oplus LF) \wedge \\ ((L3 \wedge \overline{L4} \wedge \overline{L5}) \vee (\overline{L3} \wedge L4 \wedge \overline{L5}) \vee (\overline{L3} \wedge \overline{L4} \wedge L5)) = 1.$$

Это уравнение называется характеристическим уравнением задачи. Его решение мы можем поручить компьютеру. В самом деле, компьютеру ничего не стоит перебрать все возможные комбинации значений пяти логических переменных, входящих в уравнение, и выбрать такой вариант, при котором наше уравнение превращается в тождество. Программа, решающая эту задачу, может выглядеть следующим образом:

```
CLS
```

```
FOR LG = -1 TO 0
```

```
FOR LF = -1 TO 0
FOR L3 = -1 TO 0
FOR L4 = -1 TO 0
FOR L5 = -1 TO 0
LA = LG XOR L5
LB = LF XOR L3
LC = NOT LG XOR L4
LD = (L3 AND NOT L4 AND NOT L5) OR (NOT L3 AND L4 AND
NOT L5) OR
(NOT L3 AND NOT L4 AND L5)
LE = LG XOR LF
L = LA AND LB AND LC AND LD AND LE
IF L THEN
IF LG THEN PRINT «сосуд изготовлен в Греции»
IF LF THEN PRINT «сосуд изготовлен в Финикии»
IF L3 THEN PRINT «в III в.»
IF L4 THEN PRINT «в IV в.»
IF L5 THEN PRINT «в V в.»
END IF
NEXT: NEXT: NEXT: NEXT: NEXT
```

§ 8. Импликация

Решим такую задачу.

Семья состоит из пяти человек: Алексея, Веры, Глеба, Даши и Евгения. Когда семья смотрит телевизор, то соблюдаются следующие условия:

1. Смотрят либо Даша, либо Евгений, либо оба вместе.
2. Смотрят либо Вера, либо Глеб, но не вместе.
3. Даша и Глеб либо смотрят вместе, либо оба не смотрят.
4. Если телевизор смотрит Алексей, то смотрит и Вера.
5. Если смотрит Евгений, то смотрят Алексей и Даша.

Кто смотрит телевизор?

С первыми тремя условиями все ясно. Условие 1 — это логическое сложение двух выражений «Телевизор смотрит Даша» и «Телевизор смотрит Евгений». Условие 2 — это «исключающее или» выражений «Телевизор смотрит Вера» и «Телевизор смотрит Евгений». Условие 3 — это эквиваленция двух выражений «Телевизор смотрит Глеб» и «Телевизор смотрит Даша». Условие 4 состоит из двух простых высказываний: телевизор смотрит Алексей, и телевизор смотрит Вера. Эти высказывания соединены логической связкой «Если... то». Эту связку иногда пишут по другому: из первого высказывания следует второе. Первое высказывание называют посылкой, второе — следствием.

Логической связке «Если... то» в математической логике соответствует логическая операция «импликация». Обозначается импликация значком $\Rightarrow$. Чтобы вывести таблицу истинности импликации, посмотрим, какие ситуации могут возникнуть у телевизора при выполнении четвертого условия. Как и прежде, логическое значение «истина» будем обозначать как «1», а логическое значение «ложь» — «0».

Если Алексей пришел смотреть телевизор, то условие 4 будет выполняться (иметь значение «истина»), если и Вера пришла смотреть телевизор, т. е. импликация двух истинных логических выражений есть «истина»:

$$1 \Rightarrow 1 = 1.$$

Если Алексей пришел смотреть телевизор, а Вера не пришла, то условие 4 не выполняется, т. е.

$$1 \Rightarrow 0 = 0.$$

Но если Алексей не пришел смотреть телевизор, то Вера может приходить, а может не приходить, — в условии 4 ничего не сказано, что должна делать Вера в случае отсутствия Алексея. Поэтому ситуации «Алексей не пришел, Вера пришла» и «Алексей не пришел, Вера не пришла» дают значение «истина» для условия 4:

$$0 \Rightarrow 1 = 1$$

$$0 \Rightarrow 0 = 1$$

В результате получаем следующую таблицу истинности импликации:

a	b	$a \Rightarrow b$
0	0	1
0	1	1
1	0	0
1	1	1

На языке QBASIC импликация обозначается как IMP.

Теперь можно составить программу, решающую эту задачу. Введем следующие обозначения:

LD — телевизор смотрит Даша;

LE — телевизор смотрит Евгений;

LV — телевизор смотрит Вера;

LG — телевизор смотрит Глеб;

LA — телевизор смотрит Алексей.

Тогда формализованные условия будут выглядеть следующим образом:

1. $LD \vee LE$;

2. $LV \oplus LG$;

3. $LD \Leftrightarrow LG$;
4. $LA \Rightarrow LV$;
5. $LE \Rightarrow (LA \wedge LD)$.

Программа:

```
FOR LD = -1 TO 0
FOR LE = -1 TO 0
FOR LV = -1 TO 0
FOR LG = -1 TO 0
FOR LA = -1 TO 0
L1 = LD OR LE
L2 = LV XOR LG
L3 = LD EQV LG
L4 = LA IMP LV
L5 = LE IMP (LA AND LD)
L = L1 AND L2 AND L3 AND L4 AND L5
IF L THEN
IF LD THEN PRINT «Телевизор смотрит Даша»
IF LE THEN PRINT «Телевизор смотрит Евгений»
IF LV THEN PRINT «Телевизор смотрит Вера»
IF LG THEN PRINT «Телевизор смотрит Глеб»
IF LA THEN PRINT «Телевизор смотрит Алексей»
END IF
NEXT: NEXT: NEXT: NEXT: NEXT
```

Эта программа выдаст ответ «Телевизор смотрит Даша» и «Телевизор смотрит Глеб».

Задачи

I. Алгоритм и программа

1. Написать алгоритмы вычисления следующих выражений:

a) $\frac{8+1}{3} + 2(15+17)$;

b) $\frac{95}{100} - \left(\frac{3}{10} + \frac{2}{5}\right)$;

c) $\frac{5}{6} - \left(\frac{3}{10} + \frac{2}{5}\right)$;

d) $\frac{1}{4} + \left(1 - \frac{11}{12}\right)$;

e) $6\frac{3}{16} - \left(2\frac{3}{8} + 3\frac{5}{12}\right)$;

f) $\left(63\frac{2}{3} + 3\frac{1}{8}\right) - \left(13 - 10\frac{5}{9}\right)$;

g) $\left(15\frac{1}{2} - 2\frac{3}{8}\right) - \left(5\frac{5}{6} + 6\frac{3}{4}\right) + \left(10\frac{2}{3} - 5\frac{5}{8}\right)$;

h) $\frac{1}{36} : \left(\frac{11}{12} - \frac{5}{9}\right) + \frac{12}{13}$;

i) $\frac{1}{8} : \left(3\frac{15}{17} - 1\frac{123}{158}\right) + \frac{12}{13}$.

Написать программы решения следующих задач:

2. Аквариум имеет длину 50 см, ширину 40 см. Какова высота аквариума, если в него вмещается 80 000 куб. см воды.

3. На двух полках было 70 книг. На одну полку добавили 15 книг, с другой сняли 8 книг. Сколько книг стало на обеих полках?

4. В четырех товарных вагонах было 42 000 кг груза, во всех вагонах поровну. На одной из станций в один из вагонов загрузили еще 620 кг, из второго вагона перегрузили в третий 1340 кг, а из четвертого выгрузили 960 кг. Сколько килограммов груза стало в каждом вагоне?

5. Выручка трех отделов магазина за день составила 35 000 рублей. Выручка первого отдела была 12 000 рублей, выручка второго — на 2000 рублей меньше. Какова выручка третьего отдела?

6. Три колхоза вывозили с железнодорожной станции минеральные удобрения. Первый вывез 156 тонн, второй на 5 тонн больше, чем первый,

а третий — столько же, сколько первый и второй вместе. Сколько тонн удобрений вывезли все три колхоза вместе?

7. Мастер за час обрабатывает 12 заготовок, а его ученик — 8 заготовок. Над выполнением полученного задания сначала работал мастер 2 часа, а затем мастер и ученик работали вместе в течение 3 часов. Сколько заготовок они сделали за это время?

8. От двух пристаней одновременно навстречу друг другу отошли два катера и встретились через 2 часа. Скорость одного из них равна 22 км/ч, скорость другого — 27 км/ч. Найти расстояние между пристанями.

9. На концерте было 894 зрителя. Из них 396 человек купили билеты по 150 рублей, остальные — по 200 рублей. Сколько денег было выручено от продажи билетов?

10. На складе лесопилки находится 18 штабелей бревен по 64 бревна в каждом и 9 штабелей по 42 бревна в каждом. Сколько всего бревен на складе?

11. От станции одновременно отошли два поезда в противоположных направлениях. Скорость одного поезда 55 км/ч, скорость другого — 45 км/ч. Какое расстояние будет между поездами через 6 часов?

12. На первом участке автомобиль двигался 3 часа со скоростью 40 км/ч, на втором — 1 час со скоростью 60 км/ч. Какова средняя скорость автомобиля на всем пути?

13. Теплоход прошел по реке 70 км за 2 часа и 90 км по озеру за 3 часа. С какой средней скоростью прошел теплоход весь путь?

14. Овощевод снял с одного куста помидоров 15 плодов по 250 г, 10 плодов по 330 г и 8 плодов по 210 г. Найдите среднюю массу помидоров.

15. В магазин завезли 850 кг огурцов. Первый покупатель взял 1 % всех огурцов, второй взял 3 % всех огурцов. Сколько огурцов осталось в магазине? Сколько огурцов останется в магазине, если второй покупатель возьмет 3 % от оставшихся огурцов?

16. Бригаде поручили отремонтировать участок дороги длиной 750 м. В 1-й день бригада отремонтировала 40 % дороги, во 2-й — 30 %. Сколько метров дороги осталось бригаде отремонтировать через 2 дня работы?

17. Из зерна пшеницы получается 80 % муки (по весу), из ржи — 75 %. Сколько муки получится из 4 тонн пшеницы и 5 тонн ржи?

18. Ученики 4 А класса собрали 220 кг макулатуры, ученики 4 Б — на 60 кг больше, а ученики 4 В — на 10 кг меньше, чем ученики 4 А и 4 Б классов вместе. Сколько килограммов макулатуры собрали ученики трех классов вместе?

19. В магазине заплатили 78 рублей за сыр, за молоко на 8 рублей меньше, чем за сыр, а за сахар — на 2 рубля больше, чем за сыр. Сколько стоит вся покупка?

20. Доярки надоили за июль 300 тыс. литров молока, что на 4 тыс. литров больше, чем они надоили в июне, и на 6 тыс. литров меньше, чем они надоили в августе. Сколько литров молока доярки надоили за летние месяцы?

21. Мальчику 10 лет. Он на 2 года старше своего брата, а их отец в 5 раз старше младшего из мальчиков. Сколько лет отцу?

22. Первый рабочий за 1 час делает 32 детали, а второй за 4 часа делает столько же деталей, сколько первый делает за 5 часов. За сколько часов, работая вместе, они сделают 216 деталей?

23. Одна машинистка печатает 10 страниц за 1 час, а вторая за 5 часов печатает столько же, сколько первая за 4 часа. Сколько страниц отпечатают обе машинистки за 3 часа совместной работы?

24. В первой коробке лежало 12 мячей, во второй — в 2,5 раза больше, чем в первой, а в третьей — в 1,2 раза меньше, чем в первой. На сколько мячей больше во второй коробке, чем в третьей?

25. Ширина прямоугольного параллелепипеда равна 3, длина в 2,5 раза больше ширины, а высота в 1,5 раз больше длины. Вычислить объем параллелепипеда.

II. Оператор присваивания

26. Какие из следующих последовательностей символов являются идентификаторами, а какие — нет?

- | | | | | | |
|--------|-----------|-------------|---------|----------|-----------|
| a) x | e) ax256 | i) a_1 | o) itis | s) m'x | w) x1x2x3 |
| b) x*x | f) ax.256 | j) a.1 | p) АБВГ | t) 'mmm' | x) 1x2x3x |
| c) x1 | g) 2x | l) s+2 | q) ABCD | u) abc% | |
| d) ax | h) a-1 | m) velocity | r) m15x | v) a%bc | |

27. Какие из следующих последовательностей символов являются операторами присваивания?

- | | | |
|--|-----------------------|---------------------------------|
| a) $abc = 256$ | d) $y = k * x + b$ | g) $y = x ;$ |
| b) $a + b = c + d$ | e) $y = kx + b;$ | h) $a1 = \text{abs}(x) - z;$ |
| c) $x^2 = y;$ | f) $xyz = x^2 + y^2;$ | i) $a * x^2 + b * y^2 + c = 0;$ |
| j) $2 + 3 = a;$ | | |
| k) <code>petya.ivanov = 15 + 8.</code> | | |

l) `skorost = rasstoyanie/vremya`

28. Перевести на язык QBASIC следующие выражения:

- увеличить значение переменной N на 1;
- увеличить значение переменной A вдвое;
- вычислить корень квадратный из числа, записанного в переменной B, результат записать в переменную C;
- вычислить среднее арифметическое чисел A и B, результат записать в переменную A;
- увеличить значение переменной C на 5;
- возвести переменную B в квадрат;
- вычислить среднее арифметическое чисел A, B, C; результат записать в переменную X.

29. Увеличить значение переменной A вдвое, не используя операции умножения.

30. Увеличить значение переменной A в 4 раза за две операции сложения.

31. Увеличить значение переменной A в 6 раз за три операции сложения. Использовать вспомогательную переменную.

32. Даны два числа a и b . Число a возвести в куб, число b возвести в квадрат. Вычислить сумму получившихся чисел. Результат запомнить в переменной C .

33. Дан квадрат со стороной a . Вычислить периметр квадрата, используя только одну переменную.

34. Дан прямоугольник, у которого ширина вдвое меньше длины. Вычислить периметр прямоугольника, используя только одну переменную.

35. У аквариума ширина вдвое меньше длины, а высота в 1,5 раза больше ширины. Вычислить емкость аквариума. Использовать только одну переменную.

36. Дано число a . Не пользуясь никакими другими арифметическими операциями, кроме умножения, получить:

- a) a^4 за две операции;
- b) a^6 за три операции;
- c) a^7 за четыре операции;
- d) a^8 за три операции;
- e) a^9 за четыре операции;
- f) a^{10} за четыре операции.

37. Записать следующие формулы в виде арифметических выражений

a) $x^{(y+z)}$;

b) $1 + x + \frac{x^2}{2} + \frac{x^3}{2 \cdot 3}$;

c) $3 \sin x + 4 \cos^2 x^2 - 1$;

d) $\sqrt{1 + \sqrt{1 + \sqrt{2}}}$;

e) $\frac{a - b}{c + \frac{b}{c + \frac{b}{c - d}}}$;

f) $1 + \frac{1}{1 + \frac{1}{1 + x}}$;

g) $\left(x - 1 + \frac{1}{x}\right) \times \left(\frac{x^2 - x}{(2 - x)^2}\right)$.

