ЕГЭ за 30 дней

О. Б. Богомолова

ИНФОРМАТИКА

ЭКСПРЕСС-РЕПЕТИТОР

О. Б. БОГОМОЛОВА

ЕГЭ за 30 дней

ИНФОРМАТИКА

ЭКСПРЕСС-РЕПЕТИТОР

АСТ • Астрель
Москва

УДК 373:3
ББК 60я72
Б74

Богомолова, Ольга Борисовна

Б74 Информатика : ЕГЭ за 30 дней : экспресс-репетитор / О. Б. Богомолова. — Москва: АСТ, Астрель, 2014. — 446, [2] с. — (Единый государственный экзамен).

ISBN 978-5-17-080257-9 (ООО «Издательство АСТ»)

ISBN 978-5-271-46341-9 (ООО «Издательство Астрель»)

Пособие поможет школьнику подготовиться к Единому государственному экзамену, освежить в памяти необходимый теоретический материал, а также ознакомиться с принципами решения типовых задач ЕГЭ, предлагавшихся в разные годы. Отличительная особенность данного пособия в том, что при разборе задач не только показано, как они решаются, но и объяснено, почему задачи решаются именно так. Способы решения многократно апробированы автором в собственной педагогической работе.

Для школьников 10—11 классов, учителей информатики и методистов.

УДК 373:3
ББК 60я72

ISBN 978-5-17-080257-9 (ООО «Издательство АСТ»)
ISBN 978-5-271-46341-9 (ООО «Издательство Астрель»)
ISBN 978-985-18-2792-9 (ООО «Харвест»)

© Богомолова О.Б.
© ООО «Издательство АСТ»

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Единый государственный экзамен (ЕГЭ) в настоящее время признан в качестве основной формы объективной оценки качества подготовки школьников, освоивших образовательные программы среднего (полного) общего образования. Результаты ЕГЭ являются результатами одновременно государственной (итоговой) аттестации для образовательных учреждений среднего (полного) общего образования и вступительных экзаменов по соответствующим общеобразовательным предметам — для образовательных учреждений среднего и высшего профессионального образования.

В настоящее время является обязательной сдача ЕГЭ по русскому языку и математике, а ещё по 12 предметам сдача единого государственного экзамена проводится по выбору учащегося. Полученные при этом баллы являются основным критерием зачисления абитуриента на выбранную им специальность профессионального образования.

Информатика (прежнее название предмета — «Информатика и информационно-коммуникационные технологии» — действовало до введения в 2012 г. нового Федерального государственного образовательного стандарта) относится к 12 предметам, сдача ЕГЭ по которым производится учащимися на добровольной основе. Однако перечень вузов и ссузов, требующих наличия свидетельства об успешной сдаче ЕГЭ по информатике для поступления на основные специальности, постоянно растет. Возрастает год от года и сложность заданий, предлагаемых на ЕГЭ по информатике.

Поэтому подготовка к ЕГЭ является высоко актуальной задачей как для самих учащихся старшей школы, так и для учителей информатики.

Наилучшей стратегией такой подготовки является, конечно же, системное и целенаправленное формирование основных информационных компетенций школьников, отработка решения разнообразных заданий и выработка навыков работы с основными средствами ИКТ по всем без исключения изучаемым темам курса. Однако нынешние реалии, к сожалению, требуют принимать в расчет и недостаточное количество часов, отпущенных на изучение предмета «Информатика» федеральными учебными планами, и практическое отсутствие задачников-практикумов, поддерживающих не фрагментарное ознакомление с отдельными темами, а плотное прохождение всего курса.

Эти сложности заставляют выстраивать стратегию подготовки к ЕГЭ прежде всего на базе индивидуальной работы школьников с учебным материалом под руководством и при активной консультационной поддержке со стороны учителя. То есть учитель вначале проводит для класса разбор решения типовых заданий по данной теме, объясняя, почему ту или иную задачу лучше всего решать именно так, а затем решение подобных задач отрабатывается в ходе массивной практической работы с последующим контролем (ручным или автоматизированным) на базе нескольких вариантов заданий, максимально раскрывающих возможные вариации их условий.

Подготовиться к сдаче ЕГЭ на 90—100 баллов — задача достаточно сложная, требующая обоюдной заинтересованности и обоюдного напряжения сил как учащегося, так и учителя, но, как показывает педагогическая практика автора данной книги, вполне выполнимая.

Предлагаемое вашему вниманию учебное пособие основано на результатах многолетней педагогической практики автора. Учтён анализ тематики заданий ЕГЭ за последние несколько лет; разобраны принципы решения наиболее характерных вариантов заданий, предлагавшихся на ЕГЭ. При этом пособие нацелено прежде всего на самостоятельную (в том числе под контролем со стороны учителя) индивидуальную подготовку школьника.

В соответствии с названием пособия такая подготовка в «экспресс-режиме» действительно может быть проведена в течение 30 дней, но — и это следует понимать правильно — в расчёте на работу с материалом в течение нескольких часов! Поэтому если речь идёт об использовании данного пособия в виде классно-урочной формы обучения, то общее время подготовки к ЕГЭ составит не менее 72 часов (темп прохождения конкретных тем при этом может определять сам учитель).

Работу с пособием рекомендуется начинать с выполнения заданий входного теста. Результаты решения задач необходимо сверить с предлагаемыми в конце входного теста ответами; номера занятий (разделов пособия), приведённые в последней графе таблицы ответов, позволят определить, на какие именно темы курса нужно обратить повышенное внимание.

Далее работа с материалом пособия может проводиться как по всем темам подряд, так и (если речь идет о «экспресс-коррекции» знаний) только по темам, вызвавшим трудности при выполнении входного теста, хотя последний вариант учебной подготовки не гарантирует, что учащийся не встретит трудностей с задачами на пропущенные темы в реальном ЕГЭ, а потому является вспомогательным. Каждое занятие включает краткий теоретический материал в объёме, необходимом для решения рассматриваемых задач, собственно разбор решений, а также небольшую подборку задач по данной теме для самостоятельного решения (с ответами для самоконтроля).

После завершения изучения материала пособия рекомендуется выполнить итоговый тест. Сверяя свои результаты с приведёнными в конце него ответами, по последней графе таблицы ответов (как и во входном тестировании) можно определить «проблемные» темы, изучение которых требуется повторить.

Входное тестирование

Выполните предложенные задания и сравните свои результаты с приведёнными после теста правильными ответами. В таблице правильных ответов указаны номера занятий, которые необходимо изучить в случае неправильного или неизвестного решения той или иной задачи.

Тест

1*[1]. Автоматическое устройство осуществило перекодировку информационного сообщения на русском языке, первоначально записанного в 16-битном коде Unicode, в 8-битную кодировку КОИ-8. При этом информационное сообщение уменьшилось на 480 бит. Какова длина сообщения в символах?

1) 30 2) 60 3) 120 4) 480

2. В коробке лежат 64 цветных карандаша. Сообщение о том, что достали синий карандаш, несёт 4 бита информации. Сколько синих карандашей было в коробке?

3. Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из трёх состояний («включено», «выключено» или «мигает»). Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 20 различных сигналов?

1) 6 2) 5 3) 3 4) 4

4*. Для регистрации на сайте некоторой страны пользователю требуется придумать пароль. Длина пароля — ровно 11 символов. В качестве символов используются десятичные цифры и 12 различных букв местного алфавита, причём все буквы используются в двух начертаниях: как строчные, так и заглавные (регистр буквы имеет значение!).

Под хранение каждого такого пароля на компьютере отводится минимально возможное и одинаковое целое количество байтов, при этом используется посимвольное кодирование и все символы кодируются одинаковым и минимально возможным количеством битов.

Определите объём памяти, который занимает хранение 60 паролей.

1) 540 байт 2) 600 байт 3) 660 байт 4) 720 байт

5*. Для 5 букв русского алфавита заданы их двоичные коды (для некоторых букв — из двух бит, для некоторых — из трёх). Эти коды представлены в таблице:

В	К	А	Р	Д
000	11	01	001	10

Из четырёх полученных сообщений в этой кодировке, только одно прошло без ошибки и может быть корректно декодировано. Найдите его:

1) 110100000100110011
2) 111010000010010011
3) 110100001001100111
4) 110110000100110010

[1] Задания, отмеченные «*» взяты с сайта www.fipi.ru из демонстрационных вариантов по информатике разных лет.

6*. Известно, что длительность непрерывного подключения к сети Интернет с помощью модема для некоторых АТС не превышает 10 минут. Определите максимальный размер файла (в Килобайтах), который может быть передан за время такого подключения, если модем передаёт информацию в среднем со скоростью 32 Килобит/с?

7*. У вас есть высокоскоростной доступ в Интернет по спутниковому каналу со скоростью передачи данных от сервера к вам, равной 4 Мбит/с. В обратном направлении (с вашего компьютера к серверу) информация (запросы) передаётся через сотовый телефон, подключённый в качестве модема, со скоростью не более 128 Кбит/с. Вам нужно скачать файл объёмом 16 Мбайт. Информация передаётся с сервера на ваш компьютер фрагментами не более 1 Мбайта. Для получения каждого такого фрагмента ваш компьютер должен сначала передать серверу запрос суммарным объёмом 8 Кбайт. За какое минимально возможное число секунд вы сможете получить весь файл?

8. Между городами A, B, C, D, E, F проложены дороги, длина которых приведена в таблице. Отсутствие числа означает, что дороги между соответствующими пунктами нет. Передвигаться можно только по этим дорогам.

	A	B	C	D	E	F
A				1	4	3
B			4			5
C		4		2	1	
D	1		2			2
4	4		1			
F	3	5		2		

Какова длина кратчайшего пути между городами A и B?

1) 6 3) 8
2) 7 4) 9

9*. Сколько единиц в двоичной записи десятичного числа 194,5?

1) 5
2) 2
3) 3
4) 4

10*. Дано $A = A7_{16}$, $B = 251_8$. Какое из чисел C, записанных в двоичной системе, отвечает условию $A < C < B$?

1) 10101100_2
2) 10101010_2
3) 10101011_2
4) 10101000_2

11*. В системе счисления с некоторым основанием десятичное число 49 записывается в виде 100. Укажите это основание.

12*. Укажите через запятую в порядке возрастания все десятичные числа, не превосходящие 25, запись которых в системе счисления с основанием четыре оканчивается на 11.

13. Все 5-буквенные слова, составленные из букв М, И, Р, записаны в алфавитном порядке. Вот начало списка:

1. ИИИМИ
2. ИИИММ
3. ИИИМР
4. ИИИРИ
5. ИИИРМ

.....

Какое слово стоит на 123-м месте в списке?

14*. Строки (цепочки латинских букв) создаются по следующему правилу.

Первая строка состоит из одного символа — латинской буквы «A». Каждая из последующих цепочек создаётся такими действиями: в очередную строку сначала записывается буква, чей порядковый номер в алфавите соответствует номеру строки (на i-м шаге пишется «i»-я буква алфавита), к ней слева дважды подряд приписывается предыдущая строка.

Вот первые 4 строки, созданные по этому правилу:

(1) A

(2) AAB

(3) AABAABC

(4) AABAABCAABAABCD

Латинский алфавит (для справки): ABCDEFGHIJKLMNOPQRSTUVWXYZ

Запишите шесть символов подряд, стоящие в седьмой строке со 117-го по 122-е место (считая слева направо).

15*. Символом F обозначено одно из указанных ниже логических выражений от трёх аргументов: X, Y, Z.

Дан фрагмент таблицы истинности выражения F:

X	Y	Z	F
0	1	1	0
1	1	1	1
0	0	1	1

Какое выражение соответствует F?

1) $X \wedge \neg Y \wedge \neg Z$
2) $\neg X \wedge \neg Y \wedge Z$
3) $\neg X \vee \neg Y \vee Z$
4) $X \vee \neg Y \vee \neg Z$

16*. Какое из приведённых имён удовлетворяет логическому условию:

$\neg$ (последняя буква гласная $\rightarrow$ первая буква согласная) $\wedge$ вторая буква согласная.

1) ИРИНА
2) АРТЁМ
3) СТЕПАН
4) МАРИЯ

17*. Для какого из указанных значений X истинно высказывание $\neg((X > 2) \rightarrow (X > 3))$?

1) 1
2) 2
3) 3
4) 4

18*. Укажите значения логических переменных K, L, M, N, при которых логическое выражение $(K \vee M) \to (M \vee \neg L \vee N)$ **ложно.**

Ответ запишите в виде строки из четырёх символов: значений переменных K, L, M и N (в указанном порядке). Так, например, строка 0101 соответствует тому, что K = 0, L = 1, M = 0, N = 1.

19*. Сколько различных решений имеет уравнение $((K \vee L) \to (L \wedge M \wedge N)) = 0$, где K, L, M, N — логические переменные?

В ответе **не нужно** перечислять все различные наборы значений K, L, M и N, при которых выполнено данное равенство. В качестве ответа Вам нужно указать количество таких наборов.

20. Сколько существует различных наборов значений логических переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, которые удовлетворяют всем перечисленным ниже условиям?

$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1$;

$(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1$;

$y_5 \to x_5 = 1$.

В ответе не нужно перечислять все различные наборы значений переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, при которых выполнена данная система равенств. В качестве ответа Вам нужно указать количество таких наборов.

21. Устройство считывает четырёхзначное восьмеричное число и строит по нему новое число по следующему алгоритму:

1) вычисляется сумма первой и второй цифр;

2) вычисляется сумма третьей и четвёртой цифр;

3) эти суммы записываются друг за другом без разделителей по убыванию.

Пример. Задано число 7145. Суммы: 7 + 1 = 10; 4 + 5 = 11. Результат: 1110.

Какое из чисел может быть результатом работы такого устройства?

1) 119 2) 1213 3) 1411 4) 1715

22. Система команд исполнителя РОБОТ, живущего в прямоугольном лабиринте на клетчатой плоскости:

вверх	вниз	влево	вправо

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

Цикл

```
ПОКА
   <условие>
      последовательность команд
КОНЕЦ ПОКА
```

выполняется, пока условие **истинно.**

В конструкции

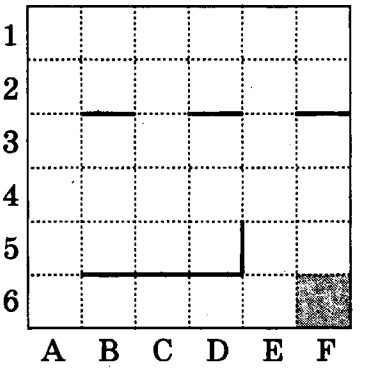
```
ЕСЛИ <условие>
   ТО команда1
   ИНАЧЕ команда2
КОНЕЦ ЕСЛИ
```

выполняется команда1 **(если** условие **истинно) или** команда2 **(если** условие **ложно).**

Если РОБОТ начнёт движение в сторону находящейся рядом с ним стены, то он разрушится и программа прервётся.

Сколько клеток лабиринта соответствуют требованию, что, начав движение в ней и выполнив предложенную программу, РОБОТ уцелеет и остановится в закрашенной клетке (клетка **F6**)?

```
НАЧАЛО
ПОКА <справа свободно ИЛИ снизу свободно>
  ПОКА <снизу свободно>
    вниз
  КОНЕЦ ПОКА
  ПОКА <справа свободно>
    вправо
  КОНЕЦ ПОКА
КОНЕЦ ПОКА
КОНЕЦ
```



1) 14 2) 17 3) 19 4) 21

23. Исполнитель Пятачок имеет две команды:

1) прибавить 1;

2) умножить на 5.

Запишите порядковые номера команд в программе для преобразования числа 1 в число 76, содержащей не более 5 строк.

Пример: 21211 соответствует программе:

умножить на 5
прибавить 1
умножить на 5
прибавить 1
прибавить 1

которая преобразует число 1 в число 32.