38. Написать операторы присваивания, соответствующие следующим формулам:

a) $c = \sqrt{a^2 + b^2}$;

b) $a = (x + y)^2$;

c) $p = \sqrt{\frac{(a + b) \cdot c}{d}} + e - r$;

d) $z = \sqrt{|x - 1|} + \sqrt{|y|}$;

e) $y = \frac{1}{x} + 1$;

f) $y = \frac{1}{x^2 - 1}$;

g) $y = \frac{1}{|x| - 1}$;

h) $z = \frac{a + b}{2a - c} (a + c)$;

i) $z = 3,14 + \frac{x + y}{1 - xy}$;

$$j) x = \frac{2}{\frac{1}{a} + \frac{1}{b}};$$

$$k) z = \frac{x^2 + y^2}{1 - \frac{x^2 - y^2}{2}};$$

$$l) x = \frac{-b + \sqrt{b^2 - 4ac}}{2a};$$

$$m) x = \frac{a}{b \cdot \frac{c}{d \cdot \frac{e}{f \cdot h}}};$$

$$n) x = \frac{5}{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d} + \frac{1}{e}};$$

$$o) z = \frac{1 - \sqrt{1 + |x^2 - y^2|}}{2 \cdot |x^2 - y^2|};$$

$$p) x = \frac{a + b - 1,7}{c + \frac{d}{e + f + 0,5}};$$

$$q) z = \left(\frac{x + 3a + y}{2x} \right)^4 - \frac{x}{x + 3a - y};$$

$$r) z = 0,693 + 2 \cdot \left(\frac{a}{4 + a} + \frac{a^3}{3(4 + a)^3} + \frac{a^5}{5(4 + a)^5} \right).$$

III. Линейные программы

39. Даны два числа. Вычислить их сумму, разность, произведение.

40. Вычислить среднее арифметическое трех чисел.

41. Дан куб со стороной a . Вычислить его объем и площадь поверхности.

42. В классе n учеников. По результатам контрольной работы получено a двоек, b троек, c четверок, остальные — пятерки. Вычислить процент оценок каждого типа.

43. Даны гипотенуза и катет прямоугольного треугольника. Вычислить периметр и площадь треугольника.

44. Прямая $y = k \cdot x + b$ задана своими коэффициентами. Вычислить периметр и площадь треугольника, образованного этой прямой и осями координат.

45. Дана окружность радиуса r . Вычислить периметр и площадь правильных многоугольников — описанного вокруг окружности и вписанного в нее.

46. Дана окружность радиуса r и точка с координатами (x, y) . Вычислить кратчайшее расстояние от точки до окружности. Вычислить координаты точки окружности, наиболее близкой к данной точке.

47. Треугольник задан координатами своих вершин. Вычислить периметр и площадь треугольника.

48. Многоугольник задан координатами своих вершин. Вычислить периметр и площадь многоугольника.

49. По заданной длине стороны правильного n -угольника найти величину его внутреннего угла, площадь, радиусы вписанной и описанной окружности.

50. Даны две прямые $y = k_1 \cdot x + b_1$ и $y = k_2 \cdot x + b_2$.

Найти координаты точки пересечения.

51. По заданным сторонам треугольника найти его углы. Выразить углы в градусах и радианах.

52. Вычислить расстояние до линии горизонта для наблюдателя, находящегося на высоте H над поверхностью Земли. Радиус Земли принять равным 6380 км.

53. Дано количество суток. Посчитать число часов, минут и секунд в данном количестве суток. Использовать только одну переменную.

IV. Условный оператор

54. Петя встал утром в 7 часов, Коля — на 13 минут раньше него, Сережа на 4 минуты позже Коли, а Саша встал на 10 минут позже Сережи. Кто встал раньше, Петя или Саша?

55. У Сережи 145 рублей. Он решил купить 3 блокнота по 48 рублей. Хватит ли денег на покупку?

56. Ребята собрали в саду 875 кг яблок и упаковали их в ящики по 25 кг в каждом. Хватило ли им 40 ящиков?

57. Из куска ткани можно сшить 17 детских плащей, расходуя на каждый 2 м ткани. Можно ли из этого куска сшить 8 плащей для взрослых, расходуя по 4 м ткани на каждый плащ?

58. В коробке 3 кг лимонов и апельсинов. Лимоны вынули и взвесили отдельно. Их масса оказалась 1 кг 760 г. Каких цитрусовых больше и на сколько?

59. Валя взяла у подруги книгу на 3 дня. В первый день она прочитала A страниц, во второй день прочитала в B раз больше, чем в первый, в третий день она прочитала на C страниц меньше, чем во второй. Успела ли Валя прочитать книгу вовремя, если в книге 194 страницы? Решить задачу при следующих значениях переменных:

— $A = 28$, $B = 3$, $C = 25$;

— $A = 30$, $B = 3$, $C = 16$.

60. Мотоциклист должен прибыть в город через T часов после выезда из пункта, находящегося на расстоянии S км от города. Успеет ли он вовремя прибыть в город, если в первый час он проедет A км, а все оставшееся время будет ехать со скоростью B км/ч? Решить задачу при следующих значениях переменных:

— $T = 4$, $S = 320$, $A = 75$, $B = 70$;

— $T = 5$, $S = 300$, $A = 65$, $B = 60$.

61. В театр пришло A зрителей. Из них B человек купили билеты по 30 рублей, остальные — по 20 рублей. Получит ли театр прибыль, если расходы на спектакль составили Z рублей? Решить задачу при следующих значениях переменных:

— $A = 800$, $B = 300$, $Z = 15000$;

— $A = 600$, $B = 100$, $Z = 15000$.

62. Расстояние между городами А и Б равно S . Из города А в город Б выехал велосипедист со скоростью V_1 , через T часов вслед ему выехал второй велосипедист со скоростью V_2 . Догонит ли второй велосипедист первого, прежде чем тот доедет до города Б? Решить задачу при следующих значениях переменных:

— $S = 50$, $V_1 = 12$, $V_2 = 16$, $T = 2$;

— $S = 50$, $V_1 = 12$, $V_2 = 16$, $T = 1$.

63. Даны два числа. Вывести их на экран в порядке возрастания.

64. Даны два числа. Вывести на экран первое число, если оно больше второго, и оба числа, если это не так.

65. Даны два числа. Меньшее из этих чисел заменить их полусуммой, а большее — удвоенным произведением.

66. Составить программу, находящую максимальное число из трех заданных.

67. Составить программу, находящую максимальное число из четырех заданных.

68. Даны четыре числа. Найти разницу между максимальным и минимальным числами.

69. Определить принадлежность точки кругу радиуса R с центром в начале координат.

70. Определить принадлежность точки заданному отрезку.

71. Даны три числа. Выяснить, существует ли треугольник с такими сторонами.

72. Две точки заданы своими координатами. Выяснить, лежат ли они на одной окружности с центром в начале координат.

73. Даны три числа a , b , c . Выяснить, выполняется ли условие $a < b < c$.

74. Числовая прямая разбита на отрезки числами $A_1, A_2, A_3 \dots A_n$. Определить, какому отрезку принадлежит заданная точка x .

75. В киоске продается газета стоимостью 3 рубля и журнал стоимостью 20 рублей. Составить программу, которая спрашивает, что вы хотите купить, принимает деньги и выдает сдачу.

76. Составить программу, моделирующую работу продавца мороженого. Программа должна запрашивать количество порций мороженого, сообщать требуемую сумму и «выдавать сдачу».

77. Составить программу, моделирующую работу продавца мороженого с выбором сорта мороженого.

78. Определить, какая из двух точек находится ближе к окружности радиуса R с центром в начале координат.

79. Даны два числа x и y . Вычислить:

a) $z = x - y$, если $x > y$, $z = y - x + 1$ в противном случае;

b) $z = 13.8$, если $x > y$, $z = 20.0$ в противном случае.

80. При выполнении условия $x - y < 0$ вычислить $z = (x + y)/2 + 1$, в противном случае проверить условие $x - y = 0$ и, если оно выполняется, вычислить $z = (x + y)/2$, в противном случае $z = 3.2$.

81. Даны действительные числа x и y . Вычислить:

$$a) z = \begin{cases} x - y, & \text{если } x > 0; \\ y - x + 1 & \text{в противном случае} \end{cases}$$

$$b) z = \begin{cases} x \times y, & \text{если } x > 0, y > 0; \\ x + y, & \text{если } x < 0, y < 0; \\ 0 & \text{Р во всех остальных случаях;} \end{cases}$$

$$c) F = z - 6,3 \times x, \quad \text{ч}$$

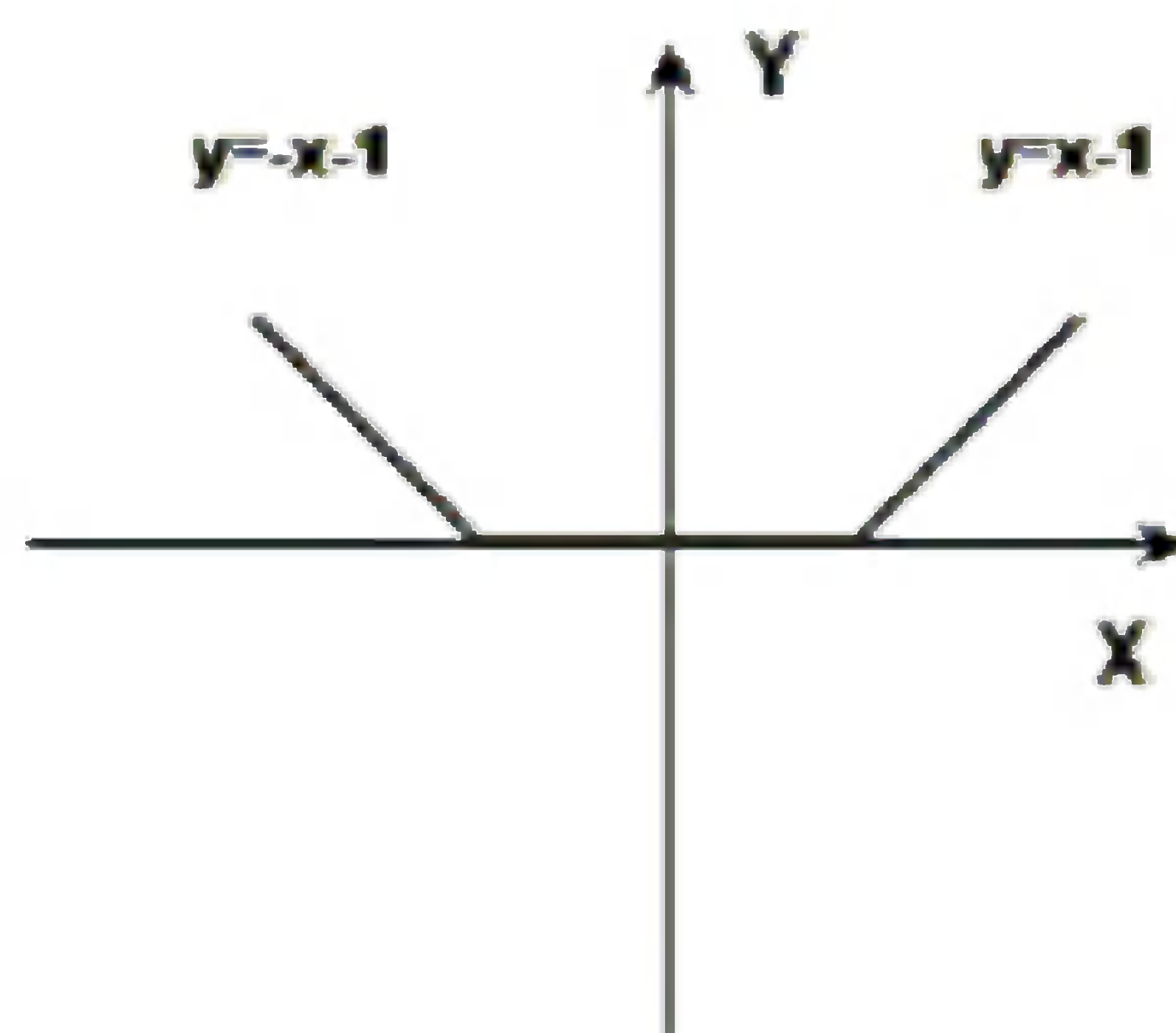
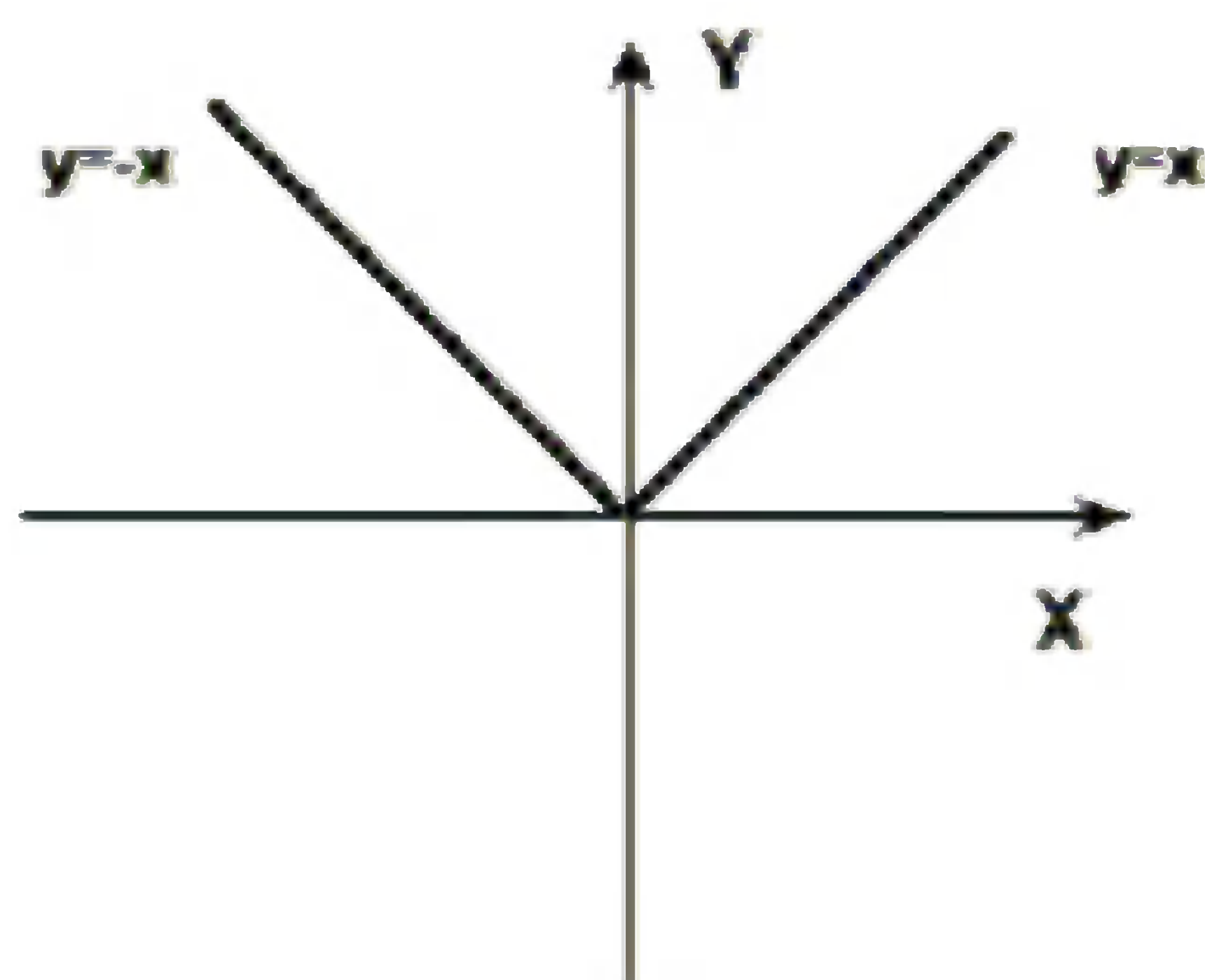
$$\text{где } z = \begin{cases} y \times y - 3, & \text{если } x < 0, \\ 0, & \text{если } 0 \leq x \leq 1, \\ y \times y, & \text{если } x > 1. \end{cases}$$

82. Вводится число x . Вывести на экран значение функции $y = \frac{1}{x}$.