Если таких программ возможно более одной, то запишите любую из них.

24. Исполнитель Вычислитель имеет две команды:

1) умножить на 4;

2) поделить на 2.

Сколько различных чисел можно получить из числа 1024 с помощью программы, которая содержит ровно 10 команд?

25*. Лена забыла пароль для входа в Windows XP, но помнила алгоритм его получения из символов A153B42FB4 в строке подсказки. Если последовательность символов B4 заменить на B52 и из получившейся строки удалить все трёхзначные числа, то полученная последовательность и будет паролем:

1) ABFB52 2) AB42FB52 3) ABFB4 4) AB52FB

26. У исполнителя Квадратик две команды, которым присвоены номера:

1. прибавь 2,

2. возведи в квадрат.

Первая из них увеличивает число на экране на 2, вторая — возводит в квадрат.

Программа для исполнителя — это последовательность команд.

Сколько есть программ, которые число 4 преобразуют в число 66?

Ответ обоснуйте.

27*. В некотором каталоге хранился файл **Задача5.** После того, как в этом каталоге создали подкаталог и переместили в созданный подкаталог файл **Задача5**, полное имя файла стало **E:\Класс9\Физика\Задачник\Задача5**. Каково было полное имя этого файла до перемещения?

1) E:\Физика\Задачник\Задача5
2) E:\Физика\Задача5
3) E:\Класс9\Задачник\Задача5
4) E:\Класс9\Физика\Задача5

28*. Для групповых операций с файлами используются маски имён файлов. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которых также могут встречаться следующие символы:

Символ ? (вопросительный знак) означает ровно один произвольный символ.

Символ «*» (звёздочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность.

Определите, по какой из масок будет выбрана указанная группа файлов:

1234.xls
23.xml
234.xls
23.xml

1) *23*.?x*
2) ?23?.x??
3) ?23?.x*
4) *23*.???

29. В терминологии сетей TCP/IP маской сети называется двоичное число, определяющее, какая часть IP-адреса узла сети относится к адресу сети, а какая к адресу самого узла в этой сети. Обычно маска записывается по тем же правилам, что и IP-адрес. Адрес сети получается в результате применения поразрядной конъюнкции к заданным IP-адресу узла и маске.

По заданным IP-адресу узла и маске определите адрес сети.

IP-адрес узла: 224.23.252.131
Маска: 255.255.240.0

При записи ответа выберите из приведённых в таблице чисел четыре элемента IP-адреса и запишите в нужном порядке соответствующие им буквы без использования точек.

A	B	C	D	E	F	G	H
255	240	252	224	131	23	8	0

Пример. Пусть искомый IP-адрес 192.168.128.0 и дана таблица

A	B	C	D	E	F	G	H
128	*168*	*255*	*8*	*127*	*0*	*17*	*192*

В этом случае правильный ответ будет записан в виде HBAF.

30*. Укажите минимальный объём памяти (в килобайтах), достаточный для хранения любого растрового изображения размером 64×64 пикселя, если известно, что в изображении используется палитра из 256 цветов. Саму палитру хранить не нужно.

1) 128
2) 2
3) 256
4) 4

31. Производится двухканальная (стерео) звукозапись с частотой дискретизации 48 кГц и 24-битным разрешением. Запись длится 1 минуту, её результаты записываются в файл, сжатие данных не производится. Какое из приведённых ниже чисел наиболее близко к размеру полученного файла, выраженному в мегабайтах?

1) 0,3 2) 4 3) 16 4) 132

32*. При работе с электронной таблицей в ячейке **A1** записана формула **=D1-$D2**. Какой вид приобретёт формула, после того как ячейку **A1** скопируют в ячейку **B1**?

Примечание: символ $ в формуле обозначает абсолютную адресацию.

1) =E1-$E2 2) =E1-$D2 3) =E2-$D2 4) =D1-$E2

33*. В электронной таблице значение формулы **=СУММ(B1:B2)** равно 5. Чему равно значение ячейки **B3**, если значение формулы **=СРЗНАЧ(B1:B3)** равно 3?

1) 8 2) 2 3) 3 4) 4

34*. Три страны: Королевство Бельгия, Королевство Нидерланды и Великое Герцогство Люксембург образуют экономико-политический союз, который носит название Бенилюкс. Ниже приведён фрагмент электронной таблицы, характеризующий каждую из стран союза и союз в целом:

	A	B	C	D
1	Страна	Население (тыс. чел.)	Площадь (кв. км)	Плотность населения (чел. / кв. км)
2	Бельгия	10 415	30 528	341
3	Нидерланды	16 357	41 526	394
4	Люксембург	502	2586	194
5	Бенилюкс в целом	27 274	74 640	

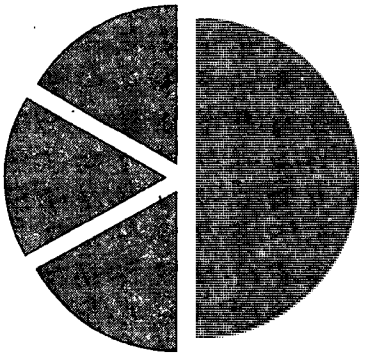
Какое значение должно стоять в ячейке D5?

1) 365 2) 929 3) 310 4) 2,74

35*. Дан фрагмент электронной таблицы:

	A	B	C	D
1	3		3	2
2	=(C1+A1)/2	=C1-D1	=A1-D1	=B1/2

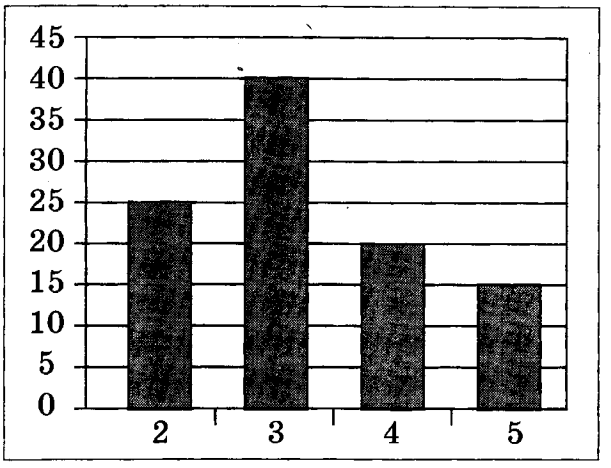
Какое число должно быть записано в ячейке B1, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:D2 соответствовала рисунку:



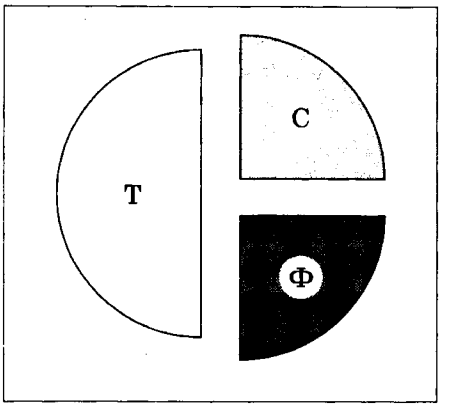
36*. В цехе трудятся рабочие трёх специальностей — токари (Т), слесари (С) и фрезеровщики (Ф). Каждый рабочий имеет разряд, не меньший второго и не больший пятого. На диаграмме I отражено количество рабочих с различными разрядами, а на диаграмме II — распределение рабочих по специальностям.

Каждый рабочий имеет только одну специальность и один разряд.

I



II



Имеются четыре утверждения:

А) Все рабочие третьего разряда могут быть токарями

Б) Все рабочие третьего разряда могут быть фрезеровщиками

В) Все слесари могут быть пятого разряда

Г) Все токари могут быть четвёртого разряда

Какое из этих утверждений следует из анализа обеих диаграмм?

1) А 3) В

2) Б 4) Г

37*. В фрагменте базы данных представлены сведения о родственных отношениях. Определите на основании приведённых данных фамилию и инициалы бабушки Ивановой А.И.

1) Иванов Т.М.

2) Черных И.А.

3) Цейс Т.Н.

4) Петренко Н.Н.

Таблица 1

ID	Фамилия_И.О.	Пол
71	Иванов Т.М.	М
85	Петренко И.Т.	М
13	Черных И.А.	Ж
42	Петренко А.И.	Ж
23	Иванова А.И.	Ж
96	Петренко Н.Н.	Ж
82	Черных А.Н.	М
95	Цейс Т.Н.	Ж
10	Цейс Н.А.	М
...		

Таблица 2

ID_Родителя	ID_Ребёнка
23	71
13	23
85	23
82	13
95	13
85	42
82	10
95	10
...	...

38*. Между четырьмя местными аэропортами: ОКТЯБРЬ. БЕРЕГ. КРАСНЫЙ и СОСНОВО ежедневно выполняются авиарейсы. Приведён фрагмент расписания перелётов между ними:

Аэропорт вылета	Аэропорт прилёта	Время вылета	Время прилёта
СОСНОВО	КРАСНЫЙ	06:20	08:35
КРАСНЫЙ	ОКТЯБРЬ	10:25	12:35
ОКТЯБРЬ	КРАСНЫЙ	11:45	13:30
БЕРЕГ	СОСНОВО	12:15	14:25
СОСНОВО	ОКТЯБРЬ	12:45	16:35
КРАСНЫЙ	СОСНОВО	13:15	15:40
ОКТЯБРЬ	СОСНОВО	13:40	17:25
ОКТЯБРЬ	БЕРЕГ	15:30	17:15
СОСНОВО	БЕРЕГ	17:35	19:30
БЕРЕГ	ОКТЯБРЬ	19:40	21:55

Путешественник оказался в аэропорту ОКТЯБРЬ в полночь (0:00). Определите самое раннее время, когда он может попасть в аэропорт СОСНОВО.

1) 15:40 2) 16:35 3) 17:15 4) 17:25

39*. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдёт поисковый сервер по каждому запросу.

Для обозначения логической операции «ИЛИ» в запросе используется символ «|», а для логической операции «И» — символ «&».

А	разведение & содержание & меченосцы & сомики
Б	содержание & меченосцы
В	(содержание & меченосцы) | сомики
Г	содержание & меченосцы & сомики

40. В языке запросов поискового сервера для обозначения логической операции «ИЛИ» используется символ «|», а для логической операции «И» — символ «&».

В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

Какое количество страниц (в тысячах) будет найдено по запросу *Огурцы* ?

Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Запрос	Найдено страниц (в тысячах)
Огурцы | Кабачки	12 000
Огурцы & Кабачки	6500
Кабачки	7700

41*. Определите значение переменной *c* после выполнения следующего фрагмента программы (записанного ниже на разных языках программирования):

Бейсик

```
a = 100
b = 30
a = a - b * 3
IF a > b THEN
   c = a - b
ELSE c = b - a
ENDIF
```

Паскаль

```
a := 100;
b := 30;
a := a - b *3;
if a > b then
     c := a - b
else c := b - a;
```

Си

```
a = 100;
b = 30;
a = a - b * 3;
if (a > b)
     c = a - b;
else c = b - a;
```

Алгоритмический язык

```
a := 100
b := 30
a := a - b * 3
если a > b то
     c := a  b
иначе c := b - a
все
```

1) 20 2) 70 3) 20 4) 180

42*. Требовалось написать программу, которая решает уравнение $ax + b = 0$ относительно x для любых чисел a и b, введённых с клавиатуры. Все числа считаются действительными. Программист торопился и написал программу неправильно.

Программа на Паскале

```
var a, b, x: real;
begin
readln(a,b,x);
if b = 0 then
write('x = 0') else
if a = 0 then
write('нет решений') else
write('x =', -b / a);
end.
```

Программа на Бэйсике

```
INPUT a, b, x
IF b = 0 THEN
PRINT "x = 0" ELSE
IF a = 0 THEN
PRINT "нет решений" ELSE
PRINT "x =", -b / a
ENDIF
ENDIF
END
```

Программа на Си

```
void main(void)
{float a, b, x;
 scanf("%f%f%f", &a,&b,&x);
 if (b == 0) printf("x = 0");
 else
 if (a == 0) printf("нет решений");
 else
 printf("x=%f", -b / a);
}
```

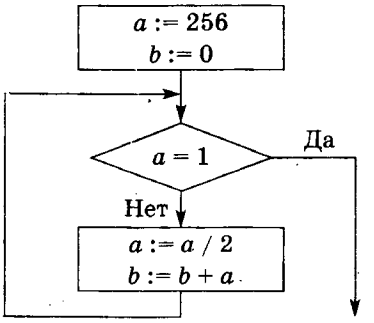
Последовательно выполните три задания:

1) Приведите пример таких чисел a, b, x, при которых программа неверно решает поставленную задачу.

2) Укажите, какая часть программы является лишней.

3) Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой способ доработки исходной программы).

43*. Запишите значение переменной b после выполнения фрагмента алгоритма:



Примечание: знаком ":=" обозначена операция присваивания.
В бланк ответа впишите только число.

44. Получив на вход число x, программа печатает два числа a и b. Укажите наименьшее из таких чисел x, при вводе которых программа напечатает сначала число 4, а потом число 24.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0;
  b := 1;
  while x > 0 do begin
    a := a + 1;
    b := b * (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

45*. В программе используется одномерный целочисленный массив A с индексами от 0 до 9. Ниже представлен фрагмент программы, записанный на разных языках программирования, в котором значения элементов сначала задаются, а затем меняются.

Бейсик

```
For i = 0 To 9
A.SetValue(9-i, i)
Next
For i = 0 To 4
K = A.GetValue(i)
A.SetValue(A.GetValue(9-i), i)
A.SetValue(k, 9-i)
Next
```

Паскаль

```
for i := 0 to 9 do
A[i] := 9 - i;
for i := 0 to 4 do
begin
k := A[i];
A[i] := A[9-i];
A[9-i] := k;
end;
```

Си

```
for (i = 0; i <= 9; i++)
A[i] = 9 - i;
for (i = 0; i <= 4; i++)
{
k = A[i];
A[i] = A[9-i];
A[9-i] = k;
}
```

Алгоритмический язык

```
нц для i от 0 до 9
A[i] := 9 - i
кц
нц для i от 0 до 4
k := A[i]
A[i] := A[9-i]
A[9-i] := k
кц
```

Чему будут равны элементы этого массива после выполнения фрагмента программы?

1) 9 8 7 6 5 4 3 2 1 0
2) 0 1 2 3 4 5 6 7 8 9
3) 9 8 7 6 5 5 6 7 8 9
4) 0 1 2 3 4 4 3 2 1 0

46*. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать значения от 0 до 1000. Опишите на русском языке или на одном из языков программирования алгоритм, который позволяет подсчитать и вывести среднее арифметическое элементов массива, имеющих нечётное значение. Гарантируется, что в исходном массиве хотя бы один элемент имеет нечётное значение.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из них.

Паскаль

```
const N = 30;
var
a: array [1..N] of integer;
i, x, y: integer;
s: real;
begin
for i := 1 to N do readln(a[i]);
...
end.
```

Бейсик

```
N = 30
DIM A(N) AS INTEGER
DIM I, X, Y AS INTEGER
DIM S AS SINGLE
FOR I = 1 TO N
INPUT A(I)
NEXT I
...
END
```

Си

```
#include <stdio.h>
#define N 30
void main(void)
{int a[N];
int i, x, y;
float s;
for (i = 0; i < N; i++)
scanf("%d", &a[i]);
...
}
```

Естественный язык

```
Объявляем массив A из 30 элементов.
Объявляем целочисленные переменные I, X,
Y.
Объявляем вещественную переменную S.
В цикле от 1 до 30 вводим элементы мас-
сива A с 1-го по 30-й.
...
```

В качестве ответа Вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например, Borland Pascal 7.0) или в виде блок-схемы. В этом случае вы должны использовать переменные, аналогичные переменным, используемым в алгоритме, записанном на естественном языке, с учётом синтаксиса и особенностей используемого вами языка программирования.