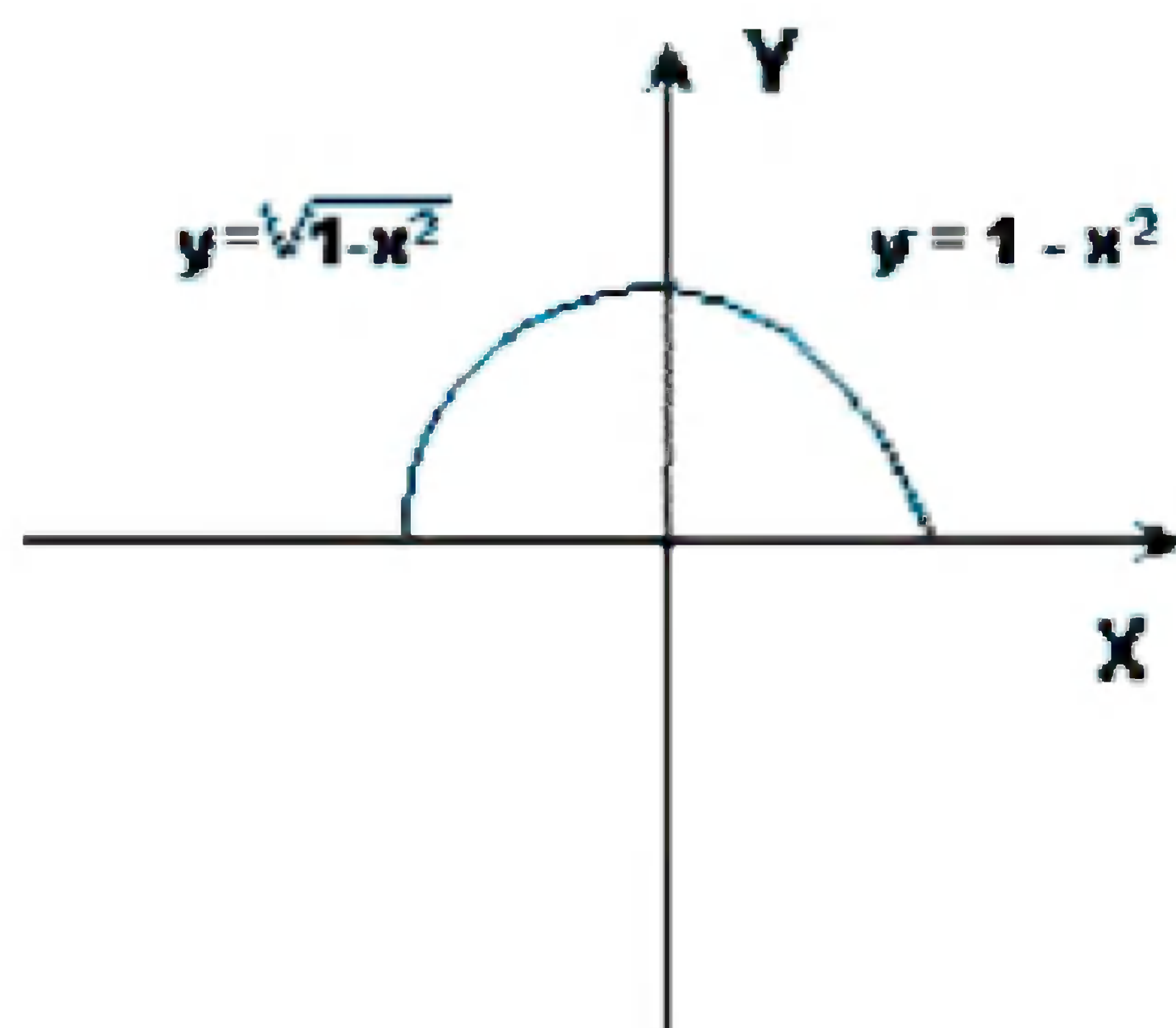
Если $x = 0$, вывести сообщение «функция не определена».

83. Вводится число x . Вычислить значения функций, графики которых приведены на рис.:

84. Вводится число x . Если $-1 \leq x \leq 1$, вывести на экран значение функции, график которой приведен на рис. Если $x < -1$ или $x > 1$, вывести на



экран надпись: «Функция не определена».



V. Циклы и массивы

85. Написать таблицу перевода температур из градусов Цельсия (C) в градусы Фаренгейта (F) по формуле $F = 1,8 \times C + 32$.

86. Напечатать таблицу соответствия между весом в фунтах и весом в килограммах. 1 фунт = 0,4 кг.

87. Напечатать таблицу перевода расстояний в дюймах в сантиметры. 1 дюйм = 2,54 см.

88. Возвести число в 7-ю степень, не пользуясь операцией возведения в степень.

89. Умножить число на 7, пользуясь только операцией сложения.

90. Составить таблицу умножения для числа 12.

91. Составить таблицу расстояний до линии горизонта для наблюдателя, находящегося на высоте H от поверхности Земли для величин H от 10 до 100 м с интервалом 10 м.

92. Дан прямоугольный параллелепипед высотой h. В основании параллелепипеда лежит квадрат со стороной a. Объем параллелепипеда равен 1. Составить таблицу значений a для $0,1 \leq h \leq 2$ с шагом 0,1.

93. Одноклеточная амеба каждые 3 часа делится на две клетки. Определить, сколько клеток будет через 3, 6, 9... 24 часа.

94. Вычислить среднее арифметическое элементов массива.

95. Вывести на экран положительные элементы массива.

96. Вычислить среднее арифметическое положительных и среднее арифметическое отрицательных элементов массива.

97. В массив занесены результаты контрольной работы. Посчитать процент оценок каждого типа.

98. В массив занесены данные о росте учеников класса. Рост мальчиков кодируется знаком «+», рост девочек — знаком «—». Посчитать средний рост мальчиков и средний рост девочек.

99. В массив A занесены среднесуточные температуры за апрель. В массив B занесены данные о количестве выпавших осадков (в мм) в эти же дни. Посчитать количество осадков, выпавших в виде дождя и количество осадков, выпавших в виде снега. Вывести на экран дни, в которые шел дождь, дни, в которые шел снег, и дни, в которые не было осадков.

100. На плоскости задана система точек. Найти точку, наиболее близкую к началу координат, и точку, наиболее удаленную от начала координат.

101. На плоскости задана система точек. Вывести на экран координаты точек, лежащих в 1-й четверти.

102. На плоскости задана система точек. Вывести на экран последовательно координаты точек, лежащих в 1-й, 2-й, 3-й, 4-й четвертях.

VI. Графика

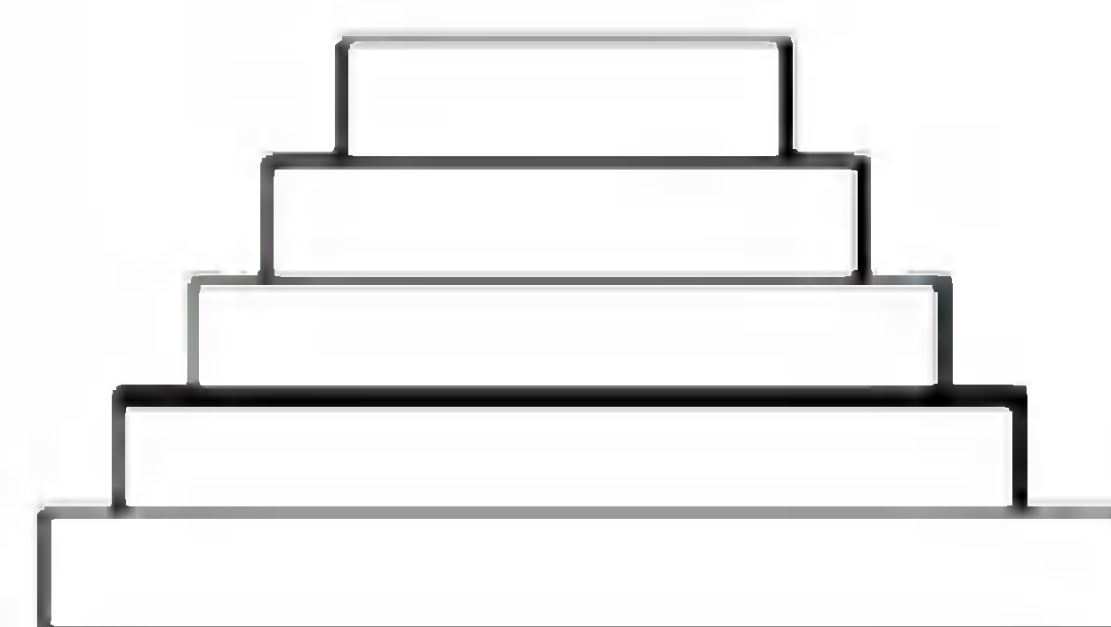
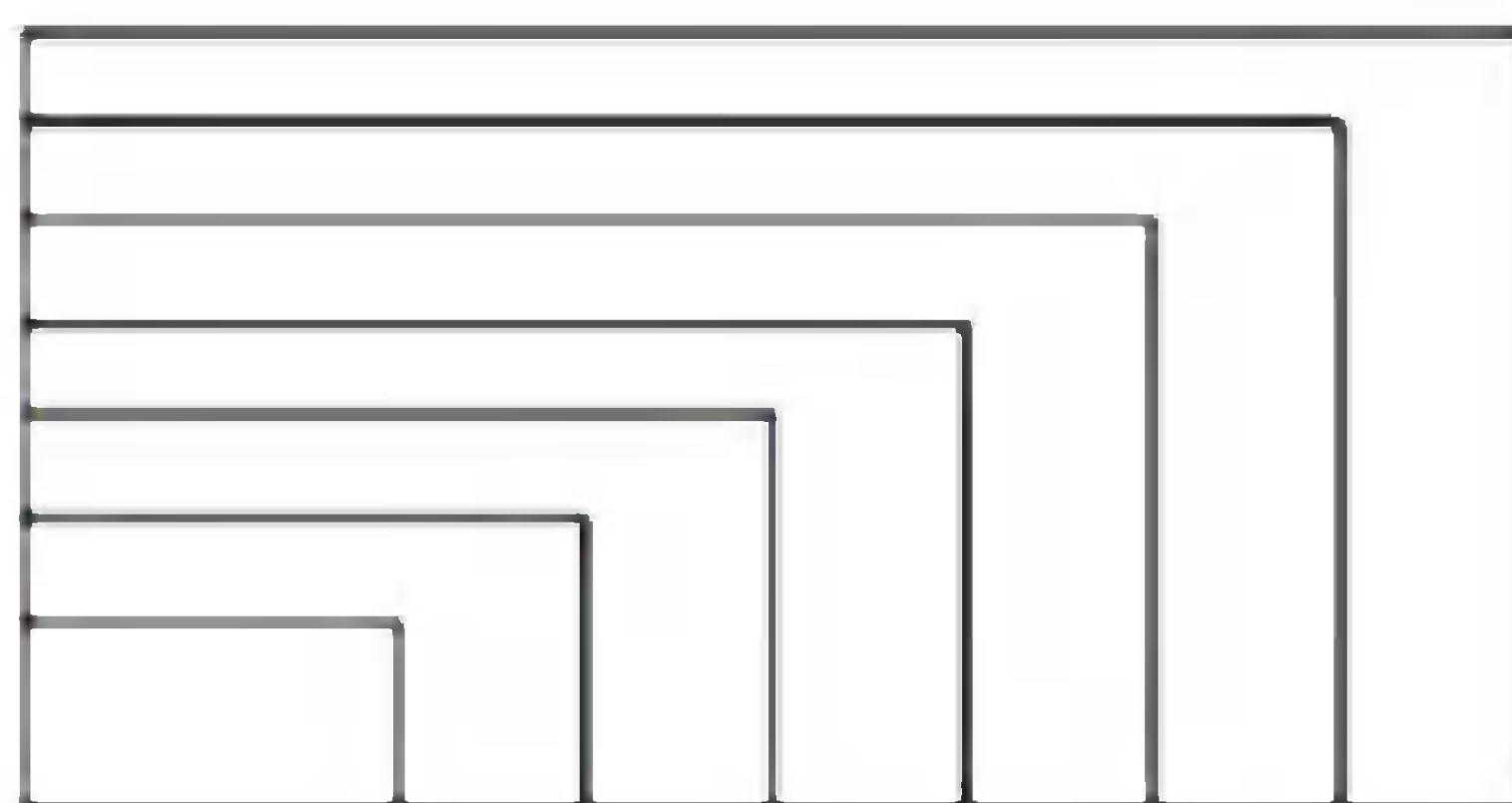
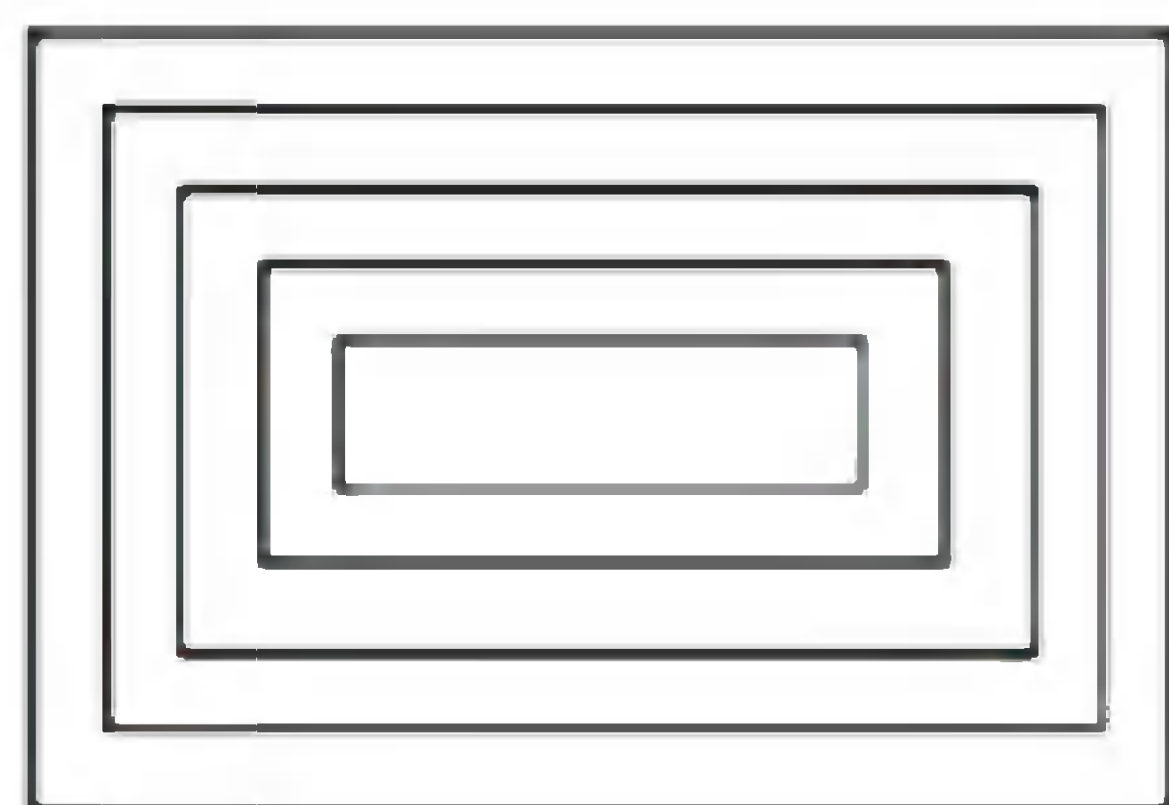
103. Нарисовать шахматную доску.