47*. Определите, какое число будет напечатано в результате следующей программы (для Вашего удобства программа представлена на четырёх языках):

Бейсик

```
Module A14
  Sub Main()
  Dim d, a, b, t, M, R As Double
    a = 0; b:=1
    d = 0.1
      t = a; M = a; R = F(a)
    While t < b
        If F(t) < R Then
          M = t
          R = F(t)
        End If
      t = t + d
    End While
    Console.Write(M)
  End Sub

Function F(ByVal x As Double) As Double
  Return (x - 1) * (x - 3)
  End Function

End Module
```

Паскаль

```
Program A14;
Uses crt;
Var d, a, b, t, M, R: real;
Function F(x : real): real;
  begin
    F := (x - 1) * (x - 3);
  end;
BEGIN
  a := 0; b := 1;
  d := 0.1;
  t := a; M := a; R := F(a);
  while t < b do
    begin
      if (F(t) < R) then
        begin
          M := t; R := F(t);
        end;
      t := t + d;
    end;
  write(M);
END.
```

Окончание табл.

Си	Алгоритмический язык

`#include <stdio.h>`
`double F(double x)`
`{`
`  return (x - 1) * (x - 3);`
`}`
`void main()`
`{`
`  double d, a, b, t, M, R;`
`  a = 0; b = 1;`
`  d = 0.1;`
`  t = a; M = a; R = F(a);`
`  while (t < b) {`
`   if (F(t) < R) {`
`     M = t; R = F(t);`
`     }`
`   t = t + d;`
`   }`
`  printf("%f", M);`
`}`
	алг A14
	нач
	  вещ d, a, b, t, M, R
	  a:= 0; b:= 1
	  d:= 0.1
	  t:= a; M:= a; R:= F(a)
	  нц пока t < b
	  если F(t) < R
	  то
	  M := t; R := F(t)
	  все
	  t := t + d
	  кц
	  вывод M
	кон
	алг вещ F(вещ x)
	нач
	знач := (x - 1) * (x - 3)
	кон

1) 1 **2) 2** **3) –3** **4) 24**

48*. Что будет напечатано в результате выполнения этой программы?

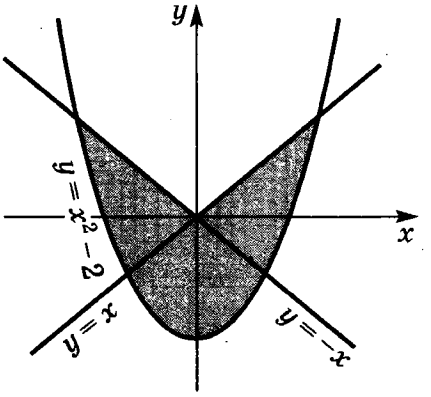
```
Program Task;
Uses crt;
const L = 5;
type
    atype = array [1..L] of integer;
var R: atype;
    N, p: integer;

Procedure Multiply3_2(L, p: integer; var R: atype );
var i, n, t: integer;
begin
p := 0;
for i := 1 to L do
  begin
    t := 2 * R[i] + p;
    R[i] := (t) mod (3);
    p := (t) div (3);
  end;
end;

Function Calc3 (L: integer; R: atype): integer;
var N, i, T: integer;
begin
  N := 0;
  T := 1;
  for i := 1 to L do
   begin
    N := N + T * R[i];
    T := T * 3;
   end;
  Calc3 := N;
end;
```

```
BEGIN
  R[1] := 2; R[2] := 2; R[3] := 0; R[4] := 1; R[5] := 0;
  Multiply3_2(L, p, R);
  if (p > 0) then
   begin
    write('Переполнение');
    halt;
   end;
  N := Calc3(L,R);
  write(N);
  writeln;
END.
```

49*. Требовалось написать программу, которая вводит координаты точки (x, y действительные числа) и определяет принадлежность этой точки заданной области (включая её границы). Программа была составлена неправильно.



Паскаль

```
var x, y: real;
begin
  readln(x, y);
  if y <= x then
    if y <= -x then
      if y >= x * x - 2 then
        write('принадлежит')
        else
write('не принадлежит')
end.
```

Бейсик

```
INPUT x, y
IF y <= x THEN
  IF y <= -x THEN
    IF y >= x * x - 2 THEN
      PRINT "принадлежит"
        ELSE
PRINT "не принадлежит"
      ENDIF
    ENDIF
  ENDIF
END
```

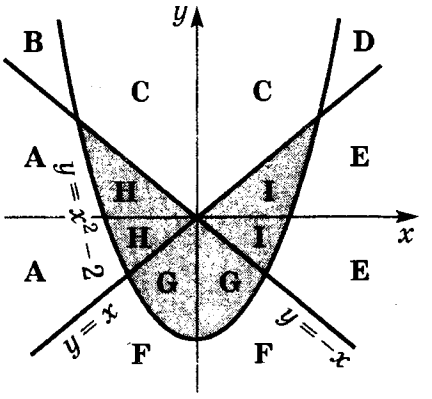
Си

```
void main(void)
{float x, y;
 scanf ("%f%f", &x, &y);
 if (y <= x)
   if (y <= -x)
     if (y >= x * x - 2)
       printf("принадлежит");
 else
printf("не принадлежит");
```

Алгоритмический язык

```
алг
нач
вещ x, y
ввод x, y
если y <= x то
  если y <= -x то
    если y >= x * x - 2 то
      вывод 'принадлежит'
    иначе
      вывод 'не принадлежит'
  все
все
кон
```

1. Заполните таблицу, которая показывает, как работает программа при значениях x и y, принадлежащих различным областям (A, B, C, D, E, F, G, H и I).



Область	Условие 1 ($y <= x$)	Условие 2 ($y <= -x$)	Условие 3 ($y >= x*x - 2$)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					
G					
H					
I					

В столбцах условий укажите «да», если условие выполняется, «нет», если условие не выполняется, «—» (прочерк), если условие не проверяется, «неизв.», если программа ведёт себя по-разному для разных значений, принадлежащих данной области. В столбце «Программа выведет» укажите, что программа выведет на экран. Если программа ничего не выводит, то запишите «—» (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, то напишите «неизв.». В последнем столбце укажите «да» или «нет».

2. Укажите, как доработать программу для её правильного функционирования. Достаточно указать любой способ доработки исходной программы.)

50*. На автозаправочных станциях (АЗС) продаётся бензин с маркировкой 92, 95 и 98. В городе N был проведён мониторинг цены бензина на различных АЗС.

Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет определять для каждого вида бензина, сколько АЗС продают его дешевле всего. На вход программе в первой строке подаётся число данных о стоимости бензина. В каждой из последующих N строк находится информация в следующем формате:

<Компания> <Улица> <Марка> <Цена>,

где <Компания> — строка, состоящая не более, чем из 20 символов без пробелов, <Улица> — строка, состоящая не более, чем из 20 символов без пробелов, <Марка> — одно из чисел —

92, 95 или 98, <Цена> — целое число в диапазоне от 1000 до 3000, обозначающее стоимость одного литра бензина в копейках. <Компания> и <Улица>, <Улица> и <Марка>, а также <Марка> и <цена> разделены ровно одним пробелом.

Пример входной строки:

Синойл Цветочная 95 2250

Программа должна выводить через пробел 3 числа — количество АЗС, продающих дешевле всего 92-й, 95-й и 98-й бензин соответственно. Если бензин какой-то марки нигде не продавался, то следует вывести 0.

Пример выходных данных:

12 1 0

Ответы и решения

№ задания	Ответ / решение	№ занятия при неправильном решении
1	60 (вариант №2)	1
2	4	
3	3 (вариант № 3)	
4	540 (вариант №1)	
5	110100001001100111 (вариант № 3)	2
6	2400	3
7	40	
8	7 (вариант № 2).	4
9	4 (вариант № 4).	5
10	10101000_2 (вариант № 4).	
11	7	
12	5, 21	
13	MMMPP	6
14	AABAAB	
15	X $\vee$ BY $\vee$ BZ (вариант № 4)	7
16	ИРИНА (вариант №1)	
17	3 (вариант №3)	
18	1100	
19	10	8
20	31	
21	1411 (вариант № 3)	9
22	19 (вариант № 3)	10
23	11221	11
24	11	
25	ABFB52 (вариант № 1)	

Продолжение табл.

№ задания	Ответ / решение	№ занятия при неправильном решении
26	4	12
27	E:\Класс9\Физика\Задача5 (вариант № 4)	13
28	*23*.??? (вариант № 4)	
29	DFBH	14
30	4 кбайт	15
31	16 (вариант № 3)	
32	=E1-$D2 (вариант №2)	16
33	4 (вариант №4)	
34	365 (вариант № 1)	
35	2	17
36	А (вариант №1)	
37	Цейс Т.Н. (вариант № 3)	18
38	17:25 (вариант № 4)	
39	АГБВ	19
40	10800	
41	20 (вариант № 1)	20

42	1. Лишней частью программы является ввод в операторе `readln` значения x, которое должно вычисляться в программе.
	2. Ошибка: первый оператор `if` в случае, если $b = 0$, не анализирует значение a и сразу выдаёт результат «$x = 0$».
	3. Возможный вид доработанной программы, не содержащей лишних фрагментов:
	`var a, b, x: real;`
	`begin`
	`readln(a,b);`
	`if a = 0 then`
	`  begin`
	`  if b = 0 then`
	`  write('любое число')`
	`  else`
	`  write('нет решений')`
	`  end`
	`  else`
	`  write('x =',-b / a);`
	`end.`	

43	255	21
44	1146	22
45	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (вариант № 2)	23

<table>
<tr><th>№ задания</th><th>Ответ / решение</th><th>№ занятия при неправильном решении</th></tr>
<tr><td>46</td><td>

```
const N = 30;
var a: array [1..N] of integer;
    i, x, y: integer;
    s: real;
begin
  for I := 1 to N do readln(a[i]); // считали массив
  x := 0; y := 0;
// x - переменная для подсчета суммы элементов,
// удовлетворяющих заданному условию;
//y – переменная для подсчета количества этих элементов;
//изначально обе переменные обнуляются
  for i := 1 to N do
//полный просмотр массива
    if (a[i] mod 2 = 1) then begin
//если текущий элемент удовлетворяет условию (нечетный), то
    x := x + a[i];
//прибавляем этот элемент к накапливаемой сумме
    y := y + 1;
//и увеличиваем количество просуммированных элементов на 1
  end;
  s := x / y;
//по окончании цикла вычисляем среднее арифметическое, деля
//полученную сумму нечетных чисел на их количество
writeln(s);
//вывод среднего арифметического значения нечетных элементов
end.
```

</td><td>24</td></tr>
<tr><td>47</td><td>1 (вариант № 1)</td><td>25</td></tr>
<tr><td>48</td><td>70</td><td>26</td></tr>
<tr><td>49</td><td>1.
<table>
<tr><th>Область</th><th>Условие 1 (y <= x)</th><th>Условие 2 (y <= –x)</th><th>Условие 3 (y >= x*x – 2)</th><th>Программа выведет</th><th>Область обрабатывается верно</th></tr>
<tr><td>A</td><td>Нет</td><td>—</td><td>—</td><td>—</td><td>Нет</td></tr>
<tr><td>B</td><td>Нет</td><td>—</td><td>—</td><td>—</td><td>Нет</td></tr>
<tr><td>C</td><td>Нет</td><td>—</td><td>—</td><td>—</td><td>Нет</td></tr>
<tr><td>D</td><td>Нет</td><td>—</td><td>—</td><td>—</td><td>Нет</td></tr>
<tr><td>E</td><td>Да</td><td>Нет</td><td>—</td><td>—</td><td>Нет</td></tr>
<tr><td>F</td><td>Да</td><td>Да</td><td>Нет</td><td>Не принадлежит</td><td>Да</td></tr>
<tr><td>G</td><td>Да</td><td>Да</td><td>Да</td><td>Принадлежит</td><td>Да</td></tr>
<tr><td>H</td><td>Нет</td><td></td><td>—</td><td>—</td><td>Нет</td></tr>
<tr><td>I</td><td>Да</td><td>Нет</td><td>—</td><td>—</td><td>Нет</td></tr>
</table></td><td>27</td></tr>
<tr><td>49</td><td>2. Пример исправления программы:

```
var x, y: real;
begin
readln(x, y);
if (y >= x * x - 2) and
   ((y <= -x) or (y <= x)) then
write('принадлежит')
else
write('не принадлежит')
end.
```

</td><td>27</td></tr>
</table>

Окончание табл.

№ задания	Ответ / решение	№ занятия при неправильном решении

50	
	program C4;
	var min, ans: array[92..98] of integer;
	c: char;
	i, k, N, b: integer;
	begin
	for i := 92 to 98 do
	//инициализация начальных значений
	begin
	//предполагаемых минимумов и счетчиков
	min[i] := 3001;
	ans[i] := 0;
	end;
	
	readln(N);
	//считано количество строк
	for i := 1 to N do
	//чтение строк входных данных
	begin
	repeat
	read(c);
	until c=' ';
	//пропуск названия компании
	repeat
	read(c);
	until c=' ';
	//пропуск названия улицы
	readln(k,b);
	//чтение номера марки бензина и его цены
	
	//поиск минимума с подсчетом количества элементов,
	//равных этому минимуму
	if b < min[k] then
	begin
	min[k] := b;
	ans[k] := 1
	end else
	if min[k] = b then
	ans[k] := ans[k] + 1;
	end;
	
	//вывод результатов
	writeln(ans[92],' ', ans[95],' ', ans[98])
	end.
		28, 29, 30

Информация. Измерение информации. Кодирование информации

A11, B1 Измерение количества информации

Конспект

Определение количества информации

Подходы к измерению информации

Содержательный (вероятностный)	Алфавитный (объёмный)
Через неопределённость знаний с учётом вероятности событий	Через количество символов с учётом информационного веса символа

Формула Хартли

Для N равновероятных возможных вариантов события количество информации (I), которое несёт сообщение о выборе (совершении) одного конкретного варианта, определяется формулой Хартли:

$I = \log_2 N$,

где log — функция логарифма по основанию 2, обратная возведению значения основания логарифма в степень, равную I, т.е. из формулы Хартли следует зависимость:

$N = 2^I$,

для облегчения вычислений значений N, представляющих собой степень числа.

Формула Шеннона. Связь количества информации с понятием вероятностей

Для N событий с различными вероятностями p_1, p_2, ... p_N количество информации определяется формулой Шеннона:

$$I = -\sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i}.$$

Если события равновероятны, т.е. $p_1 = p_2 = \ldots = p_N = p$, то формула Шеннона преобразуется в формулу Хартли (которая, таким обра-

При алфавитном подходе информационное сообщение рассматривается как некоторое количество (K) знаков (символов, кодов) из некоторого используемого полного набора, называемого алфавитом. Количество (N) знаков в алфавите называется мощностью этого алфавита.

В данном конкретном сообщении не обязательно используются все знаки алфавита. Мощность алфавита определяется не набором знаков, используемых в конкретном сообщении, а количеством знаков, которые могут быть вообще использованы в сообщениях, кодируемых в соответствии с данным алфавитом.

Алгоритм определения количества информации в сообщении:

1) определяется мощность используемого алфавита N;

2) определяется количество информации, приходящееся в алфавите на один его знак:

- если использование всех знаков считается равновероятным, то используется формула Хартли (либо её следствие: $N = 2^I$);
- если известны вероятности использования тех или иных знаков (на основе составленной таблицы частоты встречаемости этих знаков), то используется формула Шеннона;

зом, представляет собой частный случай формулы Шеннона).