104. Раскрасить шахматную доску в шахматном порядке.

105. Нарисовать российский флаг.

106. Нарисовать светофор.

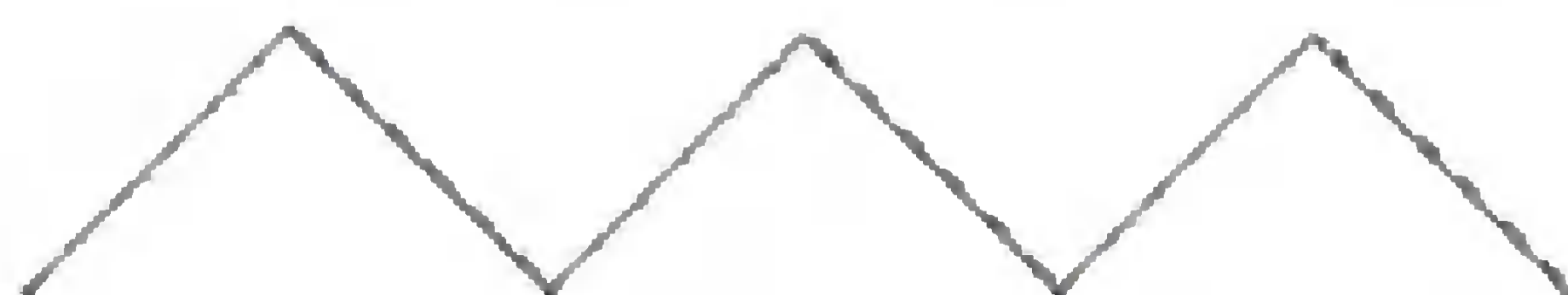
107. Нарисовать следующие фигуры из прямоугольников:



108. Нарисовать систему концентрических окружностей. Раскрасить кольца в разные цвета.

109. Нарисовать радугу.

110. Нарисовать следующие ломаные линии:



111. Построить графики следующих функций:

a) $y = 2x^2$;

b) $y = 2x^2 + 2$;

c) $y = 2(x + 2)^2$;

d) $y = x^2 - 2x + 3$;

e) $y = x^2 + 2x - 3$;

f) $y = 1 - x^2$;

112. Построить параболу с вершиной в точке (x_0, y_0) .

113. Построить параболу с вершиной в точке (x_0, y_0) , перевернутую «верх ногами».

114. Построить графики следующих функций:

a) $y = \frac{1}{x+1}$;

b) $y = \frac{1}{x} + 1$;

c) $y = \frac{1}{1-x}$;

d) $y = \frac{1}{|x|}$;

e) $y = \frac{1}{|x|} + 1$;

f) $y = \frac{1}{|x|+1}$;

$$g) y = \frac{1}{|x + 1|};$$

$$h) y = \frac{1}{x^2};$$

$$i) y = \frac{1}{x^2 + 1};$$

$$j) y = \frac{1}{x^2 - 1};$$

$$k) y = \frac{1}{(x + 1)^2};$$

$$l) y = \frac{1}{(x - 1)^2};$$

$$m) y = \frac{1}{1 - x^2};$$

$$n) y = \frac{1}{x^2 - 2x + 3};$$

$$o) y = \frac{1}{x^2 + 2x - 3};$$

115. Построить окружность как график функции $x^2 + y^2 = r^2$.

116. Нарисовать окружность. Заштриховать окружность вертикальными линиями, горизонтальными линиями, в клеточку.

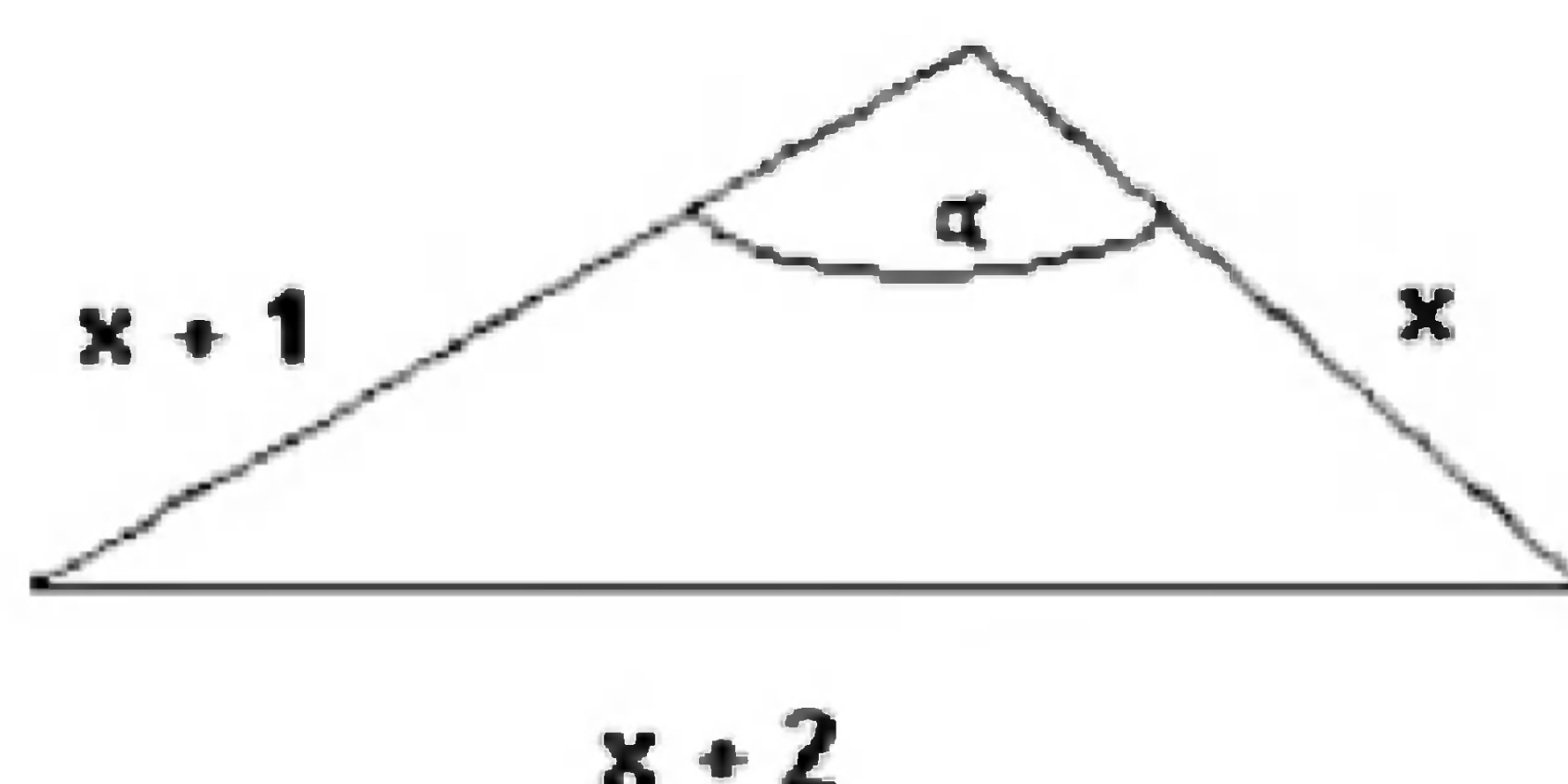
117. Отрезок скользит вниз, опираясь на стороны прямого угла так, что один конец отрезка скользит по вертикальной стороне угла, другой — по горизонтальной. Нарисовать:

- последовательные положения отрезка;
- траекторию движения середины отрезка.

118. Два пешехода идут во взаимно перпендикулярных направлениях с разными скоростями.

Нарисовать линию, которую описывает середина отрезка, соединяющая пешеходов.

119. Дан треугольник со сторонами x , $x + 1$, $x + 2$. Угол α , лежащий против большей стороны, является функцией от x . Построить график функции $\alpha = f(x)$.



VII. Арифметическая и геометрическая прогрессии

120. Даны два первых члена арифметической прогрессии. Задать прогрессию при помощи рекуррентной формулы. Вычислить n -й член прогрессии:

а) $\frac{1}{3}, -1 \dots$

в) $-8, -6,5$

б) $2,3, 1\dots$

г) $11, 7\dots$

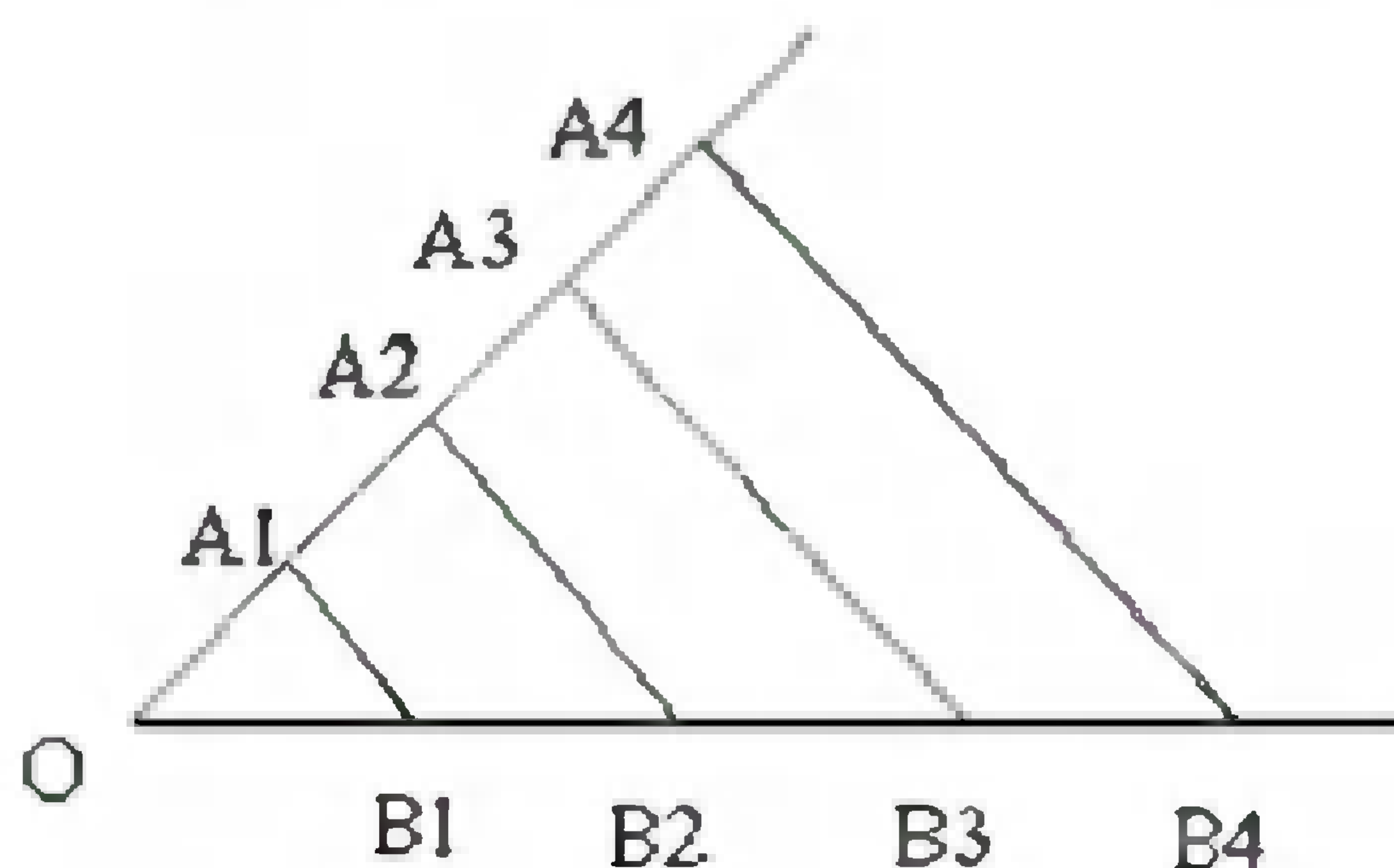
121. Тело в первую секунду движения прошло 7 м, затем за каждую последующую секунду — на 3 м больше, чем за предыдущую. Какое расстояние прошло тело за 8-ю секунду?

122. Поезд, отойдя от станции, равномерно увеличивал скорость на 50 м в минуту. Какую скорость приобрел поезд за 20 минут?

123. Даны n -й член арифметической прогрессии и разность. Вычислить 1-й член.

124. Даны n -й член арифметической прогрессии и разность. Вычислить k -й член.

125. На стороне OA угла AOB от его вершины отложены равные отрезки и через их концы проведены параллельные прямые. Длина отрезка A_1B_1 равна 1,5 см. Найти длину n -го отрезка.