Связь между количеством информации и вероятностью события

Для N равновероятных событий вероятность одного отдельного события $p = 1/N$. С учётом этого формула Хартли может быть преобразована в соотношение:

$$I = \log_2 \frac{1}{p}.$$

3) вычисленное количество информации (I), приходящееся на один знак, умножается на количество (K) знаков в данном сообщении:

$$I_\Sigma = I \cdot K.$$

В теории информации количество информации может быть дробной величиной. Количество информации может составлять только целое число битов (дробное значение при необходимости **округляется в большую сторону**).

Для обозначения количеств информации, бóльших, чем байт, приняты следующие производные величины:

1 килобайт (КБ) = (2^{10} = 1024) байт;

1 Мегабайт (МБ) = (2^{10} = 1024) килобайт = (2^{20} = 1048576) байт;

1 Гигабайт (ГБ) = (2^{10} = 1024) Мегабайт = (2^{20} = 1048576) килобайт = = (2^{30} = 1073741824) байт;

1 Терабайт (ТБ) = (2^{10} = 1024) Гигабайт = (2^{20} = 1048576) Мегабайт = = (2^{30} = 1073741824) килобайт = (2^{40} = 1099511627776) байт.

Таблицы степеней

Существенную помощь при вычислениях может оказать таблица значений степеней различных оснований систем счисления: 2, 3 и т.д. Ниже представлены две такие таблицы:

Таблица 1

n	0	1	2	3	4	5	6	7	8	9	10
2^n	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Значение	1	2	4	8	16	32	64	128	256	516	1024

Таблица 2

n	0	1	2	3	4	5	6	7	8	9	10
3^n	3^0	3^1	3^2	3^3	3^4	3^5	3^6	3^7	3^8	3^9	3^{10}
Значение	1	3	9	27	81	243	729	2187	6561	19683	59049

Продолжить их или составить аналогичные таблицы для других оснований систем счисления можно самостоятельно (каждое очередное значение есть произведение предыдущего на соответствующее основание системы счисления).

Разбор типовых задач

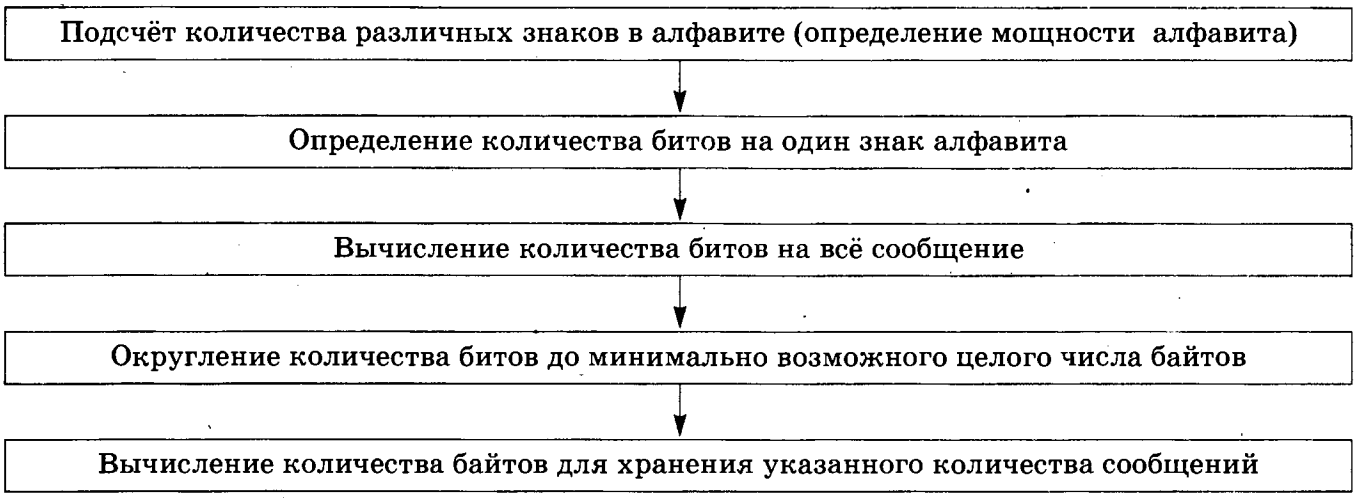
Задача 1*. Для регистрации на сайте некоторой страны пользователю требуется придумать пароль. Длина пароля ровна 11 символов. В качестве символов используются десятичные цифры и 12 различных букв местного алфавита, причём все буквы используются в двух начертаниях: как строчные, так и заглавные (регистр буквы имеет значение!).

Под хранение каждого такого пароля на компьютере отводится минимально возможное и одинаковое целое количество байтов, при этом используется посимвольное кодирование и все символы кодируются одинаковым и минимально возможным количеством битов.

Определите объём памяти, который занимает хранение 60 паролей.

1) 540 байт 2) 600 байт 3) 660 байт 4) 720 байт

Схема решения задачи



Решение

Данная задача решается с использованием алфавитного подхода к измерению количества информации.

1) Определяется мощность используемого алфавита. Используются десятичные цифры (10 различных знаков) и 12 букв, каждая из которых может иметь два возможных начертания ($12 \times 2 = 24$ различных знака). Итого мощность (N) используемого алфавита составляет: $10 + 12 \times 2 = 34$ знака.

2) Исходя из известной мощности ($N = 34$) алфавита определяется количество битов, соответствующее каждому знаку. Речь идёт о целом (не дробном!) количестве битов, минимально достаточном для представления одного знака такого алфавита. Поэтому выбираем ближайшее большее число N, равное степени числа 2: $N = 2^6 = 64$ (значения $2^5 = 32$ недостаточно). Тогда согласно формуле $N = 2^I$ получается: $I = 6$ бит на один знак алфавита.

3) Длина пароля (т.е. длина сообщения, кодируемого с использованием рассмотренного алфавита) равна 11 символам ($K = 11$). Тогда согласно формуле $I_\Sigma = I \cdot K$, количество информации в битах, соответствующее всему такому сообщению (паролю), равно $6 \cdot 11 = 66$ битов.

4) По условию задачи, под хранение каждого такого пароля на компьютере отводится минимально возможное и одинаковое целое количество байтов. Определяется минимальное количество целых байтов, достаточное, чтобы уместить 66 битов. 1 байт равен 8 битам. Выполняется деление количества битов (66) на 8 с округлением результата до целого в большую сторону: $66 / 8 = 8{,}25 \rightarrow 9$ байтов.

Следует не забывать выполнять перевод количества битов в количество байтов!

5) Для хранения 60 паролей требуется $60 \cdot 9 = 540$ байтов.

Ответ: 540 байтов

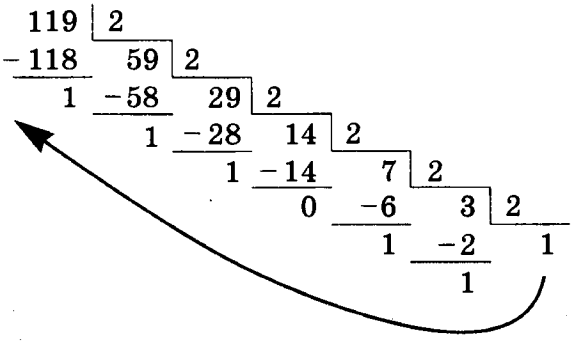
Задача 2*. В велокроссе участвуют 119 спортсменов. Специальное устройство регистрирует прохождение каждым из участников промежуточного финиша, записывая его номер с использованием минимально возможного количества бит, одинакового для каждого спортсмена. Каков информационный объём сообщения, записанного устройством, после того как промежуточный финиш прошли 70 велосипедистов?

1) 70 бит 2) 70 байт 3) 490 бит 4) 119 байт

Решение

Данная задача требует умения преобразовывать числа из десятичной системы счисления в двоичную.

1) Максимальное значение номера спортсмена равно 119_{10}, что соответствует 1110111_2:



2) Для хранения этого наибольшего номера (а значит, и для хранения любого номера, меньшего его) достаточно 7 битов.

> В данной задаче, в отличие от предыдущей, не сказано, что для хранения сообщения (номера) отводится целое количество байтов, а в вариантах ответа есть значения и в битах, и в байтах. Поэтому пока значение оставляется в битах.

3) Регистрирующее устройство записало 70 таких сообщений, тогда общее количество информации (в битах) равно $70 \cdot 7 = 490$ битов.

Ответ: 490 битов.

Задача 3*. Автоматическое устройство осуществило перекодировку информационного сообщения на русском языке длиной в 20 символов, первоначально записанного в 2-байтном коде Unicode, в 8-битную кодировку КОИ-8. На сколько бит уменьшилась длина сообщения?

Решение

Информационный объём исходного сообщения (I_1) равен:

20 (символов) × 2 (байта) = 40 байтов = $40 \cdot 8 = 320$ битов.

Информационный объём перекодированного сообщения (I_2) равен:

20 (символов) × 8 (битов) = 160 битов.

Уменьшение информационного объёма («длины») сообщения при его перекодировке составляет:

$\Delta I = I_1 - I_2 = 320 - 160 = 160$ битов.

Ответ: 160 битов.

Задача 4. В коробке лежат 8 чёрных бусин и 24 белых. Сколько информации несёт сообщение о том, что из коробки достали чёрную бусину?

Решение (вариант 1)

При решении этой задачи применяется вероятностный подход к определению количества информации с использованием формулы Шеннона:

$$I = -\sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i},$$

где I — количество информации, N — количество возможных событий, а p_i — вероятность каждого из отдельных событий.

Опираясь на эту формулу, можно вычислить количество информации, которую несёт отдельное событие из числа нескольких неравновероятных событий:

$$h_i = -\log_2 p_i = \log_2 \frac{1}{p_i}.$$

Тогда решение задачи сводится к следующим трём действиям.

1. Общее количество бусин в коробке: 8 + 24 = 32 штуки.

2. Вероятность того, что из коробки будет извлечена чёрная бусина: $p_ч = 8/32 = 1/4$.

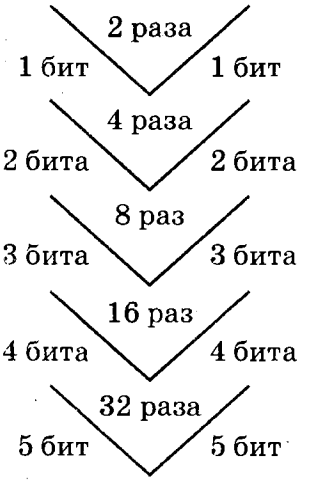
3. Согласно приведённой выше формуле, $h_ч = \log_2(1/p_ч) = \log_2(4) = 2$ бита (т. к. $4 = 2^2$).

Решение (вариант 2)

1. Общее количество бусин в коробке: 8 + 24 = 32 штуки.

2. Предположим, что мы вытаскиваем из коробки бусины наугад. Очевидно, что чем больше в ней чёрных бусин, тем чаще мы будем вынимать именно чёрную бусину, и наоборот. Очевидно также, что количество случаев, в которых мы вынимаем из коробки чёрную бусину, пропорционально отношению количества чёрных бусин к общему количеству бусин в коробке. Тогда количество случаев, когда из коробки вынимается чёрная бусина, будет соответствовать 8/32 = 1/4.

3. Известно, что количество информации, равное 1 биту, соответствует снятию неопределённости при помощи ответа «да» или «нет» на один элементарный вопрос, т. е., 1 бит соответствует уменьшению неопределённости в 2 раза. При уменьшении неопределённости в 4 раза можно задать последовательно два вопроса, на которые даются ответы «да» или «нет». В общем случае количество информации в n бит позволяет уменьшить неопределённость в 2^n раз. Для облегчения вычислений составляется схема:



В данной задаче количество случаев вынимания чёрной бусины равно 1/4. Тогда можно считать, что это аналогично выбору одного из четырёх возможных объектов или уменьшению неопределённости в 4 раза. Отсюда из приведённой выше схемы легко получить, что это соответствует 2 битам информации.

Ответ: 2 бита.

Задача 5. В закрытом ящике находится 32 карандаша, некоторые из них синего цвета. Наугад вынимается один карандаш. Сообщение «этот карандаш — НЕ синий» несёт 4 бита информации. Сколько синих карандашей в ящике?

Решение (вариант 1)

В этой задаче требуется знакомство с понятием вероятности обратного события, — в частности, знания того, что если вероятность некоторого события равна p, то вероятность того, что это событие НЕ ПРОИЗОЙДЁТ, равна $(1 - p)$.

1. Сообщение «вынутый карандаш — НЕ синий» несёт 4 бита информации. Тогда, согласно формуле

$$h_{не\_с} = -\log_2 p_{не\_с} = \log_2 \frac{1}{p_{не\ с}}$$

вероятность того, что вынутый карандаш — НЕ синий, равна 1/16.

2. Тогда вероятность обратного события (что будет вынут синий карандаш) равна: $p_с = 1 - p_{не\_с} = 1 - 1/16 = 15/16$.

3. Общее количество карандашей равно 32, тогда синих среди них: $32 \cdot 15/16 = 30$ штук.

Решение (вариант 2)

1. По схеме-«ёлочке» определяется, что 4-м битам информации соответствует уменьшение неопределённости в 16 раз.

2. Уменьшение неопределённости в 16 раз соответствует тому, что достать из коробки НЕ синий карандаш можно в одном случае из 16. Тогда во всех остальных случаях оттуда достаётся именно синий карандаш. То есть, синий карандаш будет извлекаться из ящика в 15 случаях из 16-ти.

3. Учитывая, что количество случаев вынимания синих карандашей пропорционально отношению количества синих карандашей к общему их количеству, получается, что синих карандашей в ящике — $32 \cdot 15/16 = 30$ штук.

Ответ: 30 синих карандашей.

Задача 6. Светодиодная панель содержит пять излучающих элементов, каждый из которых может светиться или красным, или зелёным светом (или не светится вообще). Сколько различных сигналов можно передать при помощи такой панели?

Решение

1. Каждый излучающий элемент может находиться в одном из трёх состояний: «не горит», «красный», «зелёный», — следовательно, это аналог троичной системы счисления.

2. Всего панель содержит 5 таких элементов, значит, она является аналогом пятизначного троичного числа.

3. Количество различных сигналов, которые можно подавать при помощи такой панели, равно количеству всех возможных значений троичных чисел от 00000 до 22222 включительно, т.е. $3^5 = 243$ сигнала.

> Количество различных состояний панели, имеющей M элементов, каждый из которых может находиться в N различных состояниях, равно количеству различных M-разрядных чисел (начиная с нуля) в системе счисления с основанием N и вычисляется как N^M.

Ответ: 243 сигнала.

Задача 7. Для кодирования сообщений используются ряды красных и зелёных флажков (без пропусков между ними).Сколько различных сообщений можно закодировать, используя не менее 5 и не более 10 флажков в такой цепочке?

Решение (способ 1)

1. Цепочка представляет собой набор флажков двух видов, без пропусков между ними. Можно обозначить красный цвет флажков цифрой 0, а зелёный — цифрой 1. Такая цепочка флажков аналогична последовательности цифр 0 и 1 соответствующей длины.

2. Речь идёт о двоичной системе счисления, значит, количество различных сообщений, кодируемых N двоичными цифрами, можно вычислить как 2^N.

3. Цепочка флажков (или строка цифр 0 и 1) может иметь длину 5, 6, 7, 8, 9 или 10 штук. Тогда:

- количество сообщений, передаваемых цепочкой из 5 флажков, равно $2^5 = 32$;
- количество сообщений, передаваемых цепочкой из 6 флажков, равно $2^6 = 64$;
- количество сообщений, передаваемых цепочкой из 7 флажков, равно $2^7 = 128$;
- количество сообщений, передаваемых цепочкой из 8 флажков, равно $2^8 = 256$;
- количество сообщений, передаваемых цепочкой из 9 флажков, равно $2^9 = 512$;
- количество сообщений, передаваемых цепочкой из 10 флажков, равно $2^{10} = 1024$.

Всего при помощи таких цепочек флажков можно передать:
32 + 64 + 128 + 256 + 512 + 1024 = 2016 сообщений.

Ответ: 2016 сообщений.

> Для решения подобных задач очень полезно помнить таблицу степеней двоек.

Решение (способ 2)

1. Цепочка представляет собой набор флажков двух видов, без пропусков между ними. Можно обозначить красный цвет флажков цифрой 0, а зелёный — цифрой 1. Такая цепочка флажков аналогична последовательности цифр 0 и 1 соответствующей длины.

2. Возможные цепочки флажков (строки цифр 0 и 1):

10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	0	0	0	0

Здесь единицами закодированы возможные длины цепочек, а нулями — не используемые (в том числе нулевая). Получается 10-разрядное двоичное число: 11111100000.

3. Количество всех возможных сообщений, которые можно передать с помощью такого набора цепочек нулей и единиц, равно десятичному значению полученного двоичного числа:

$11111100000 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 = 2016$.

Ответ: 2016 сообщений.

Задача 8. У малыша есть кубики четырёх цветов. Он собирает из них башенки высотой по 6 кубиков. Кубики он кладёт в любом порядке. Башенки считаются разными, если отличаются хотя бы одним кубиком или порядком их выкладывания. Сколько различных башенок может сложить малыш?

Решение

Четыре цвета кубиков эквивалентны четырём видам цифр. Поэтому в решении данной задачи используется четверичная система счисления.

Башенки из 6 кубиков каждая эквивалентны 6-разрядным числам.

Следовательно, надо определить, сколько существует 6-разрядных чисел в четверичной системе счисления.

> Количество всех M-разрядных чисел в системе счисления с основанием N равно N^M. Максимальное значение M-разрядного числа в системе счисления с основанием N равно $(N^M - 1)$.

Количество всех возможных 6-разрядных чисел в системе счисления с основанием 4 равно $4^6 = 4096$. Различных башен из 6 кубиков 4 видов тоже может быть 4096.

Ответ: 4096 башен.

Задачи для самостоятельного решения

1. Сообщение на русском языке закодировано 2-байтным кодом Unicode и имеет длину 1025024 символа. На сколько килобайт уменьшится длина этого сообщения после его перекодировки в 8-битный код КОИ-8R? В ответе нужно записать только число.
2. Каждый пользователь сайта получает при регистрации пароль из 16 символов из набора A, B, C, E, H, K, M. Каждый пароль записывается в памяти сервера минимально возможным и одинаковым целым количеством байтов (при этом используется посим-

вольное кодирование, все символы кодируются одинаковым и минимально возможным количеством битов).

Определите объём памяти, требуемый для записи 40 паролей.

1) 160 байт 2) 200 байт 3) 240 байт 4) 280 байт

3. В школе 32 учащихся учатся в двух классах, 5А и 5Б. Одна из учениц заболела Сообщение «Заболевшая девочка учится в 5Б классе» содержит 4 бита информации. Сколько учащихся учатся в классе 5А?

   1) 4 2) 16 3) 28 4) 30

4. Дети украшают класс к Новому году разноцветными флажками — красными, зелёными и синими, подвешивая цепочки флажков к потолку. На нитку можно нанизать ровно четыре флажка. Они могут повторяться и нанизываться в любом порядке. Сколько разных ниток с флажками можно сделать? (Они считаются разными, если отличаются хотя бы одним флажком или порядком флажков на нитке.)
5. Для кодирования сообщений решено использовать последовательности разной длины, состоящие из цифр «1» и «0».Сколько различных сообщений можно закодировать, используя в каждом из них не менее 4-х и не более 8 знаков?
6. Объём сообщения равен 16Кбайт. Сообщение содержит 8192 символа. Какова максимальная мощность алфавита, использованного при передаче сообщения?
7. Для передачи сообщений из Цветочного города в Солнечный город и обратно Винтик и Шпунтик сделали световой телеграф: четыре прожектора расположены в одну линию и имеют цветные стекла, с помощью которых можно установить цвет каждого прожектора — красный, зелёный или синий. Какое количество различных сообщений можно передать с помощью такого оптического телеграфа?
8. Какое наименьшее количество прожекторов надо установить в оптическом телеграфе Винтику и Шпунтику (см. предыдущую задачу), чтобы с его помощью можно было передать 18 различных сигналов?
9. В базе данных вуза хранится информация о номерах студенческих билетов. Длина номера — ровно 8 символов. В качестве символов могут использоваться буквы русского алфавита (кроме «ё», «о», «ь», «ъ» и «ы», как строчные, так и заглавные — они считаются различными), знак «-» и десятичные цифры. Под каждый такой номер отводится минимально возможное и одинаковое целое количество байтов, при этом используется посимвольное кодирование и все символы кодируются одинаковым и минимально возможным количеством битов. Определите объём памяти в байтах, который необходим для хранения 150 номеров студенческих билетов.
10. В ходе телевизионного шоу проводится СМС-голосование зрителей: каждый телезритель выбирает одного из 6 артистов-участников шоу. Каждый голос телезрителя кодируется минимально необходимым количеством бит. За время телепередачи в голосовании приняли участие 983 040 зрителей. Каким будет объём собранной информации о голосовании в килобайтах?

Ответы для самопроверки

№ задания	Ответ	№ задания	Ответ
1	11	6	65536
2	3	7	81
3	4	8	3
4	81	9	1050
5	496	10	360

Информация. Измерение информации. Кодирование информации

A9 Неравномерный двоичный код

Конспект

Кодирование — это перевод информации с одного языка на другой (запись в другой системе символов, в другом алфавите).

Главное при таком кодировании — обеспечить возможность однозначного декодирования записанной с помощью этих кодов строки (поочередного, слева направо, выделения и распознавания из сплошной последовательности нулей и единиц кодов отдельных букв). Для этого коды символам необходимо назначать в соответствии с *условиями Фано*:

Закодированное сообщение можно однозначно декодировать *с начала*, если выполняется **условие Фано**: никакое кодовое слово не является началом другого кодового слова.

Закодированное сообщение можно однозначно декодировать *с конца*, если выполняется **обратное условие Фано**: никакое кодовое слово не является окончанием другого кодового слова.

Условие Фано — это достаточное, но не необходимое условие однозначного декодирования.

Разбор типовых задач

Задача 1. Для кодирования некоторой последовательности, состоящей из букв А, Б, В, Г и Д, решили использовать неравномерный двоичный код, позволяющий однозначно декодировать двоичную последовательность, появляющуюся на приёмной стороне канала связи. Использовали код: А—00, Б—10, В—110, Г—111. Укажите, каким кодовым словом должна быть закодирована буква Д. Длина этого кодового слова должна быть наименьшей из всех возможных. Код должен удовлетворять свойству однозначного декодирования.

1) 1 2) 01 3) 010 4) 011

Решение

Опираясь на правила Фано, нетрудно сформулировать для решения подобных задач простое правило:

1) берётся интересующий код (в данном случае — код буквы Д; для определения правильного варианта ответа нужно поочередно проверять каждый из этих вариантов);

2) этот код по очереди сравнивается с каждым из уже существующих кодов (в данном случае — для букв А, Б, В и Г), при этом:

- если длина обоих сравниваемых кодов совпадает, то проверяется равенство этих кодов: если проверяемый код равен существующему коду другой буквы, то этот код неправильный (данный вариант ответа неверен);
- если длина сравниваемых кодов различна, то более короткий код условно «подписывается» под более длинным, выравнивая по левому знаку, и сравниваются эти две записи: ес-

ли все знаки более короткого кода совпадают с соответствующими им знаками в начале более длинного кода, то прямое правило Фано не выполнено;

- если не выполнено прямое правило Фано, то проверяется обратное правило Фано — более короткий код условно «подписывается» под более длинным, выравнив их по правому знаку, и сравниваются эти две записи: если все знаки более короткого кода совпадают с соответствующими им знаками в начале более длинного кода, то обратное правило Фано не выполнено;
- если для данного варианта проверяемого кода не выполняется ни прямое, ни обратное правило Фано, то данный код использовать нельзя (данный вариант ответа неправильный).

Решение задачи удобно записать в виде таблицы:

Проверяемый код буквы Д	Существующие коды букв А, Б, В и Г				Вывод
	А	Б	В	Г	
	00	10	110	111	
1	00 1 Совпадения нет	10 1 **Совпадение есть!**	110 1 **Совпадение есть!**	111 1 **Совпадение есть!**	Код 1 для буквы Д непригоден
01	00 01 Коды не равны	10 01 Коды не равны	110 01 Совпадения нет	111 01 Совпадения нет	**Код 01 для буквы Д пригоден**
010	010 00 Совпадения нет	010 10 Совпадения нет	110 010 Коды не равны	111 010 Коды не равны	**Код 010 для буквы Д пригоден**
011	011 00 Совпадения нет	011 10 Совпадения нет	110 011 Коды не равны	111 011 Коды не равны	**Код 011 для буквы Д пригоден**

Таким образом, допустимыми для буквы Д являются коды: 01, 010 и 011.

По условию задачи, длина кодового слова должна быть наименьшей из всех возможных. Этому условию из вышеперечисленных удовлетворяет код 01.

Ответ: 01.

Задача 2*. Для кодирования некоторой последовательности, состоящей из букв А, Б, В, Г и Д, решили использовать неравномерный двоичный код, позволяющий однозначно декодировать двоичную последовательность, появляющуюся на приёмной стороне канала связи. Использовали код: А—1, Б—000, В—001, Г—011. Укажите, каким кодовым словом должна быть закодирована буква Д. Длина этого кодового слова должна быть наименьшей из всех возможных. Код должен удовлетворять свойству однозначного декодирования.

1) 00 2) 01 3) 11 4) 010

Решение

Проверяемый код буквы Д	Существующие коды букв А, Б, В и Г				Вывод
	А	Б	В	Г	
	1	000	001	011	
00	00 1 Совпадения нет	000 00 **Совпадение есть!**	001 00 **Совпадение есть!**	011 00 Совпадения нет	Код 01 для буквы Д непригоден

Окончание табл.

Проверяемый код буквы Д	Существующие коды букв А, Б, В и Г				Вывод
	А	Б	В	Г	
	1	000	001	011	
01	01 1 Совпадения нет	000 01 Совпадения нет	001 01 Совпадения нет	011 01 **Совпадение есть!**	Код 01 для буквы Д непригоден
11	11 1 **Совпадение есть!**	000 11 Совпадения нет	001 11 Совпадения нет	011 11 Совпадения нет	Код 01 для буквы Д непригоден
010	010 1 Совпадения нет	000 010 Коды не равны	001 010 Коды не равны	011 010 Коды не равны	**Код 011 для буквы Д пригоден**

Таким образом, допустимым для буквы Д является код 010. Это единственный возможный вариант, поэтому условие о том, что кодовое слово должно иметь минимально возможную длину, в данном случае не требуется.

Ответ: код 010 (вариант №4).

Задача 3. Для кодирования некоторой последовательности, состоящей из букв A, B, C, D и E, используется неравномерный двоичный код, позволяющий однозначно декодировать двоичную последовательность. Использован следующий код: A—110, B—100, C—01, D—11. Каким кодом должна быть закодирована буква E. Длина этого кода должна быть наименьшей из возможных. Код должен удовлетворять свойству однозначного декодирования.

1) 10 2) 0 3) 00 4) 010

Решение

Обратим внимание, что для исходных кодов: A—110, B—100, C—01, D—11 не выполняется прямое правило Фано, так как код буквы D (11) совпадает с началом кода буквы A (110). В этом случае нужно проверить, выполняется ли для этих кодов обратное правило Фано:

— коды букв A и B не равны; коды букв C и D также не равны;
— код буквы C (01) не совпадает с окончанием ни кода буквы A (110), ни буквы B (100);
— код буквы D (11) также не совпадает с окончанием ни кода буквы A (110), ни буквы B (100).

Следовательно, для данного набора кодов выполняется обратное правило Фано; оно и должно использоваться при решении задачи.

Проверяемый код буквы Е	Существующие коды букв А, В, С и D				Вывод
	А	В	С	D	
	110	100	01	11	
10	110 10 **Совпадение есть!**	100 10 Совпадения нет	01 10 Коды не равны	11 10 Коды не равны	Код 10 для буквы Е непригоден
0	110 0 **Совпадение есть!**	100 0 **Совпадение есть!**	01 0 Совпадения нет	11 0 Совпадения нет	Код 0 для буквы Е непригоден

Окончание табл.

Проверяемый код буквы E	Существующие коды букв A, B, C и D				Вывод
	A	B	C	D	
	110	100	01	11	
00	110 00 Совпадения нет	100 00 **Совпадение есть!**	01 00 Коды не равны	11 00 Коды не равны	Код 00 для буквы E непригоден
010	110 010 Коды не равны	100 010 Коды не равны	010 01 Совпадения нет	010 11 Совпадения нет	**Код 010 для буквы E пригоден**

Таким образом, допустимым для буквы E является код 010. Это единственный возможный вариант, поэтому условие о том, что кодовое слово должно иметь минимально возможную длину, в данном случае не требуется.

Ответ: код 010 (вариант №4).

Рекомендуется начинать решение задач такого типа с анализа выполнимости правил Фано для исходных кодов, указанных в условии задачи (т.е. без учёта искомого кода в вариантах ответов). В зависимости от того, какое из двух правил Фано выполняется для исходных кодов, при дальнейшем решении задачи производится сравнение более короткого кода с началом (при выполнении прямого правила Фано) или с концом (при выполнении обратного правила Фано) более длинного кода.

Задача 4*. Для 5 букв латинского алфавита заданы их двоичные коды (для некоторых букв — из двух бит, для некоторых — из трёх). Эти коды представлены в таблице:

A	B	C	D	E
000	01	100	10	011

Определить, какой набор букв закодирован двоичной строкой 0110100011000.

1) EBCEA 2) BDDEA 3) BDCEA 4) EBAEA

Решение

Наиболее оптимальным является использование при решении подобных задач правил Фано:

— если для заданных кодов выполняется прямое правило Фано, то декодирование нужно выполнять с начала двоичной строки (слева направо);

— если для заданных кодов выполняется обратное правило Фано, то декодирование нужно выполнять с конца двоичной строки (справа налево).

Таким образом, удаётся сразу определить направление однозначного декодирования заданной двоичной строки. После этого остаётся выполнить декодирование двоичной строки путём подстановки заданных кодов, и поскольку выполняется соответствующее правило Фано, на каждом шаге возможна подстановка только одного кода (расшифровка выполняется однозначно).

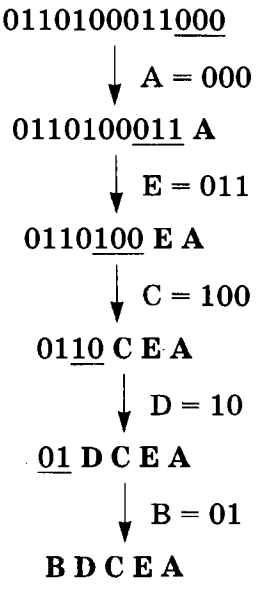
1. Совпадения кодов одинаковой длины (букв A, C и E, а также букв B и D) отсутствуют.

2. Код буквы B (01) совпадает с началом кода буквы E (011), а код буквы D (10) совпадает с началом кода буквы C (100), следовательно, для данной комбинации кодов *прямое правило Фано не выполняется.*

3. Код буквы B (01) не совпадает с концом кода ни одной из букв: A (000), C (100), E (011); код буквы D (10) также не совпадает с концом кода ни одной из букв: A (000), C (100), E (011), следовательно, для данной комбинации кодов *выполняется обратное правило Фано.*

Следовательно, декодирование двоичной строки нужно выполнять с конца, справа налево.