126. Даны первые два члена геометрической прогрессии. Вывести на экран 10 членов прогрессии:

а) $2, -6\dots$

в) $-0.125, 0.25\dots$

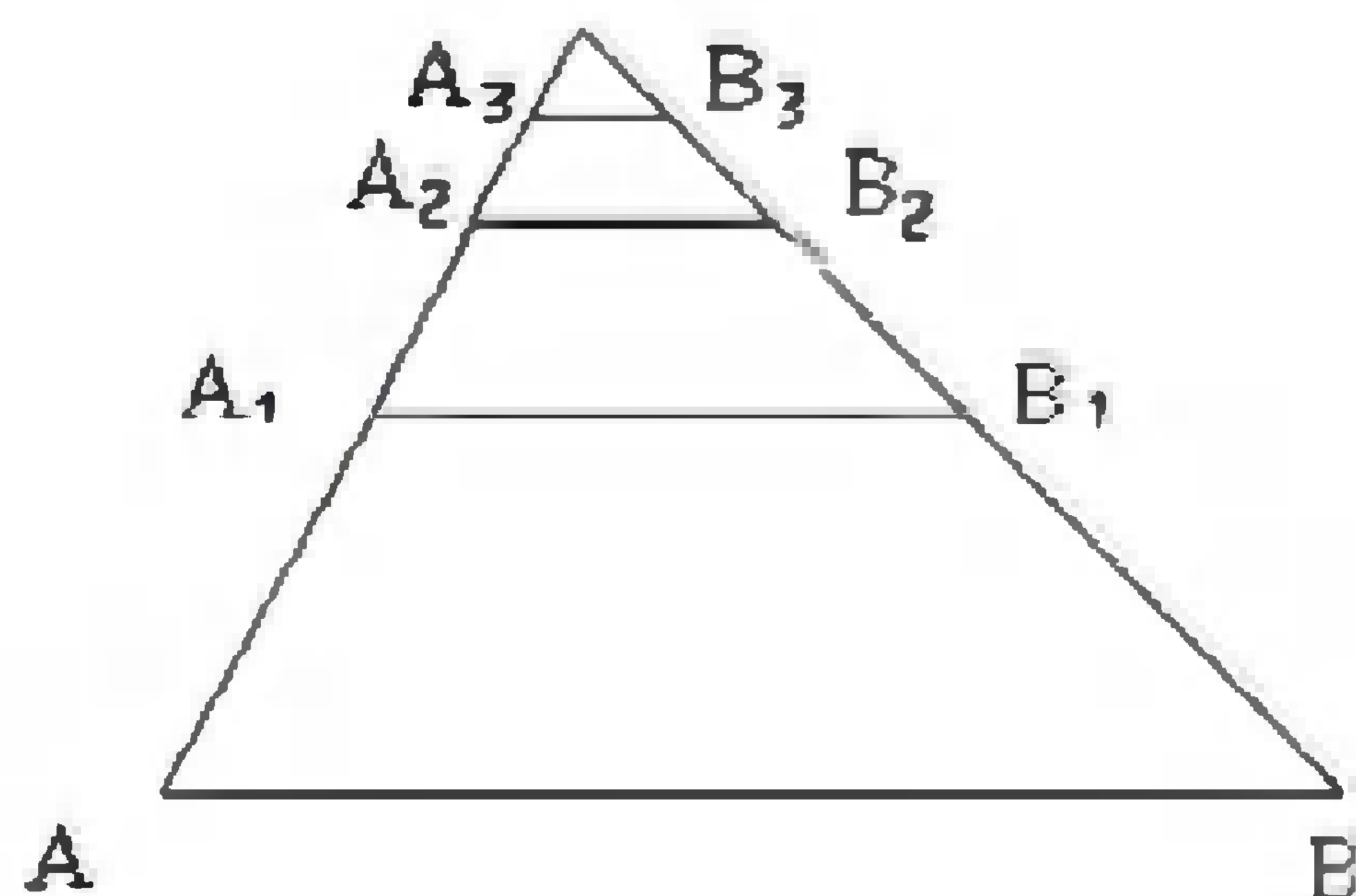
б) $-40, -20$

г) $-10, 10\dots$

127. После каждого движения поршня разряжающего насоса из сосуда удаляется 20 % находящегося в нем воздуха. Определить давление воздуха внутри сосуда после шести движений поршня, если первоначальное давление было 760 мм рт. ст.

128. Срочный вклад, положенный в банк, ежегодно увеличивается на 3 %. Каким будет вклад через 5 лет, если вначале он равнялся 100 000 руб.?

129. В треугольнике ABC провели среднюю линию A_1B_1 , затем в треугольнике A_1B_1C провели среднюю линию A_2B_2 , затем в треугольнике A_2B_2C провели среднюю линию A_3B_3 и т. д. Найти площадь n -го треугольника, если площадь треугольника ABC равна 1024 см^2 .



VIII. Вычисление сумм и произведений

130. Найдите сумму:

- а) всех двузначных чисел;
- б) всех трехзначных чисел;
- в) всех натуральных четных чисел, не превосходящих 200;
- г) всех натуральных нечетных чисел, не превосходящих 150;
- д) всех натуральных чисел, кратных 3, заключенных в промежутке от 100 до 300;
- е) меньших 100 и не кратных 3;
- ж) больших 50, но меньших 150 и не кратных 5.

131. Тело в первую секунду движения прошло 7 м, а за каждую следующую секунду — на 3 м больше, чем за предыдущую. Какое расстояние прошло тело за 8 секунд?

132. Поезд, отойдя от станции, равномерно увеличивал скорость на 50 м в минуту. Какой путь поезд прошел за 20 минут, двигаясь равноускоренно?

133. При свободном падении тело за первую секунду проходит 4,9 м и за каждую следующую секунду на 9,8 м больше, чем за предыдущую. Какое расстояние пролетит свободно падающее тело за 7 секунд?

134. Шары расположены в форме треугольника так, что в первом ряду расположен один шар, во втором — два, в третьем — три и т. д. Сколько потребуется шаров, чтобы составить треугольник из 30 рядов?

135. Члены арифметической прогрессии 2, 5 ... с четными номерами заменили противоположными им числами. В результате получили новую последовательность. Найти сумму первых 50 членов.

136. Вычислить:

$$P = (1 + \sin 0,1) (1 + \sin 0,2) (1 + \sin 0,3) \dots (1 + \sin 10).$$

137. Дано натуральное n , действительное a . Вычислить:

а) $\left(1 + \frac{1}{1^2}\right) \left(1 + \frac{1}{2^2}\right) \left(1 + \frac{1}{3^2}\right) \dots \left(1 + \frac{1}{n^2}\right);$

б) $a(a + 1)(a + 2)(a + 2) \dots (a + n - 1);$

$$в) \frac{1}{a} + \frac{1}{a(a+1)} + \frac{1}{a(a+1)(a+2)} + \dots + \frac{1}{a(a+1)(a+2)\dots(a+n-1)}$$

$$г) \frac{(a-2)(a-4)(a-8)\dots(a-64)}{(a-1)(a-3)(a-7)\dots(a-63)}$$

138. Дано натуральное n . Вычислить суммы:

$$1! + 2! + 3! + 4! + \dots + n!$$

$$1! + 3! + 5! + 7! + \dots + (2n+1)!!$$

139. Вычислить $S = m(m+1)(m+2)\dots n$, где m и n — натуральные числа, $m < n$.

140. Вычислить сумму 10 слагаемых:

$$S = 2^1 + 2^2 + 2^3 + 2^4 + \dots + 2^{10}.$$

141. Вычислить сумму:

$$S = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \dots + \frac{10}{11}$$

142. Дано натуральное n . Вычислить сумму:

$$S = 1 + \frac{1}{a^2} + \frac{1}{a^4} + \frac{1}{a^6} + \dots + \frac{1}{a^{2n}}$$

IX. Циклы типа «До» и типа «Пока»

143. Шары расположены в форме треугольника так, что в первом ряду расположен один шар, во втором — два, в третьем — три и т. д. Сколько рядов удастся построить, если имеется n шаров?

144. После каждого движения поршня разряжающего насоса из сосуда удаляется 20 % находящегося там воздуха. Сколько движений поршня нужно сделать, чтобы достичь давления P при начальном давлении P_n ?

145. Найдите натуральное число, которое:

а) в 5 раз меньше суммы предшествующих ему натуральных чисел;

б) равно сумме предшествующих ему натуральных чисел.

146. Найти сумму:

а) всех положительных членов арифметической прогрессии 8.2, 7.4 ...;

б) всех отрицательных членов арифметической прогрессии —6.5, —6 ...

147. в первый день пробежал 10 км. Каждый следующий день он увеличивал дневную норму на 10 % от нормы предыдущего дня. Найти:

а) какой путь пробежит спортсмен в 7-й день;

б) через сколько дней спортсмен будет пробегать более 20 км в день;

в) через сколько дней спортсмен пробежит суммарный путь более 100 км.

148. Сколько чисел нужно взять в последовательности $1+2+3+4+\dots$ чтобы получить число большее N ?

149. Сколько чисел нужно взять в последовательности $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots$ чтобы получить число больше N ?

150. Дано целое число $m > 0$. Найти наибольшее целое k , при котором $4^k < m$.

151. Дано натуральное число n . Получить наименьшее число вида 2^k , превосходящее n .

152. Дано положительное число $a < 1$. Найти:

a) наибольшее число вида $\frac{1}{2^n}$ меньше a ;

b) наименьшее число вида $\frac{1}{3^n}$ больше a .

153. Вычислить наибольшее целое положительное число n , удовлетворяющее условию:

a) $3n^2 - 127n < 0$;

b) $-4n + 81\sqrt{n} + 3 \geq 0$;

c) $7n^3 + 81n^2 - 10^6 < 0$;

d) $-n^n + 300n^2 - 1000 > 0$.

154. Для данного числа a найти такое натуральное n , для которого $n! < a$, $(n+1)! > a$.

155. Число x возводят в квадрат и прибавляют 1. Полученное число снова возводят в квадрат и прибавляют 1 и т. д. Задать полученную числовую последовательность рекуррентной формулой. Вычислить n -й член последовательности. Найти первый член последовательности, больший заданного числа z .

156. Дана числовая последовательность $a_n = 100 + n - n^3$. Найти номер и значение первого отрицательного члена.

157. Дана числовая последовательность $a_n = n^2 - 125n + 1665$. Найти номер и значение первого отрицательного члена.

158. Послан человек из Москвы в Вологду, и велено ему в хождении своем совершать во всякий день по 40 верст. На следующий день вслед ему послан второй человек, и приказано ему проходить в день по 45 верст. На какой день второй человек догонит первого?

159. За два часа до обеденного перерыва 40 бабушек встали в очередь за пенсией. Кассир обслуживает клиента в среднем за 1 минуту. Первая бабушка мучила кассира вопросами 9 минут 15 секунд. Каждая следующая бабушка, «мотая на ус» ответы, адресованные предыдущим бабушкам, мучает кассира на 12 секунд меньше. Сколько бабушек успеет обслужить кассир до перерыва?

160. Два подмастерья одновременно отправились пешком из Виттенберга в Испанию. Первый ежедневно проходил 7 миль, второй в первый день прошел 1 милю, во второй — 2 мили, в третий — 3 мили и т. д. — в каждый следующий день на 1 милю больше, чем накануне. Через сколько дней подмастерья встретились?

Х. Вычисления с заданной точностью

Вычислить следующие бесконечные суммы:

$$161. \sum_{n=0}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

$$162. \sum_{n=1}^{\infty} (-1)^{n+1} \frac{1}{n} \approx 0,693$$

$$163. \sum_{n=0}^{\infty} (-1)^n \frac{1}{2n+1} = \frac{\pi}{4}$$

$$164. \sum_{n=1}^{\infty} \frac{1}{n(n+1)} = 1$$

$$165. \sum_{n=1}^{\infty} \frac{1}{n(n+2)} = \frac{3}{4}$$

$$166. \sum_{n=1}^{\infty} \frac{1}{n(n+1)(n+2)} = \frac{1}{4}$$

$$167. \sum_{n=0}^{\infty} \frac{1}{n!} = e$$

$$168. \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} = \frac{1}{e}$$

Дано действительное x . Вычислить:

$$169. \sum_{n=1}^{\infty} \frac{(-1)^{n+2}}{n! x^n}$$

$$170. \sum_{n=0}^{\infty} \frac{x}{n!} = e^x$$

$$171. \sum_{n=0}^{\infty} \frac{(-x)^n}{n!} = \frac{1}{e^x}$$

$$172. \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!!} = \sin(x)$$

$$173. \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!!} = \cos(x)$$

XI. Дроби и радикалы

Составить программы, вычисляющие следующие выражения:

$$174. 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{\dots 2 + \frac{1}{2}}}} = \sqrt{2}$$

$$175. \frac{1}{1 + \frac{1}{3 + \frac{1}{5 + \frac{1}{7 + \frac{1}{\dots 101 + \frac{1}{103}}}}}}$$

$$176. \frac{x}{1 + \frac{2x^2}{1 + \frac{3x^3}{1 + \frac{4x^4}{\dots 1 + \frac{(n-1)x^{n-1}}{x^n}}}}}$$

$$177. \frac{x}{x^2 + \frac{2}{x^2 + \frac{4}{x^2 + \frac{8}{\dots x^2 + \frac{256}{x^2}}}}}$$

$$178. 1 + \frac{1}{2 + \frac{2}{3 + \frac{3}{\dots n + \frac{n}{x}}}}$$

$$179. \sqrt{1 + \sqrt{2 + \sqrt{3 + \dots + \sqrt{n}}}}$$

$$180. \sqrt{2 + \sqrt{2 + \sqrt{2 + \dots + \sqrt{2}}}} \text{ — } n \text{ корней.}$$

$$181. \sqrt{n + \sqrt{n-1 + \sqrt{n-2 + \dots + \sqrt{2 + \sqrt{1}}}}}$$

$$182. \sqrt{3 + \sqrt{6 + \sqrt{9 + \dots + \sqrt{3(n-1) + \sqrt{3n}}}}}$$

$$183. \sqrt{\frac{1}{2}} \cdot \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2}}} \cdot \sqrt{\frac{1}{2} + \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2}}}} \dots = \frac{2}{\pi}.$$

ХII. Задачи на использование рекуррентных формул

184. Члены арифметической прогрессии 2, 5 ... с четными номерами заменили обратными им числами. В результате получили новую последовательность. Найти сумму первых 50 членов.

185. Из бильярдных шаров строится треугольная пирамида. Определить:

- а) сколько шаров потребуется для построения пирамиды из 10 слоев;
- б) сколько слоев удастся построить, если имеется 100 шаров?

186. На втулку диаметром 30 мм требуется намотать пружинную ленту толщиной 0,25 мм так, чтобы общий диаметр втулки с намотанной на ней лентой был равен 130 мм. Какой длины должна быть лента?