4. Построение дерева декодирования (напомним коды букв: А—000, В—01, С—100, D—10, Е—011):



Ответ: строка BDCEA (вариант №3).

Если для заданной последовательности кодов выполняется прямое правило Фано, то её декодирование необходимо вести с начала (слева направо).

Если для заданной последовательности кодов выполняется обратное правило Фано, то её декодирование необходимо вести с конца (справа налево).

Задача 5*. Для 5 букв русского алфавита заданы их двоичные коды (для некоторых букв — из двух бит, для некоторых — из трёх). Эти коды представлены в таблице:

В	К	А	Р	Д
000	11	01	001	10

Из четырёх полученных сообщений в этой кодировке, только одно прошло без ошибки и может быть корректно декодировано. Найдите его:

1) 110100000100110011
2) 111010000010010011
3) 110100001001100111
4) 110110000100110010

Решение

Принцип решения аналогичен предыдущей задаче.

1. Совпадения кодов одинаковой длины (букв К, А и Д, а также букв В и Р) отсутствуют.

2. Код буквы К (11) не совпадает с началом кода ни одной из букв: В (000) и Р (001); код буквы А (01) не совпадает с началом кода ни одной из букв: В (000) и Р (001); код буквы Д (10) также не совпадает с началом кода ни одной из букв: В (000) и Р (001), следовательно, для данной комбинации кодов *выполняется прямое правило Фано.*

3. Код буквы А (01) совпадает с концом кода буквы Р (001), следовательно, для данной комбинации кодов *обратное правило Фано не выполняется.*

Следовательно, декодирование двоичной строки нужно выполнять с начала, слева направо.

4. Построение дерева декодирования для каждого из проверяемых вариантов ответа (коды букв: В—000, К—11, А—01, Р—001, Д—10):

1) 110100000100110011

↓ К = 11

К 0100000100110011

↓ А = 01

К А 000000100110011

↓ В = 000

К А В 00100110011

↓ Р = 001

К А В Р 00110011

↓ Р = 001

К А В Р Р 10011

↓ Д = 10

К А В Р Р Д 011

✕

Код 011 не существует

Следовательно, вариант ответа №1 неправилен.

2) 111010000010010011

↓ К = 11

К 1010000010010011

↓ Д = 10

К Д 100000010010011

↓ Д = 10

К Д Д 000010010011

↓ В = 000

К Д Д В 010010011

↓ А = 01

К Д Д В А 0010011

↓ Р = 001

К Д Д В А Р 0011

↓ Р = 001

К Д Д В А Р Р 1

✕

Код 1 не существует

Следовательно, вариант ответа №2 неправилен.

3) 110100001001100111

↓ К = 11

К 0100001001100111

↓ А = 01

К А 00001001100111

↓ В = 000

К А В 01001100111

↓ А = 01

К А В А 001100111

↓ Р = 001

К А В А Р 100111

↓ Д = 10

К А В А Р Д 0111

↓ А = 01

К А В А Р Д А 11

↓ К = 11

К А В А Р Д А К

Двоичная строка успешно расшифрована, следовательно, вариант ответа №3 является правильным.

4) Оставшийся, четвёртый вариант ответа можно уже не проверять, однако для полноты решения, дерево декодирования для него имеент вид:

110110000100110010

↓ К = 11

К 0110000100110010

↓ А = 01

К А 10000100110010

↓ Д = 10

К А Д 000100110010

↓ В = 000

К А Д В 100110010

↓ Д = 10

К А Д В Д 0110010

↓ А = 01

К А Д В Д А 10010

↓ Д = 10

К А Д В Д А Д 010

↓ А = 01

К А Д В Д А Д А 0

✕

Код 0 не существует

Следовательно, вариант ответа № 4 неправилен

Ответ: правильным является вариант ответа №3 (110100001001100111).

Задача 5*. Для 5 букв латинского алфавита заданы их двоичные коды (для некоторых букв — из двух бит, для некоторых — из трёх). Эти коды представлены в таблице:

A	B	C	D	E
011	01	100	10	110

Определить, какой набор букв мог быть закодирован двоичной строкой 1000110110110.

1) CAADE
2) CABDE
3) BEDAC
4) CABED

Решение

1. Совпадения кодов одинаковой длины (букв A, C и E, а также букв B и D) отсутствуют.

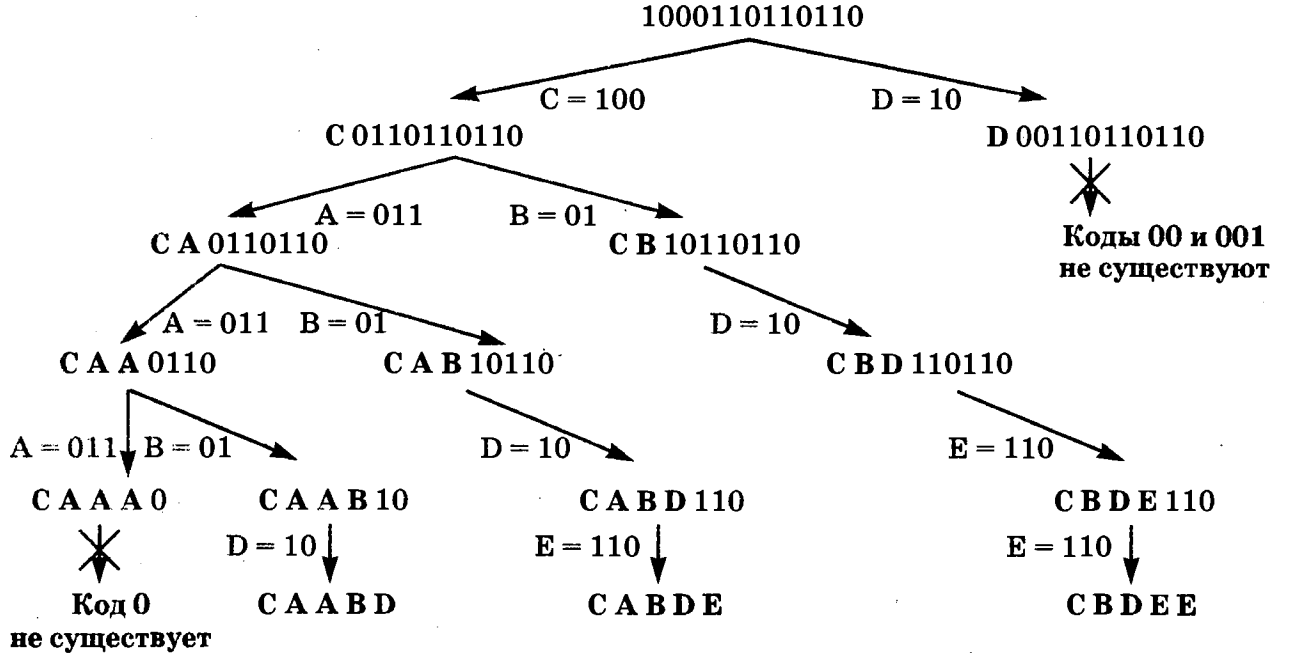
2. Код буквы B (01) совпадает с началом кода буквы A (011), а код буквы D (10) совпадает с началом кода буквы C (100), следовательно, для данной комбинации кодов *прямое правило Фано не выполняется.*

3. Код буквы D (10) также совпадает с концом кода буквы E (110), следовательно, для данной комбинации кодов *обратное правило Фано также не выполняется.*

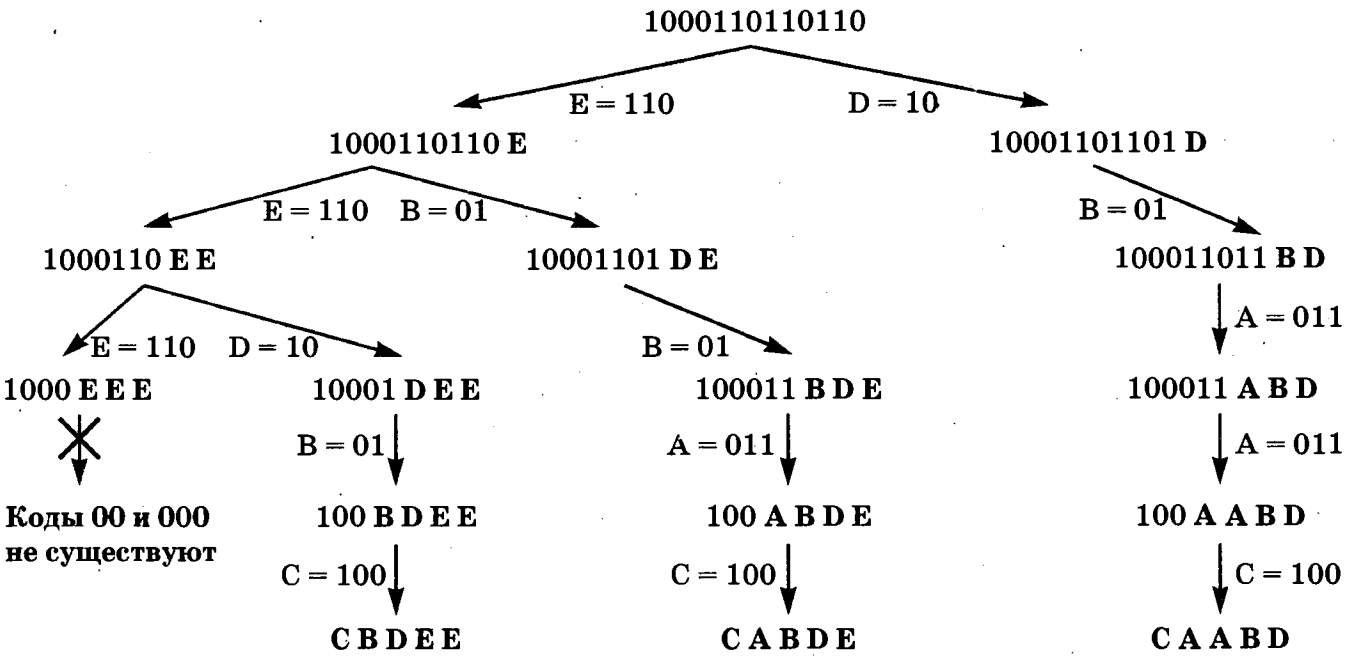
Следовательно, необходимо попытаться построить дерево декодирования для обоих возможных вариантов (с начала и с конца строки) в расчёте, что данная конкретная двоичная строка поддаётся декодированию несмотря на невыполнение правил Фано для исходных кодов.

4. Построение дерева декодирования (коды букв: A—011, B—01, C—100, D—10, E—110).

1) Попытка декодирования с начала строки (слева направо):



2) Попытка декодирования с конца строки (справа налево):



5. Поскольку ни одно из двух правил Фано не выполнялось, декодирование заданной двоичной строки выполняется неоднозначно (однако в результате обеих попыток декодирования получены одни и те же результаты). Все получившиеся комбинации: CAABD, CABDE, CBDEE — сравниваются с предлагаемыми вариантами ответов: 1) CAADE, 2) CABDE, 3) BEDAC, 4) CABED.

Ответ: строка CABDE (вариант ответа №2).

Задачи для самостоятельного решения

1. Для кодирования некоторой последовательности, состоящей из букв А, Б, В, Г и Д, решили использовать неравномерный двоичный код, позволяющий однозначно декодировать двоичную последовательность, появляющуюся на приёмной стороне канала связи. Использовали код: А—0, Б—111, В—110, Г—100. Укажите, каким кодовым словом должна быть закодирована буква Д.
   Длина этого кодового слова должна быть наименьшей из всех возможных. Код должен удовлетворять свойству однозначного декодирования.
   1) 11 3) 00
   2) 1010 4) 101
2. Для кодирования текстовой строки из букв A, B, C, D и E используется неравномерный двоичный код, позволяющий однозначно декодировать получаемую двоичную последовательность. Для букв A, B, C и D использованы следующие коды: A—100, B—101, C—111, D—110.
   Каким кодом из перечисленных ниже может быть закодирована буква E?
   Код должен удовлетворять свойству однозначного декодирования. Если можно использовать более одного кода, то укажите самый короткий.
   1) 000 3) 10
   2) 01 4) 11

3. Для кодирования последовательности из букв A, B, C, D и E используется неравномерный двоичный код, позволяющий однозначно декодировать получаемую двоичную последовательность. При этом используются следующие коды: A—011, B—010, C—000, D—001. Укажите, каким кодом (из перечисленных ниже) может быть закодирована буква E.

1) 1 3) 01
2) 0 4) 00

4. Для кодирования последовательности из букв A, B, C, D и E используется неравномерный двоичный код, позволяющий однозначно декодировать получаемую двоичную последовательность. При этом используются коды: A—00, B—10, C—110, D—111. Каким кодом должна быть закодирована буква E? Длина этого кода должна быть наименьшей из возможных.

Код должен удовлетворять свойству однозначного декодирования.

1)1 3) 010
2) 01 4) 011

5. Для кодирования последовательности из букв A, B, C, D и E используется неравномерный двоичный код, позволяющий однозначно декодировать получаемую двоичную последовательность. При этом используются следующие коды: A—1110, B—0, C—10, D—110. Каким кодом может быть закодирована буква E? Код должен удовлетворять свойству однозначного декодирования.

1) 0001
2) 0011
3) 0111
4) 1111

6. Для кодирования последовательности из букв A, B, C, D и E используется неравномерный двоичный код, позволяющий однозначно декодировать получаемую двоичную последовательность. Для букв A, B, C и D используются такие коды: A—011, B—010, C—000, D—001. Каким кодом из перечисленных ниже **НЕ может** быть закодирована буква E?

1) 1 3) 11
2) 0 4) 10

7. Для кодирования последовательности из букв A, B, C, D и E используется неравномерный двоичный код, позволяющий однозначно декодировать получаемую двоичную последовательность. При этом используются следующие коды: A—100, B—101, C—111, D—110.

Каким кодом из перечисленных ниже может быть закодирована буква E?

Код должен удовлетворять свойству однозначного декодирования. Если можно использовать более одного кода, то укажите кратчайший из них.

1) 1 3) 01
2) 0 4) 10

8. Для кодирования последовательности из букв A, B, C, D и E используется неравномерный двоичный код, позволяющий однозначно декодировать получаемую двоичную последовательность. При этом используются следующие коды: A—011, B—010, C—000, D—001.

Каким кодом из перечисленных ниже может быть закодирована буква E?

Код должен удовлетворять свойству однозначного декодирования. Если можно использовать более одного кода, то укажите кратчайший из них.

1) 1 3) 01
2) 0 4) 10

9. Для 5 букв русского алфавита заданы их двоичные коды (для некоторых букв — из двух бит, для некоторых — из трёх). Эти коды представлены в таблице:

Б	К	О	Т	Л
00	10	110	011	111

Из четырёх полученных сообщений в этой кодировке только одно прошло без ошибки и может быть корректно декодировано. Найдите его:

1) 101100111101110
2) 101101111100010
3) 101101100010001
4) 101101111101110

10. Майор Пронин ведёт наблюдение за шпионом. Ему удалось узнать пароль, который шпион должен назвать для входа в конспиративную квартиру. Для передачи сообщений в полицейское управление Пронин использует код, фрагмент которого представлен в таблице:

Г	И	В	Л	К	О	А
0101	1010	00	001	1101	11	110

В полицейском управлении получили от Пронина закодированное сообщение: 00110010101110. Какому паролю оно соответствует?

1) ВИЛОК
2) ОКОВА
3) ВОЛГА
4) ГОЛОВА

Ответы для самопроверки

№ задания	Ответ
1	4
2	2
3	1
4	2
5	4
6	2
7	2
8	1
9	2
10	3

Информация. Измерение информации. Кодирование информации

B10 Передача информации по коммуникационным каналам

Конспект

Передача информации

Передача информации — это информационный процесс, при котором производится перемещение информации через пространство и/или через время от одного субъекта (*источника информации*) к другому субъекту (*приёмнику информации*).