187. В 8 часов утра в воскресенье гусеница начала подниматься по дереву. В течение дня (до 18 часов) она поднимается на высоту 5 м, в течение ночи спускается на 2 м. В какой день и час она поднимется на высоту 9 м?

188. Определить суммарный объем вложенных друг в друга шаров, толщина стенок которых равна K . Радиус внешнего шара равен R .

189. Через сколько минут можно ставить суп на стол, если каждую минуту температура супа падает на 10 % от разности температур супа и комнаты? Принять начальную температуру супа равной 100° , температуру комнаты — 22° , температуру подаваемого супа — 60° .

190. Слесарь-водопроводчик чинит протекающий кран в течение полчаса. Из неисправного крана в час вытекает 0,5 литра воды. Сколько воды вытечет из всех кранов, которые слесарь успеет починить за день, если он работает с 8 до 17 часов с часовым перерывом на обед?

191. Рабочий обслуживает 16 ткацких станков, работающих автоматически. Производительность каждого станка a м/ч. Рабочий в начале рабочего дня включил первый станок и затем через каждые 5 мин включал следующий. Сколько метров ткани выработали станки за первые 2 часа?

192. Найти сумму всех несократимых дробей со знаменателем 3, заключенных между целыми положительными числами m и n ($m < n$).

193. Вычислить n -й член последовательности 2, -5 , 8, -11 , 14, -17 ...

194. Дано действительное число x , натуральное n . Вычислить суммы:

а) $S = \sin(x) + \sin(x^2) + \sin(x^3) + \dots + \sin(x^n)$;

б) $S = \sin(x) + \sin^2(x) + \sin^3(x) + \dots + \sin^n(x)$;

в) $S = \sin(x) + \sin(\sin(x)) + \sin(\sin(\sin(x))) + \dots$;

г) $S = \frac{\cos 1}{\sin 1} + \frac{\cos 1 + \cos 2}{\sin 1 + \sin 2} + \frac{\cos 1 + \cos 2 + \cos 3}{\sin 1 + \sin 2 + \sin 3} + \dots$ — n слагаемых.

195. Вычислить сумму:

$$S = 1 + 11 + 111 + 1111 + \dots + \underbrace{111\dots 1}_n.$$

196. Вычисление квадратного корня числа методом Ньютона.

Последовательность x_n , заданная рекуррентной формулой:

$$x_0 = \frac{a}{2},$$

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{a}{x_n} \right).$$

стремится к значению, равному $\sqrt{2}$. Написать программу вычисления $\sqrt{2}$ используя данную последовательность.

197. Последовательность x_n , построенная по рекуррентной формуле

$$x_0 = \frac{a}{2},$$

$$x_{n+1} = \frac{1}{m} \left((m-1)x_n + \frac{a}{x_n^{m-1}} \right).$$

сходится к значению $\sqrt[m]{a}$. Составить программу вычисления $\sqrt[m]{a}$ используя последовательность x_n .

198. Рассмотрим правильный n -угольник, вписанный в окружность. Обозначим его периметр p_n . Увеличим количество сторон n -угольника вдвое и периметр $2n$ -угольника обозначим p_{2n} . Периметры p_n и p_{2n} связаны соотношением

$$p_{2n} = \frac{p_n}{\sqrt{\frac{1}{2} + \frac{1}{2} \sqrt{1 - \frac{p_n^2}{n^2}}}}.$$

Периметр описанного n -угольника q_n выражается через периметр вписанного n -угольника по формуле

$$q_n = \frac{p_n}{\sqrt{1 - \frac{p_n^2}{n^2}}}.$$

Эти формулы лежат в основе одного из методов вычисления отношения длины окружности к диаметру (число π). Метод заключается в следующем.

Возьмем окружность радиуса 0,5. Ее длина равна π . Построим вписанный шестиугольник. Его периметр будет равен 3. Периметр описанного шестиугольника будет равен $2\sqrt{3}$. Будем теперь удваивать количество сто-

рон вписанного и описанного n -угольников, вычисляя их периметры по приведенным выше формулам. Очевидно, при увеличении числа сторон периметры многоугольников стремятся к одной и той же величине, равной длине окружности. Таким образом, сделав достаточно много удвоений, можно как угодно близко приблизиться к значению числа π .

Составить программу, реализующую данный метод вычисления числа π .

199. Составить программу, вычисляющую последовательность чисел Фибоначчи:

$$x_0 = 0, x_1 = 1, x_n = x_{n-1} + x_{n-2}.$$

200. Составить программу, вычисляющую последовательность:

$$x_1 = 1, x_2 = 1, x_3 = 1, x_n = x_{n-1} + x_{n-3}.$$

201. Дано натуральное число n . Вычислить:

$$S = 1 \cdot 2 + 2 \cdot 3 \cdot 4 + \dots + n \cdot (n + 1) \cdot \dots \cdot 2n.$$

202. Сколько ударов сделают часы в течение суток, если они отбивают только целое число часов? Если каждые полчаса делают удар?

203. Дана последовательность:

$$a_1 = 1, a_2 = 3, a_n = a_{n-1} - 2a_{n-2}.$$

Найти:

- первый элемент, больший 100;
- сумму первых 15 членов;
- первые 10 положительных членов;
- наибольший из первых 20 членов.

204. Леспромхоз ведет заготовку деловой древесины. Ее первоначальный объем на территории леспромхоза был равен 120 000 куб. м. Ежегодный естественный прирост составляет 5,5 %. Годовой план заготовки древесины — 9500 куб. м. Какой объем деловой древесины на данной территории будет через год, через два года и т. д. — до тех пор, пока этот объем не станет меньше критического значения (23 000 куб. м)? Сколько лет можно вести заготовку древесины на этом участке?

205. Для производства вакцины на заводе планируется выращивать культуру бактерий. Известно, что если масса бактерий — x , то через день она увеличится на $(a - bx)$, где коэффициенты a и b зависят от вида бактерий. Завод ежедневно будет забирать для нужд производства m бактерий. Как изменится масса бактерий через 1, 2, 3 ... n дней?

206. Дана последовательность, составленная из целых неотрицательных чисел. Составить программу нахождения n -го члена последовательности:

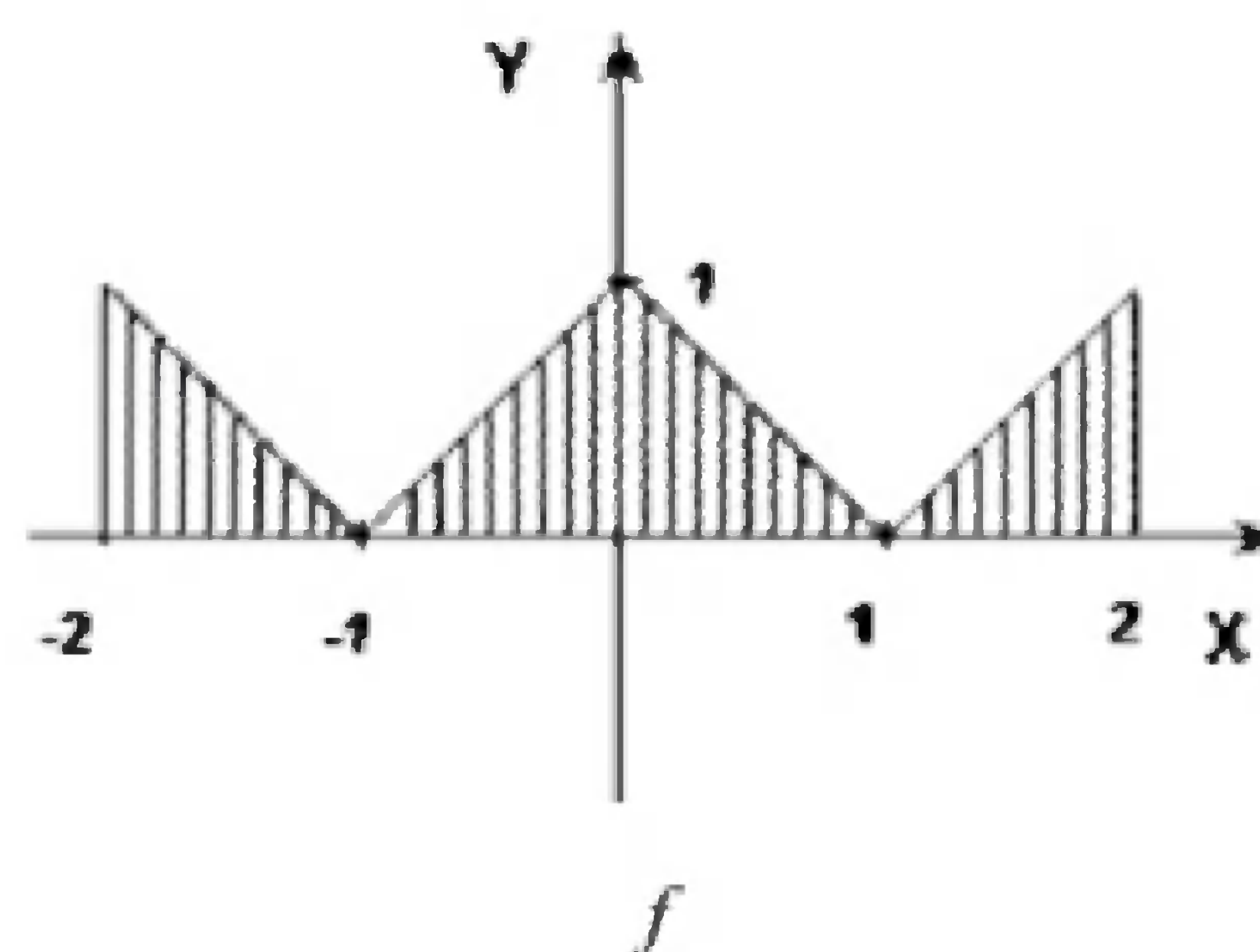
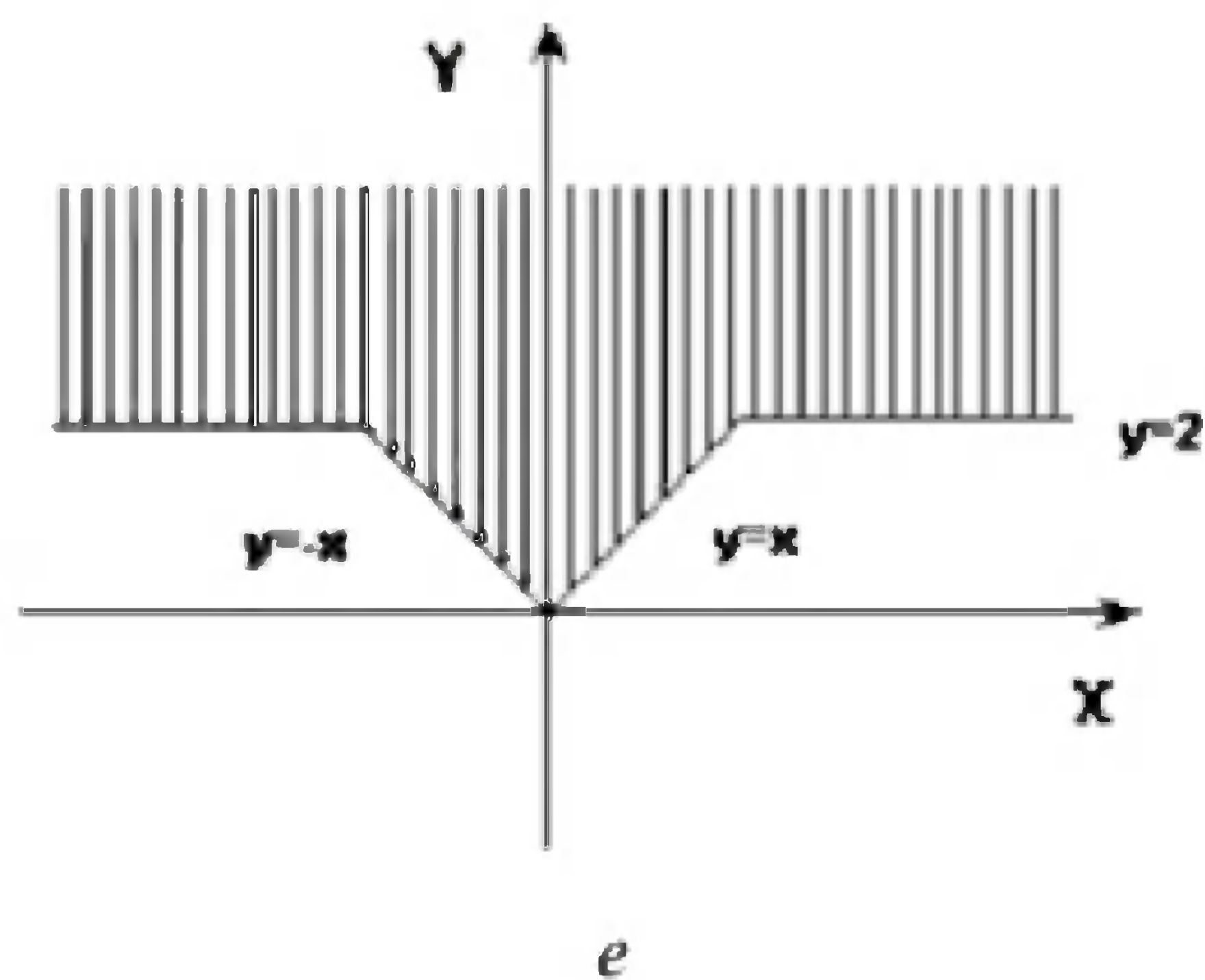
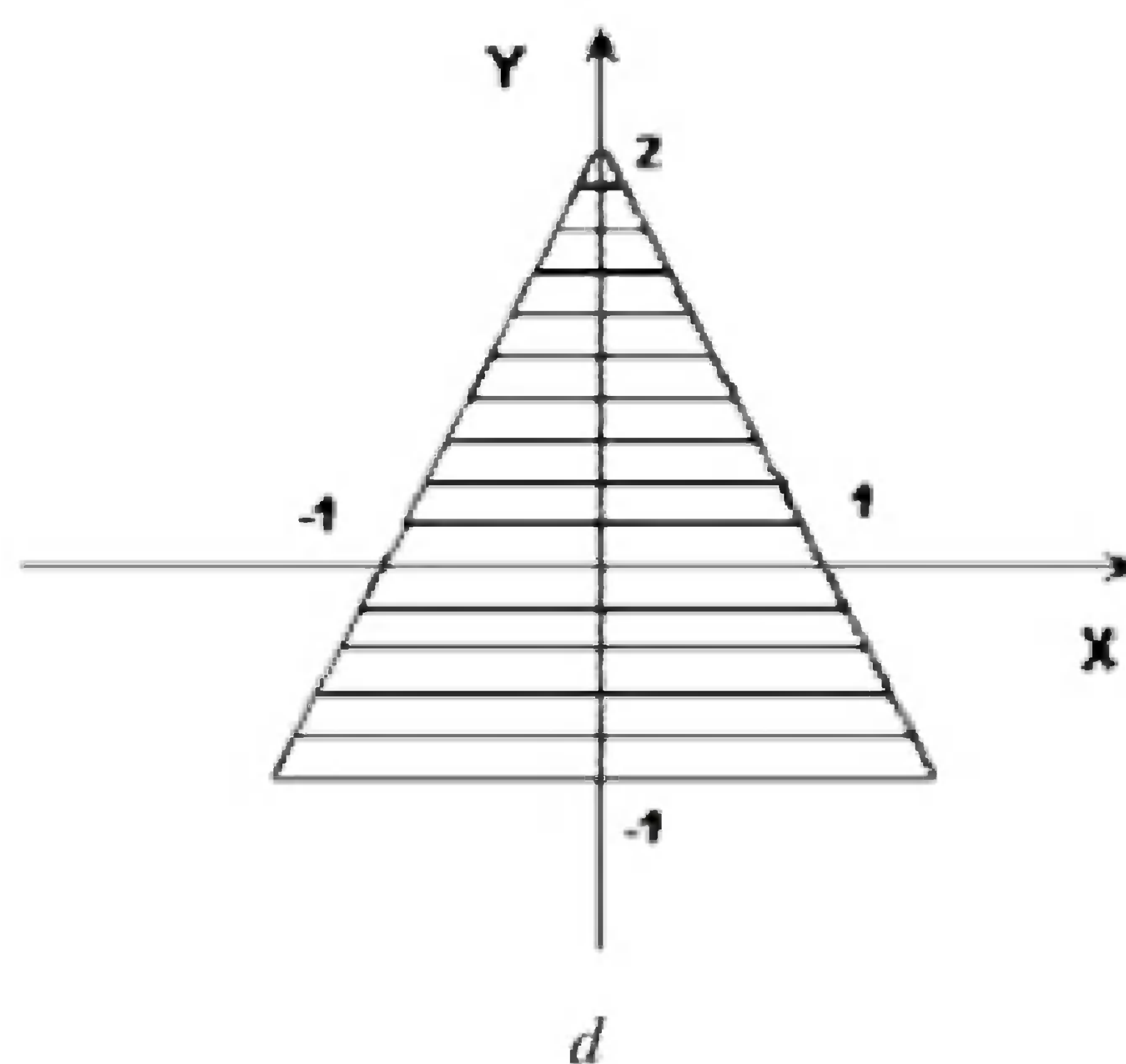
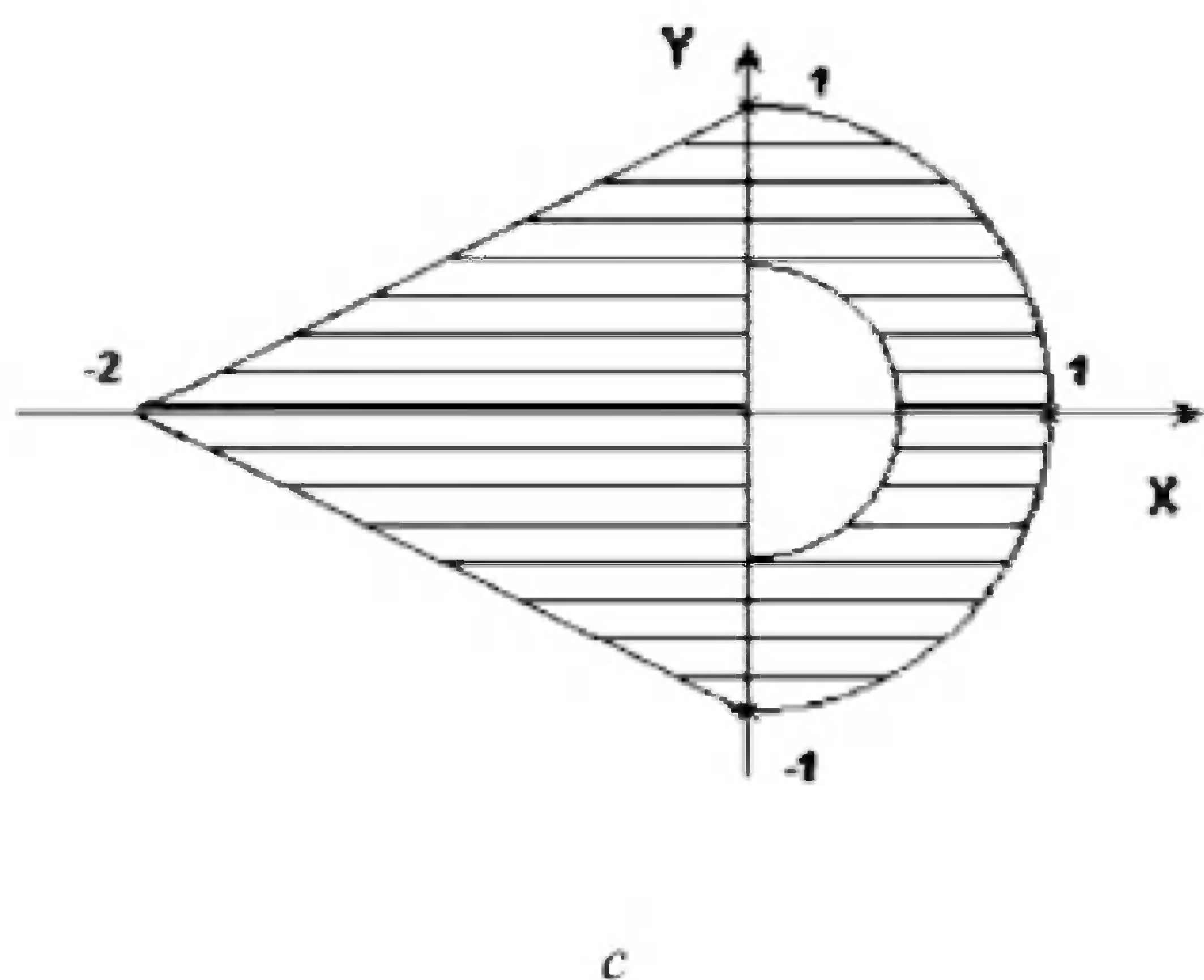
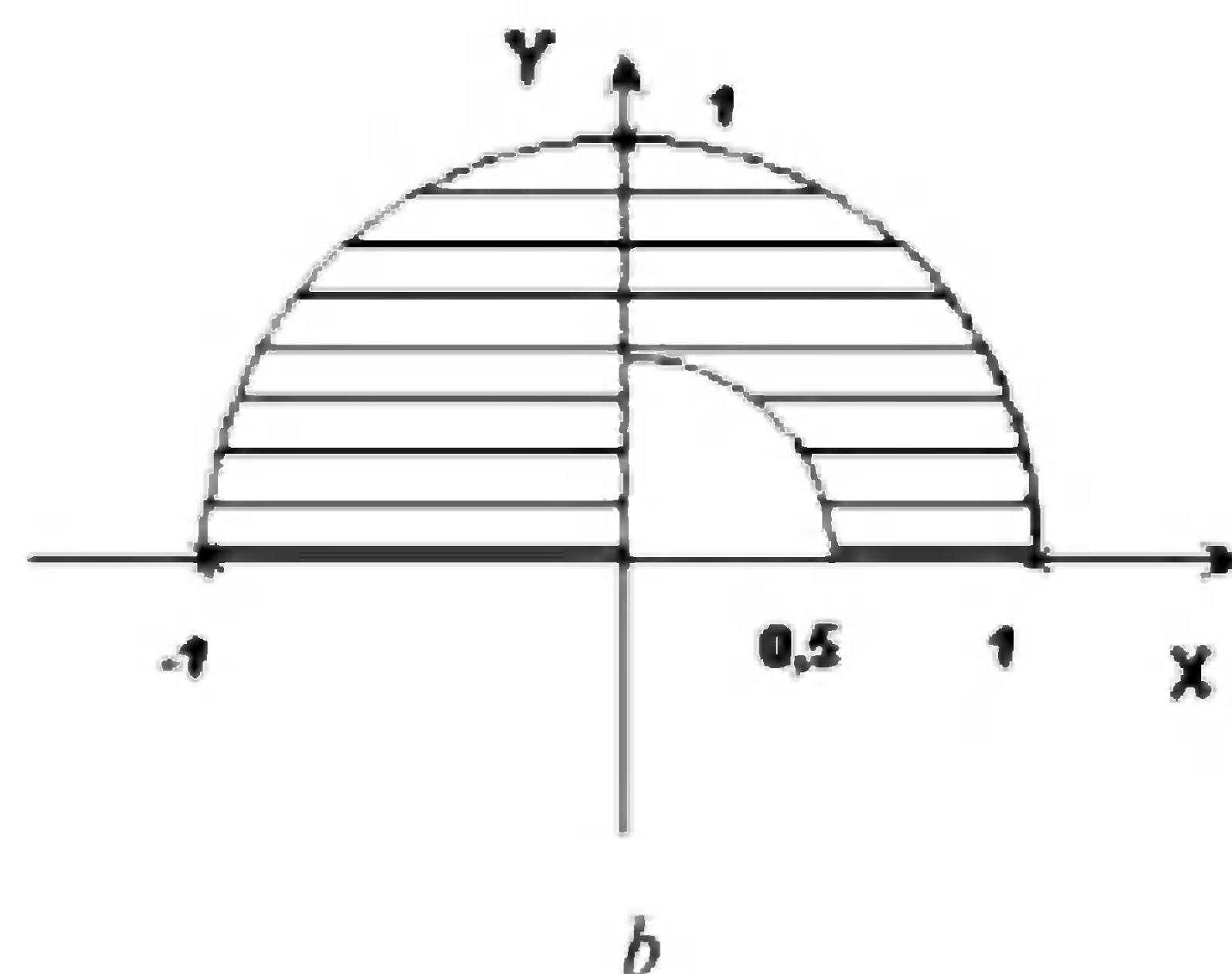
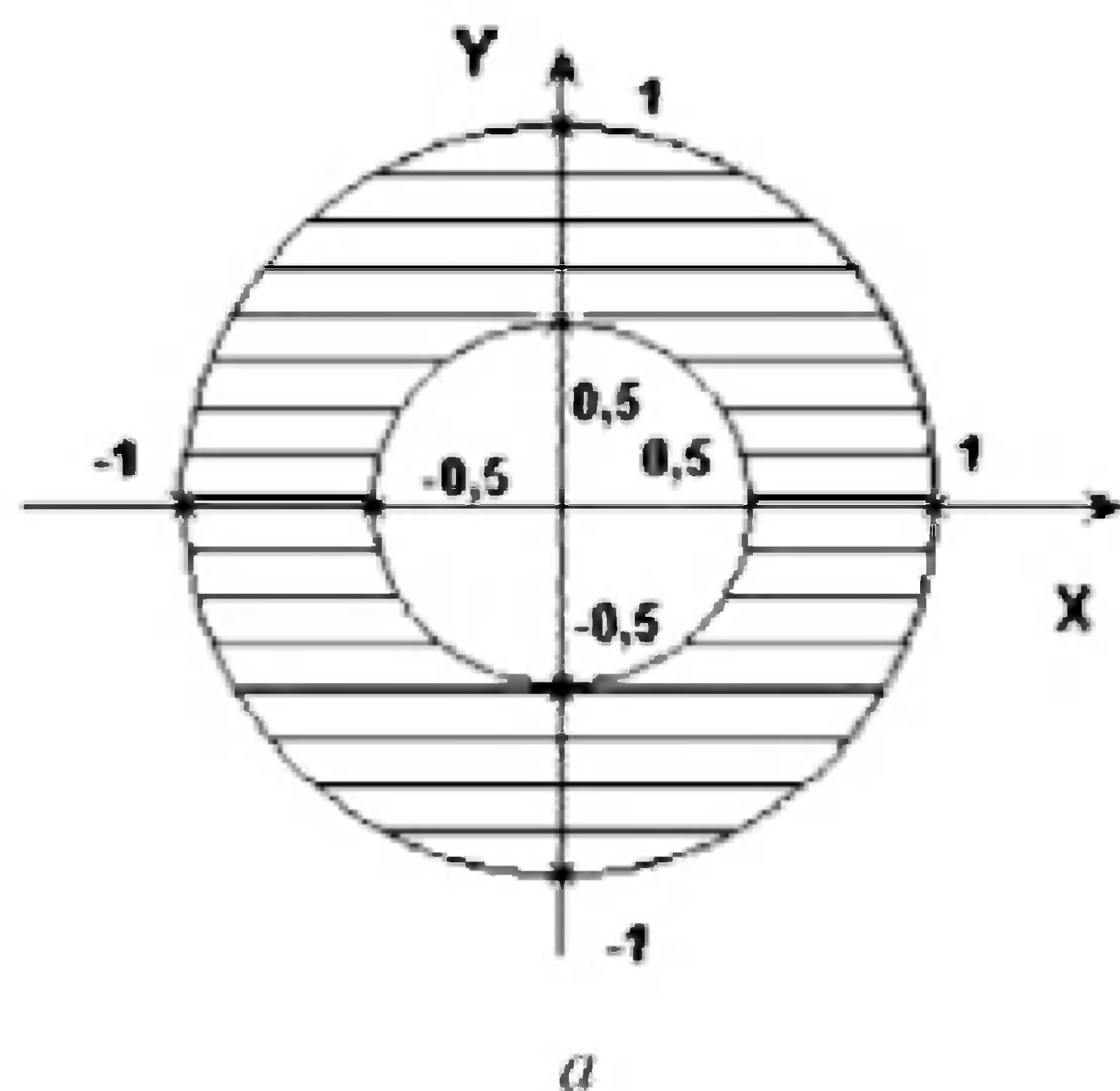
а) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 ...

б) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 11, 12, 13, 14, 15, 16, 17, 18, 19, 0, 111, 112, 113, 114, 115, 116, 117, 118, 119, 0 ...

в) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 11, 12, 13, 14, 15, 16, 17, 18, 19, 0, 21, 22, 23, 24, 25, 26, 27, 28, 29, 0 ...

ХІІІ. Логические переменные

207. Даны действительные числа x , y . Определить, принадлежит ли точка с координатами (x, y) заштрихованной части плоскости (рис. а — ф):



208. Равнобедренный треугольник задан длинами своих сторон. Вывести на экран основание треугольника.