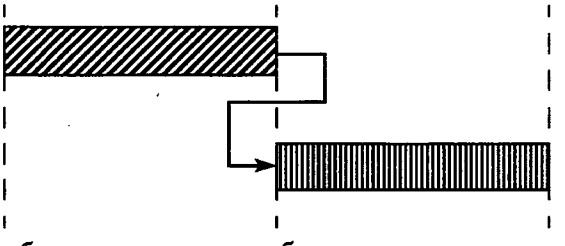
Скорость передачи информации по каналу связи (обычно рассматривается только передача сообщений) вычисляется как количество информации, переданной за одну секунду. Базовой единицей при этом является «бит в секунду» (бит/с, bits per second, bps); может также использоваться размерность «байт в секунду» и производные от неё величины (Кб/с, Мб/с и пр.).

Объём переданной информации Q вычисляется по формуле $Q = q \cdot t$, где q — пропускная способность канала (в битах в секунду или подобных единицах), а t — время передачи.

Диаграммы процессов (сетевые диаграммы, диаграммы Ганта)

В некоторых задачах, связанных с процессами передачи информации, особенно в случаях, когда один процесс начинается по истечении заданного времени после начала другого, можно существенно облегчить их решение благодаря его наглядному представлению с помощью **диаграмм процессов**. Такие диаграммы также называют **диаграммами Ганта** — по имени изобретателя таких диаграмм, американского инженера, механика и специалиста по менеджменту Генри Лоуренса Ганта.

Типичная диаграмма Ганта представляет собой отрезки (или прямоугольные полоски), размещенные вдоль горизонтальной шкалы времени, где каждый отрезок соответствует отдельной задаче или процессу. Начало, конец и длина каждого такого отрезка соответствуют началу, концу и длительности соответствующего процесса, а сами такие отрезки обычно располагаются друг за другом со сдвигом по вертикали.



Для некоторых задач удобно рисовать подобную диаграмму, изображающую два (или более) процесса и размечая на ней временные отметки их начал и окончаний. Применение диаграммы Ганта для решения задач будет рассмотрено ниже.

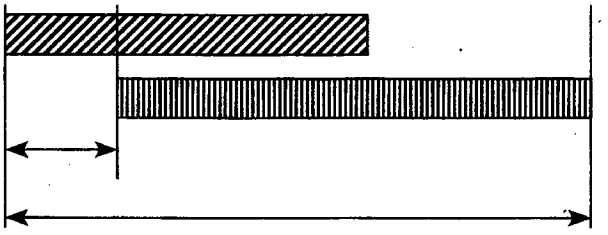
Разбор типовых задач

Задача 1*. У Кати есть доступ в Интернет по высокоскоростному одностороннему радиоканалу, обеспечивающему скорость получения информации 2^{20} бит в секунду. У Сергея нет скоростного доступа в Интернет, но есть возможность получать информацию от Кати по телефонному каналу со средней скоростью 2^{13} бит в секунду. Сергей договорился с Катей, что она скачает для него данные объёмом 9 Мбайт по высокоскоростному каналу и ретранслирует их Сергею по низкоскоростному каналу.

Компьютер Кати может начать ретрансляцию данных не раньше, чем им будут получены первые 1024 Кбайт этих данных. Каков минимально возможный промежуток времени (в секундах) с момента начала скачивания Катей данных до полного их получения Сергеем?

Решение

Прежде всего, создаётся набросок сетевой диаграммы, соответствующей условию задачи. В данном случае рассматривается два процесса передачи, осуществляемые с разной скоростью, из которых один процесс начинается спустя заданное время после начала другого. Поэтому общий вид диаграммы будет таким:



Процесс 1: компьютер Кати скачивает файл объёмом в 9 Мб ($= 9 \cdot 2^{20}$ байт $= 9 \cdot 2^{23}$ бит) со скоростью 2^{20} бит/с.

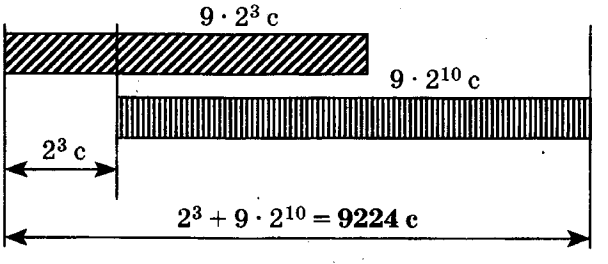
Длительность процесса 1: $9 \cdot 2^{23} / 2^{20} = 9 \cdot 2^3$ с.

Процесс 2: компьютер Сергея скачивает файл объёмом в 9 Мб ($=9 \cdot 2^{23}$ бит) со скоростью 2^{13} бит/с.

Длительность процесса 2: $9 \cdot 2^{23} / 2^{13} = 9 \cdot 2^{10}$ с.

Начало процесса 2 — через время, равное времени скачивания со скоростью 2^{20} бит/с информации объёмом 1024 Кб ($= 1024 \cdot 2^{10}$ байт $= 2^{10} \cdot 2^{10}$ байт $= 2^{20}$ байт $= 2^{23}$ бит), т. е. через $2^{23} / 2^{20} = 2^3$ с.

Сетевая диаграмма с надписанными результатами расчётов:



Благодаря сетевой диаграмме нетрудно определить, какие величины нужно суммировать для вычисления общей длительности процесса (т. е. времени, прошедшего с момента начала скачивания Катей данных до полного их получения Сергеем).

Ответ: 9224 с.

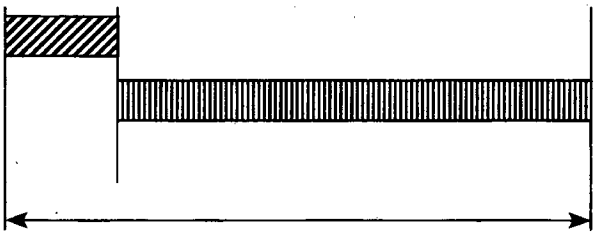
Следует не забывать приводить все величины, указанные в условии задачи, к одной размерности! Например, если скорость передачи информации задана в битах в секунду, то все значения объёмов информации нужно преобразовать в биты.

Задача 2. У вас есть высокоскоростной спутниковый доступ в Интернет, обеспечивающий получение данных со скоростью 4 Мбит/с в направлении от сервера к вам. Запросы с вашего компьютера передаются на сервер через подключенный к компьютеру сотовый телефон, выполняющий функции модема. При этом обеспечивается скорость передачи информации 128 Кбит/с. Вам нужно скачать файл с музыкальной записью объёмом 16 Мбайт. Информация по спутниковому каналу передаётся с сервера пакетами объёмом не более 1 Мбайта. Для получения каждого Мбайта ваш компьютер должен сначала передать в сеть запрос суммарным объёмом 8 Кбайт. За какое минимально возможное число секунд вы можете скачать весь требуемый файл?

Решение

Несмотря на то, что эта задача выглядит более сложной, чем предыдущая, принцип её решения в целом тот же.

Общий вид сетевой диаграммы:



Процесс 1: передача запроса суммарным объёмом 8 Кбайт ($= 2^3 \cdot 2^{10}$ байт $= = 2^3 \cdot 2^{13}$ бит) через сотовый телефон со скоростью 128 Кбит/с ($= 2^7 \cdot 2^{10}$ бит/с).

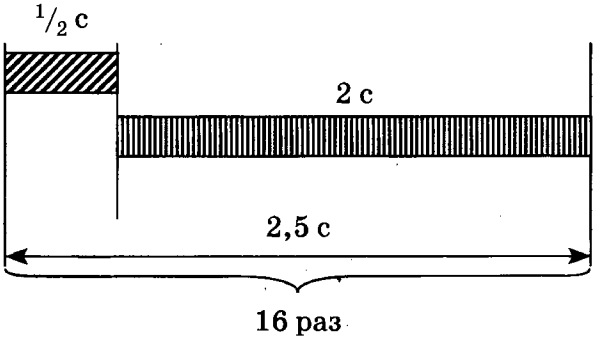
Длительность процесса 1: $2^3 \cdot 2^{13} / 2^7 \cdot 2^{10} = 2^{16} / 2^{17} =$ S с.

Процесс 2: приём очередного пакета файла объёмом 1 Мбайт ($= 2^{20}$ байт $= 2^{23}$ бит) через спутниковый канал со скоростью 4 Мбит/с ($= 2^2 \cdot 2^{20}$ бит/с).

Длительность процесса 1: $2^{23} / 2^2 \cdot 2^{20} = 2^{23} / 2^{22} = 2$ с.

Файл объёмом 16 Мбайт принимается порциями по 1 Мбайт, следовательно, весь процесс приёма информации состоит из 16 элементарных процессов, рассмотренных выше (передача команд + приём очередного фрагмента).

Сетевая диаграмма с надписанными результатами расчётов:



Итого для приёма всего файла потребуется $16 \cdot 2{,}5 = 40$ с.

Ответ: 40 с.

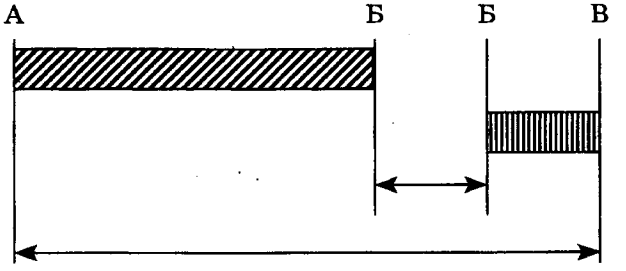
Лучше разделить описываемый процесс передачи информации на отдельные повторяющиеся этапы, чтобы вначале определить длительность отдельного такого этапа, а затем вычислить общую длительность процесса.

Задача 3. Между городами Аркадия и Виктория нет прямого канала связи. Данные из Аркадии в Викторию передаются через промежуточную станцию связи Белла с помощью соответствующих каналов связи АБ и БВ. Скорость передачи данных по каналу АБ в 4 раза ниже, чем скорость передачи данных по каналу БВ. Передача данных со станции Белла в Викторию начинается через 10 секунд после того, как закончился приём этих данных станцией Белла из Аркадии.

Сколько секунд потребуется для передачи из Аркадии в Викторию пакета данных, если для его передачи из Аркадии на станцию Белла потребовалось 1 минута 20 секунд?

Решение

Общий вид сетевой диаграммы:



Процесс 1: передача данных по каналу А—Б со скоростью j скорости передачи данных по каналу Б—В.

Длительность процесса 1: 1 мин 20 с = 80 с.

Процесс 2: передача данных по каналу Б—В со скоростью v (неизвестная величина).

Начало процесса 2 — через 10 с после завершения передачи информации по каналу А—Б.

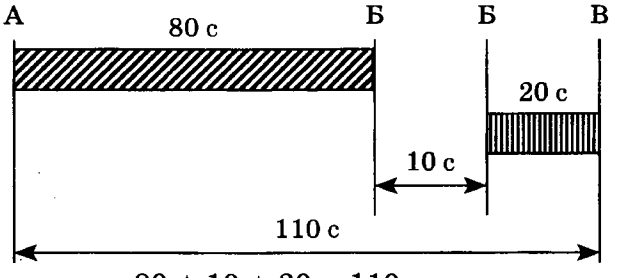
1. Объём передаваемых данных X определяется из условия, что длительность передачи данных (80 с) по каналу А—Б достигается при скорости передачи jv:

$X = 80 \cdot v / 4 = 20v$.

2. Тогда длительность передачи данных по каналу Б—В определяется уравнением:

$T_2 = X / v = 20$ с.

Сетевая диаграмма с надписанными результатами расчётов:



Общая длительность процесса: 80 + 10 + 20 = 110 с.

Ответ: 110 с.

Задача 4. Документ объёмом 10 Мбайт можно передать с одного компьютера на другой двумя способами:

А) Сжать архиватором, передать архив по каналу связи, распаковать.

Б) Передать по каналу связи без использования архиватора.

Какой способ быстрее и насколько, если:

— средняя скорость передачи данных по каналу связи составляет 2^{18} бит в секунду,

— объём сжатого архиватором документа равен 30% от исходного,

— время, требуемое на сжатие документа — 5 секунд, на распаковку — 1 секунда?

В ответе напишите букву А, если способ А быстрее или Б, если быстрее способ Б. Сразу после буквы напишите количество секунд, насколько один способ быстрее другого. Например, если способ Б быстрее способа А на 23 секунды, в ответе нужно написать «Б23».

Решение (вариант 1)

В данной задаче не требуется строить сетевую диаграмму, — только расчёты нужно выполнить дважды, для обоих предложенных вариантов (А и Б).

Вариант А:

Предполагается, что исходный массив информации сначала архивируется (5 секунд). Затем он в уже упакованном виде (30% от исходного объёма) передаётся по каналу связи с заданной средней скоростью, а после этого распаковывается (1 секунда).

Соответствующая цепочка вычислений:

$$5 + (0{,}3 \cdot 10 \cdot 2^{20} \cdot 2^3 / 2^{18}) + 1$$

мегабайты — 10

байты — $2^{20} \cdot 2^3$

биты — $/ 2^{18}$

Далее можно после некоторого количества арифметических операций вычислить длительность всего процесса в секундах, — но мы пока подождем это делать (почему — будет ясно чуть позже).

Вариант Б:

Аналогичным образом можно записать вычисления времени передачи неупакованного массива информации по каналу связи:

$$10 \cdot 2^{20} \cdot 2^3 / 2^{18}$$

мегабайты — 10

байты — $2^{20} \cdot 2^3$

биты — $/ 2^{18}$

Выполнив вычисления и сравнив результат с полученным для варианта А, определяется ответ к задаче.

Решение (вариант 2)

Можно решить эту задачу более экономно — выполнив вычисления только один раз.

Сравним предлагаемые варианты (А и Б). Очевидно, что 30% от исходного объёма массива информации приходится передавать по каналу связи в обоих случаях. А в чём различие?

В варианте А, кроме этих 30% объёма файла, по каналу связи больше ничего передавать не требуется, но зато имеются «накладные расходы» времени на упаковку/распаковку в количестве 6 секунд.

В варианте Б этих дополнительных расходов времени нет, но зато приходится передавать по каналу связи оставшиеся 70% объёма файла.

Получается, что для решения задачи достаточно сравнить между собой затраты времени именно на эти различающиеся в вариантах А и Б операции:

6 секунд на упаковку/распаковку (*VS*) передача 70% объёма файла

Другими словами, достаточно вычислить время передачи по каналу связи с заданной скоростью 70% исходного объёма информации и вычесть из этого времени 6 секунд:

- если результат отрицателен, то быстрее способ Б;
- если результат положителен, то быстрее способ А;
- если результат равен нулю, то оба способа равнозначны по скорости;
- абсолютное значение (модуль) разности — это и есть искомое время, на которое один способ быстрее другого.

В данном случае:

$$0{,}7 \cdot 10 \cdot 2^{20} \cdot 2^3 / 2^{18} - 6 =$$
$$= 0{,}7 \cdot 5 \cdot 2^{24} / 2^{18} - 6 = 3{,}5 \cdot 2^6 - 6 = 3{,}5 \cdot 64 - 6 = 218 \text{ с.}$$

Ответ: А218 (способ А быстрее способа Б на 218 секунд).

Одной из технологий, лежащих в основе функционирования сети Интернет, является пакетная передача данных: каждый файл перед его пересылкой разбивается на отдельные фрагменты (кадры); каждый кадр пересылается отдельно, после приёма проверяется на наличие ошибок и в зависимости от результатов проверки принимающий компьютер отправляет передающему запрос на повторную передачу ошибочного кадра либо подтверждение о безошибочном приёме кадра; после безошибочного приёма всех кадров они вновь соединяются в единый файл.