209. Составить программу решения уравнения вида $Ax^2 + Bx + C = 0$.
210. Задана окружность радиуса R с центром в начале координат и прямая $y = kx + b$.
Определить, имеет ли место пересечение в 1-й четверти.
211. Даны углы четырехугольника $ABCD$. Является ли данный четырехугольник параллелограммом, трапецией, ни тем, ни другим.
212. Даны длины смежных сторон параллелограмма и угол между ними. Дать параллелограмму определение: квадрат, прямоугольник (не квадрат), ромб, параллелограмм.
213. Треугольник задан длинами своих сторон. Дать ему определение: равносторонний, равнобедренный, разносторонний.
214. Треугольник задан координатами своих вершин. Дать ему определение: прямоугольный, остроугольный, тупоугольный.
215. Выяснить, лежат ли на одной прямой три точки, заданные своими координатами.
216. Прямая $y = k \cdot x + b$ делит координатную плоскость на две полуплоскости. Выяснить, в одной ли полуплоскости лежат точки с координатами (x_1, y_1) и (x_2, y_2) .
217. Треугольник задан координатами своих вершин. Выяснить, принадлежит ли заданная точка этому треугольнику.
218. Многоугольник задан координатами своих вершин. Определить, является ли он выпуклым.
219. Два треугольника заданы координатами своих вершин. Определить характер их расположения: первый треугольник лежит внутри второго, второй треугольник лежит внутри первого, треугольники пересекаются, треугольники не имеют общих точек.
220. Два отрезка заданы координатами своих вершин. Определить, пересекаются ли они.
221. Треугольник задан координатами своих вершин. Определить, пересекают ли его стороны оси координат.
222. Два треугольника заданы координатами своих вершин. Определить, равны ли они.
223. Два треугольника заданы координатами своих вершин. Определить, подобны ли они.
224. Даны три числа. Определить, являются ли они частью арифметической прогрессии, геометрической прогрессии, ни то, ни другое.
225. Даны три числа. Если среди них есть пара противоположных, вывести третье число, в противном случае вывести слово «нет».
226. Даны площадь квадрата и площадь круга. Выяснить, поместится ли круг в квадрате, поместится ли квадрат в круге?
227. Даны действительные положительные числа a, b, c, d . Выяснить, можно ли прямоугольник со сторонами a, b поместить внутри прямоугольника со сторонами c, d так, чтобы каждая из сторон одного прямоугольника была параллельна или перпендикулярна сторонам другого прямоугольника.
228. Кирпич имеет размеры a, b, c . Выяснить, пройдет ли этот кирпич в прямоугольное отверстие со сторонами x, y . Просовывать кирпич в отвер-

ствие разрешается только так, чтобы каждое из его ребер было параллельно или перпендикулярно каждой из сторон отверстия.

229. Через иллюминатор затонувшего корабля необходимо вытащить сундук с драгоценностями. Возможно ли это, если иллюминатор представляет собой круг радиуса R , а сундук — прямоугольный параллелепипед со сторонами a , b , c ?

230. Двенадцать человек несут 12 хлебов. Каждый мужчина несет по 2 хлеба, каждая женщина несет по половине хлеба, каждый ребенок несет по четверти хлеба. Сколько было мужчин, женщин и детей?

XIV. Решение логических задач

231. В соревнованиях по плаванию участвуют Наташа, Вера, Света, Люда. Болельщики высказали следующие предположения:

первой будет Наташа, Вера — второй;

первой будет Света, Люда — третьей;

Света будет второй, Вера — третьей.

По окончании соревнований выяснилось, что каждый из болельщиков в одном из двух своих предположений оказался прав, в другом — неправ. Кто из девушек оказался призерами и какие места они заняли?

232. В соревнованиях по гимнастике участвуют Алла, Валя, Сима и Даша. Болельщики высказали предположения о возможных победителях:

первой будет Сима, второй — Валя;

второй будет Сима, Даша — третьей;

Алла будет второй, Даша — четвертой.

По окончании соревнований оказалось, что в каждом предположении только одно из высказываний истинно, другое ложно. Какое место на соревнованиях заняла каждая из девушек, если все они оказались на разных местах?

233. Пятеро друзей решили записаться в кружок любителей логических задач: Андрей, Борис, Виктор, Григорий, Дмитрий. Но руководитель кружка предложил им выдержать вступительный экзамен: «Вы должны приходить к нам по возможности больше вечеров, но в разных сочетаниях, соблюдая следующие условия:

если Андрей приходит вместе с Дмитрием, то Борис должен присутствовать;

если Дмитрий отсутствует, то Борис должен быть, а Виктор пусть находит;

Андрей и Виктор не могут одновременно ни присутствовать, ни отсутствовать;

если придет Дмитрий, то Григорий пусть не приходит;

если Борис отсутствует, то Дмитрий должен присутствовать, но в том случае, если не присутствует Виктор. Если же Виктор присутствует, при отсутствии Бориса, то Дмитрий приходить не должен, а Григорий должен прийти.

Сколько вечеров и в каком составе друзья могли прийти?

234. Решить задачу 174 при измененном последнем условии: если Борис отсутствует, то Дмитрий должен присутствовать, но только в том случае, если не присутствует Виктор. Если же Виктор присутствует (независимо от присутствия или отсутствия Бориса), то Дмитрий приходить не должен, а Григорий должен прийти.

235. Четыре ученика, Витя, Петя, Юра, Сережа, заняли в соревнованиях 4 первых места. На вопрос, какие места они заняли, были даны ответы:

Петя на 2-м месте, Витя на 3-м;

Сергей на 2-м, Петя на 1-м;

Юра на 2-м, Витя на 4-м.

Указать, кто какое место занял, если в каждом ответе правильна лишь одна часть.

236. Четверо ребят, Алеша, Боря, Ваня, Гриша, соревновались в беге. После этого на вопрос о распределении мест они ответили так. Алеша: «Я не был ни первым, ни последним». Ваня: «Я был первым». Боря: «Я не был последним». Гриша: «Я был последним». Трое из этих ответов правильные, а один неверный. Как распределились места?

237. Четыре приятеля, Андрей, Борис, Виктор, Григорий, живут в разных комнатах общежития. На вопрос, где они живут, трое дали по два ответа, один из которых истинный, другой ложный. Андрей: «Я живу в первой комнате, Григорий — во второй». Борис: «Я живу в третьей комнате, Андрей — во второй». Виктор: «Я живу во второй комнате, Борис — в четвертой». Кто в какой комнате живет?

238. В финал шахматного турнира вышли Аркадий, Володя, Саша. Перед финалом один болельщик сказал, что первое место займет Аркадий, второй — что Сеня не будет последним, а третий — что Володя не будет на первом месте. После игр оказалось, что один болельщик ошибся, а два других угадали. Как закончился финал, если никакие два участника не заняли одно и то же место?

239. Решить задачу 177 при условии, что угадал только один болельщик, а два других ошиблись.

240. N хотел пригласить в гости A, B, C, D, E, F, G, H. При этом он столкнулся со следующими трудностями:

A никогда не придет, если пригласить B или C или если одновременно придут D и E;

D придет только в том случае, если будет приглашен и E;

E не примет приглашения, если придет B;

F наносит визиты только в сопровождении G;

H не будет возражать против присутствия F только в том случае, если будет приглашен и A;

если не будет приглашен F, то H будет против приглашения E;

чтобы пришел G, необходимо пригласить D или H;

G откажется от приглашения, если пригласят E без A, а также в случае приглашения B или C.

Кого мог пригласить N?

241. На собрании. Коллектив некоторого учреждения периодически собирается на собрания, причем замечено, что если Борис не пришел на соб-

рание, то отсутствует и Алексей; если Борис пришел на собрание, то присутствуют Алексей и Валерий.

Спрашивается, обязательно ли присутствует на собрании Алексей, если Валерий отсутствует; присутствует ли на собрании Валерий, если Алексей отсутствует?

242. На автоматизированном участке цеха стоит 5 станков, действия которых скоординированы следующим образом. Если работают 1-й и 3-й станки, то 4-й не работает при условии, что подключен 5-й станок. Если же 1-й станок подключен без 3-го или выключен 5-й станок, то 4-й обязательно включен. Если 5-й станок работает вместе со 2-м при включенном 1-м станке, то включен 3-й станок. Если выключены 2-й или 5-й станки, то одновременно выключен и 4-й. Мы наблюдаем работу 1-го и 4-го станков. Что можно сказать о состоянии остальных станков? Можно ли остановить для ремонта одновременно 3-й и 4-й станки, оставив хотя бы один из других станков включенными?

243. Четыре человека взялись выполнять работу маляра, слесаря, кузнеца и штукатур — каждый будет делать что-то одно. Выяснилось, что Андреев не будет маляром и не будет слесарем, Владимиров не будет кузнецом и не будет маляром, Сергеев не будет слесарем и не будет маляром, Дмитриев не будет слесарем и не будет кузнецом. Известно также, что если Андреев не будет кузнецом, то Дмитриев не будет маляром. Кто какую работу выполнял?

244. В составе экспедиции должно быть 6 специалистов: биолог, врач, синоптик, гидролог, механик, радист. Имеется 8 кандидатов, из которых нужно выбрать 6 участников экспедиции. Условные имена претендентов: А, В, С, D, E, F, G, H. Обязанности биолога могут выполнять E и G, врача — А и D, синоптика — F и G, гидролога — В и F, радиста — С и D, механика — С и H. Предусмотрено, что в экспедиции каждый из них будет выполнять только одну обязанность.

Кого и в какой должности следует включить в экспедицию, если F не может ехать без В, D — без H и без С, С не может ехать вместе с G, А вместе с В?

245. В одном проектом институте имеется три отдела: А, В, С. При обсуждении одного проекта руководство института условилось придерживаться следующего порядка:

а) если в обсуждении не принимает участия отдел В, то отдел А также не участвует в обсуждении;

б) если в обсуждении проекта принимает участие отдел В, то отделы А и С также участвуют в обсуждении.

Должен ли при этих условиях отдел С принимать участие в обсуждении проекта, если в обсуждении участвует отдел А?

246. Миша решил поступать в институт и послал домой три сообщения: если я сдам математику, то информатику я сдам только при условии, что не завалю диктант;

не может быть, чтобы я завалил и диктант, и математику;

достаточное условие завала по информатике — это двойка по диктанту.

После сдачи экзаменов оказалось, что из трех Мишиных сообщений только одно было ложным. Как Миша сдал экзамены?

247. На время ремонта общежития комендант был вынужден переселять студентов. На одну из новых комнат претендовали восемь человек, а поселить в ней можно было только четверых. Комендант стал расспрашивать студентов.

Андрей согласен на любых соседей. Борис без Кости не переселится, но Костя не хочет жить в одной комнате с Василием. Василий согласен жить с кем угодно. Дима не будет переселяться без Юры. Федя не будет жить без Гриши в одной комнате с Димой, а без Димы не будет жить в одной комнате с Костей. Гриша не хочет, чтобы его соседями были и Борис и Костя вместе, а кроме того, он не желает жить в одной комнате ни с Андреем, ни с Василием. Чтобы Юра дал согласие переехать в новую комнату, надо, чтобы туда же перебрались либо Борис, либо Федя. Кроме того, Юра не будет жить в одной комнате с Костей, если туда не переедет Гриша, и не желает жить в одной комнате ни с Андреем, ни с Василием.

Комендант оказался смекалистым человеком и учел все пожелания. Каким образом?

ОГЛАВЛЕНИЕ

Глава I. Решение задач на компьютере

§ 1. Компьютер — машина для обработки информации	4
§ 2. Алгоритм решения задачи	6
§ 3. Исполнитель алгоритма	7
§ 4. Алгоритм и программа	9
§ 5. Ввод данных	13
§ 6. Постановка задачи	15

Глава II. Основные понятия программирования

§ 1. Программа и операторы	18
§ 2. Переменные и идентификаторы	19
§ 3. Арифметическое выражение	21
§ 4. Оператор присваивания	22
§ 5. Ввод данных с клавиатуры	24
§ 6. Чтение данных	25
§ 7. Вывод данных на экран	26
§ 8. Координатный вывод	28
§ 9. Служебные слова и операнды	30
§ 10. Некоторые сведения из геометрии	32
§ 11. Тестирование программ	34
§ 12. Блок-схема алгоритма	36
§ 13. Правила составления алгоритма	37

Глава III. Ветвления и циклы

§ 1. Условный оператор	40
§ 2. Оператор безусловного перехода	42
§ 3. Вложенные условные конструкции	44
§ 4. Логические операции	46
§ 5. Циклы	48
§ 6. Блок-схема оператора цикла	51
§ 7. Массивы	52
§ 8. Заполнение массива	55

Глава IV. Графика

§ 1. Графический режим QBASIC	57
§ 2. Установка системы координат	58
§ 3. Точки и линии	59
§ 4. Прямоугольники	60
§ 5. Форматы оператора LINE. Установка текущего цвета	61
§ 6. Окружности и дуги	62
§ 7. Раскрашивание рисунка	64

§ 8. Экранная система координат. Окно просмотра	64
§ 9. Алгоритм построения графиков функций	66
§ 10. Функция, определяемая пользователем	68
§ 11. Построение графика функции с особыми точками	69

Глава V. Вычисления по рекуррентным формулам

§ 1. Арифметическая прогрессия	74
§ 2. Геометрическая прогрессия	77
§ 3. Алгоритм вычисления суммы	79
§ 4. Циклы типа «До» и типа «Пока»	80
§ 5. Числовые последовательности.	83
§ 6. Факториал	85
§ 7. Сходящиеся числовые последовательности. Предел числовой последовательности.	86
§ 8. Сумма членов бесконечной числовой последовательности. Числовой ряд.	89
§ 9. Вывод рекуррентной формулы из формулы общего члена числовой последовательности.	93
§ 10. «Золотое сечение». Цепные дроби	98

Глава VI. Задачи на проверку условий

§ 1. Логические выражения	101
§ 2. Логические переменные.	102
§ 3. Логические операции	104
§ 4. Перевод логических выражений с русского языка на язык математики.	106
§ 5. Составные высказывания и логические формулы	108
§ 6. Еще две логические операции.	109
§ 7. Решение логических задач методом составления характеристического уравнения	111
§ 8. Импликация	113

Задачи

I. Алгоритм и программа	116
II. Оператор присваивания	118
III. Линейные программы	120
IV. Условный оператор	121
V. Циклы и массивы	124
VI. Графика	124
VII. Арифметическая и геометрическая прогрессии.	127
VIII. Вычисление сумм и произведений	128
IX. Циклы типа «До» и типа «Пока».	129
X. Вычисления с заданной точностью	131
XI. Дроби и радикалы.	132
XII. Задачи на использование рекуррентных формул	133
XIII. Логические переменные	136
XIV. Решение логических задач.	138

Издательство «СОЛОН-Р»
129337, Москва, а/я 5
Телефоны:
(095) 254-44-10, 252-36-96, 252-25-21
E-mail: Solon.Pub@relcom.ru

*Приглашаем к сотрудничеству авторов — специалистов
в области компьютерных технологий
E-mail: Solon-Avtor@coba.ru*

ООО Издательство «СОЛОН-Р»
ЛР № 066584 от 14.05.99
Москва, ул. Тверская, д. 10, стр. 1, ком. 522
Формат 70×100/16. Объем 9 п. л. Тираж 4000
АООТ «ПОЛИТЕХ-4»
Москва, Б. Переяславская, 46
Заказ №