Задача 5. В используемом варианте сетевого протокола каждый кадр содержит в себе 512 байт информации из передаваемого файла плюс 128 байт служебной информации (IP-адреса отправителя и получателя, контрольная сумма и т.д.). Последний передаваемый кадр может быть неполным и содержать менее 512 байт информации из передаваемого файла при том же объёме добавляемой служебной информации. Подтверждение о безошибочном приёме кадра имеет объём 32 байта (служебная информация). Канал передачи информации является настолько надёжным, что все кадры передаются без ошибок.

Требуется передать файл объёмом 4000 байт. Канал связи является асинхронным: в направлении от источника к получателю файла достигается скорость 10 Кбит/с, а в обратном направлении — скорость 2 Кбит/с.

Определить минимальное время, за которое может быть передан весь файл.

Решение

1. Файл разбивается на фрагменты по 512 байт (последний фрагмент может быть неполным):

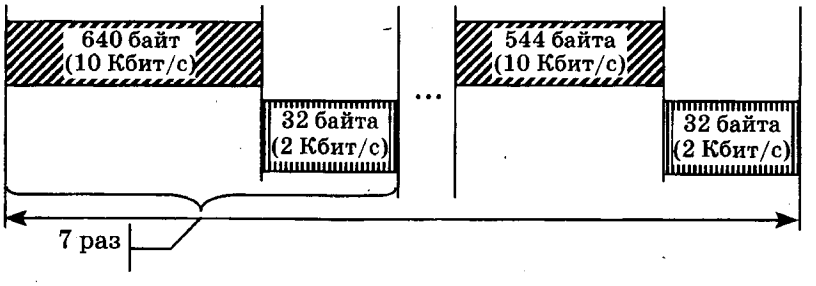
$4000 = 7 \cdot 512 + 416$.

Итого имеем 8 кадров, из которых 8-й неполный.

2. Объём полного кадра равен: 512 + 128 = 640 байт. Объём последнего (неполного) кадра равен: 416 + 128 = 544 байта.

3. Весь процесс состоит из 7 повторений элементарных процессов: передача полного кадра со скоростью 10 Кбит/с + обратная передача подтверждения (32 байта) со скоростью 2 Кбит/с и из одного элементарного процесса: передача неполного кадра со скоростью 10 Кбит/с + обратная передача подтверждения (32 байта) со скоростью 2 Кбит/с.

Общий вид сетевой диаграммы:



Вычисления:

$$7 \cdot \left(\frac{640 \cdot 2^3}{10 \cdot 2^{10}} + \frac{32 \cdot 2^3}{2 \cdot 2^{10}}\right) + \left(\frac{544 \cdot 2^3}{10 \cdot 2^{10}} + \frac{32 \cdot 2^3}{2 \cdot 2^{10}}\right) = 4{,}925 \text{ с.}$$

Задача 6*. Скорость передачи данных через ADSL-соединение равна 128000 бит/с. Через данное соединение передают файл размером 625 Кбайт. Определите время передачи файла в секундах.

Решение

Объём передаваемой информации равен 625 Кб = $625 \cdot 2^{10}$ байт = $625 \cdot 2^{13}$ бит.

Скорость передачи информации: 128000 бит/с = $2^{10} \cdot 125$ бит/с.
Тогда время, затрачиваемое на передачу информации, равно:
$625 \cdot 2^{13}$ (бит) / $2^{10} \cdot 125$ (бит/с) = $5 \cdot 2^3 = 40$ с.

Ответ: 40 секунд.

Рекомендуется большие числа преобразовывать в произведения некоторой константы (не кратной 2) на соответствующую степень двойки. Это облегчает выполнение операций умножения и деления (благодаря возможности сокращения двоек) и уменьшает вероятность ошибок в вычислениях.

Задача 7*. Скорость передачи данных через ADSL-соединение равна 1024000 бит/с. Передача файла через данное соединение заняла 5 секунд. Определите размер файла в килобайтах.

Решение
Эта задача обратна предыдущей.
Скорость передачи информации равна 1024000 бит/с = $2^{13} \cdot 125$ бит/с.
Время, затраченное на передачу информации, равно 5 секунд.
Тогда объём передаваемой информации равен:
$125 \cdot 2^{13}$ (бит/с) $\cdot$ 5 (с) = $625 \cdot 2^{13}$ бит.
Преобразуя полученное значение в килобайты (согласно условию задачи):
$625 \cdot 2^{13}$ бит = $625 \cdot 2^{10}$ байт = 625 Кб.

Ответ: 625 Кб.

Задача 8*. Скорость передачи данных через ADSL-соединение равна 256000 бит/с. Передача файла через это соединение заняла 2 минуты. Определите размер файла в килобайтах.

Решение
Скорость передачи информации равна 256000 бит/с = $2^{11} \cdot 125$ бит/с.
Время, затраченное на передачу информации, равно 2 минутам = $2 \cdot 60 = 2^3 \cdot 15$ секунд.
Тогда объём передаваемой информации равен:
$125 \cdot 2^{11}$ (бит/с) $\cdot$ $15 \cdot 2^3$ (с) = $1875 \cdot 2^{14}$ бит.
Преобразуя полученное значение в килобайты (согласно условию задачи):
$1875 \cdot 2^{14}$ бит = $1875 \cdot 2^{11}$ байт = $1875 \cdot 2$ Кб = 3750 Кб.

Ответ: 3750 Кб.

Следует не забывать преобразовывать биты в байты, килобайты и т. д. и наоборот, а секунды — в минуты и обратно, чтобы соблюсти равенство единиц измерения величин!

Задача 9*. Известно, что длительность непрерывного подключения к сети Интернет с помощью модема для некоторых АТС не превышает 10 минут. Определите максимальный размер файла (в килобайтах), который может быть передан за время такого подключения, если модем передаёт информацию в среднем со скоростью 32 Кбит/с?

Решение
Здесь несколько изменена текстовая формулировка задачи, однако ее смысл остается прежним.
Скорость передачи информации равна 32 Кбит/с = $32 \cdot 2^{10}$ бит/с = 2^{15} бит/с.
Максимальное время, затрачиваемое на передачу информации, равно 10 минут = $10 \cdot 60 =$ $= 2^3 \cdot 75$ секунд.
Тогда максимально возможный объем информации, который можно передать за это время, равен:
2^{15} (бит/с) $\cdot$ $75 \cdot 2^3$ (с) = $75 \cdot 2^{18}$ бит.

Преобразовывая полученное значение в килобайты (согласно условию задачи):

$75 \cdot 2^{18}$ бит $= 75 \cdot 2^{15}$ байт $= 75 \cdot 2^5$ Кб $= 2400$ Кб.

Ответ: 2400.

Задача 10*. Сколько секунд потребуется модему, передающему сообщения со скоростью 28800 бит/с, чтобы передать цветное растровое изображение размером 640×480 пикселей, при условии, что цвет каждого пикселя кодируется тремя байтами?

Решение

Единственное отличие от предыдущих задач — в том, что предварительно требуется вычислить объём передаваемого файла.

Объём передаваемой информации равен:

$640 \cdot 480 \cdot 3$ байт $= 640 \cdot 480 \cdot 3 \cdot 2^3$ бит $= 5 \cdot 15 \cdot 3 \cdot 2^{15}$ бит $= 225 \cdot 2^{15}$ бит.

Скорость передачи информации: 28800 бит/с $= 2^7 \cdot 225$ бит/с.

Тогда время, затрачиваемое на передачу информации, равно:

$225 \cdot 2^{15}$ (бит) / $2^7 \cdot 225$ (бит/с) $= 2^8 = 256$ с.

Ответ: 256 секунд.

Задачи для самостоятельного решения

1. У Димы есть высокоскоростное подключение к Интернету по оптоволоконному каналу со скоростью 2^{22} бит в секунду. У Оли нет доступа в Интернет, но она может получать информацию с компьютера Димы через телефонный модем со средней скоростью 2^{16} бит в секунду. Оля договорилась с Димой, что он будет скачивать для неё аудиофайл объёмом 10 Мбайт по высокоскоростному каналу и пересылать их Оле через модем. Компьютер Димы может начать передачу данных только после того, как получит первые 512 Кбайт из исходного файла. Сколько (как минимум) секунд займёт передача данных с начала скачивания файла Димой до полного получения этого файла Олей? В ответе укажите только число.
2. Документ Word объёмом 10 Мбайт можно передать с одного компьютера на другой двумя способами:
   А. Сжать архиватором, передать архив, распаковать его.
   Б. Передать файл документа без использования архиватора.
   Какой способ быстрее и насколько, если
   — средняя скорость передачи данных по каналу связи составляет 2^{18} бит в секунду,
   — объём сжатого архиватором файла документа равен 30 % от исходного,
   — время, требуемое на сжатие документа — 7 секунд, на распаковку — 1 секунда?
   В ответе напишите букву А, если способ А быстрее, или Б, если быстрее способ Б. После буквы запишите количество секунд, насколько один способ быстрее другого. Например, если способ Б быстрее способа А на 23 секунды, то ответ должен иметь вид: Б23.
3. У Алексея есть доступ в Интернет по высокоскоростному радиоканалу со скоростью передачи информации 2^{20} бит в секунду. У Бориса есть возможность получать информацию от Алексея через модем со средней скоростью 2^{14} бит в секунду. Борис попросил Алексея скачать для него файл объёмом 10 Мбайт по высокоскоростному каналу и передать его Борису через модем. Компьютер Алексея может начать передачу данных не

раньше, чем им будут получены первые 1024 Кбайт этих данных. Сколько (как минимум) секунд пройдёт с начала скачивания файла Алексеем до полного их получения Борисом? В ответе укажите только число.

4. Файл объёмом 40 Мбайт передаётся из пункта Альфа в пункт Бета по каналу связи, обеспечивающему скорость передачи данных 2^{20} бит в секунду, а затем из пункта Бета в пункт Гамма по каналу связи, обеспечивающему скорость передачи данных 2^{22} бит в секунду. Задержка в пункте Бета (время между окончанием приёма данных из пункта Альфа и началом передачи в пункт Гамма) составляет 4 секунды. Сколько времени (в секундах) прошло с начала передачи данных из пункта Альфа до их полного получения в пункте Гамма? В ответе укажите только число.
5. Между пунктом Альфа и пунктом Гамма нет прямого канала связи. Данные из Альфы в Гамму передаются через промежуточный пункт Бета с помощью каналов связи АБ и БГ. Скорость передачи данных по каналу БГ в 2 раза ниже, чем скорость передачи данных по каналу АБ. Передача данных из Беты в Гамму начинается через 5 секунд после того, как закончился приём этих данных в Бете. Сколько времени (в секундах) потребуется для передачи из Альфы в Гамму пакета данных, если для передачи этого пакета из Альфы в Бету требуется 50 секунд?
   В ответе укажите только число.
6. Скорость передачи данных по каналу ADSL равна 128000 бит/с. Передача файла по этому каналу заняла 120 секунд. Каков объём этого файла в килобайтах?
7. Файл объёмом 3072000 байт пересылается по каналу связи со скоростью 50 Кбит/с. Сколько минут будет передаваться файл?
8. Документ Word объёмом 5 Мбайт можно передать с одного компьютера на другой двумя способами:
   А) сжать архиватором, передать архив, распаковать его;
   Б) передать файл по каналу связи без использования архиватора.
   Какой способ быстрее и насколько, если:
   — средняя скорость передачи данных по каналу связи составляет 2^{20} бит в секунду;
   — объём сжатого архиватором файла равен 20 % от исходного;
   — время, требуемое на сжатие файла — 10 секунд, на распаковку — 2 секунды?
   В ответе напишите букву А, если способ А быстрее, или Б, если быстрее способ Б. Сразу после буквы запишите, на сколько секунд один способ быстрее другого.
   Например, если способ Б быстрее способа А на 23 секунды, то ответ будет иметь вид: Б23.
9. Сколько секунд потребуется модему, передающему сообщения со скоростью 24 000 бит/с, чтобы передать 200 страниц текста в 50 строк по 60 символов каждая, при условии, что каждый символ кодируется 1 байтом?
10. Лена скачивает дистрибутив ОС Linux с зарубежного сайта-репозитория, пользуясь односторонним каналом цифровой передачи данных через телевизионное эфирное вещание, обеспечивающее приём информации со скоростью 4 Мбит/с. При этом информация передаётся фрагментами по 10 Мбайт. Для начала передачи каждого фрагмента компьютер Лены должен отправить на сервер сообщение-запрос объёмом 32 Кбайт, а после получения фрагмента подтвердить его безошибочный приём отдельным сообщением объёмом 16 Кбайт. Для отправки таких сообщений Лена пользуется радиомодемом GPRS, который обеспечивает скорость передачи информации до 128 Кбит/с. Определить минимально возможное время в секундах, за которое Лена сможет скачать файл дистрибутива объёмом 350 Мбайт.

Ответы для самопроверки

№ задания	Ответ
1	1281
2	A216
3	5128
4	404
5	155
6	1875
7	8
8	A20
9	200
10	805

Моделирование и компьютерный эксперимент

A2, B9 Задачи на графах

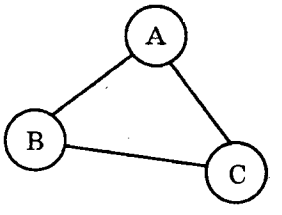
Конспект

Граф — один из способов графического представления информации, отражающий количество объектов изучаемой системы и взаимосвязи между ними.

Объекты, отраженные в графе, представлены в нём как **вершины (узлы)** графа, а связи между ними — как **дуги (рёбра)**. Таким образом, граф представляет собой совокупность непустого множества вершин и множества связей между вершинами.

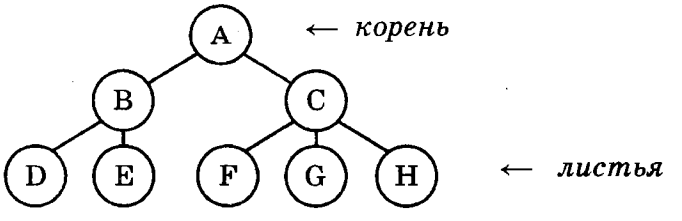
Количество вершин графа называют его **порядком**. Количество рёбер называют **размером** графа.

Путь (цепь) в графе — конечная последовательность вершин, каждая из которых (кроме последней) соединена со следующей ребром. **Циклом** называют путь, в котором первая и последняя вершины совпадают. Путь (или цикл) называют **простым**, если рёбра в нём не повторяются. Простой путь (цикл) называют **элементарным**, если вершины в нём не повторяются.



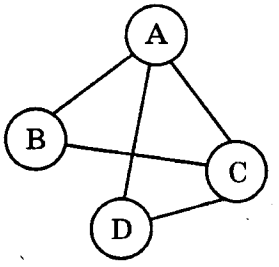
Дерево — граф, в котором существует один-единственный путь между любой парой вершин и не имеется ни одного цикла. Одна из вершин дерева (его **корень**) не имеет входящих в неё дуг, а все остальные вершины имеют ровно одну входящую дугу. При этом вершины, не имеющие исходящих из них дуг, называются **листьями**.

Пример дерева:



Способы представления графов

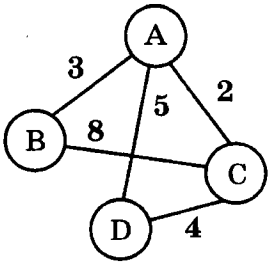
1. **Графический способ** — изображение графа.
Пример:



2. **Список рёбер** — перечисление всех рёбер графа как пар обозначений связываемых этими рёбрами вершин.
Пример: {A,B}, {A,D}, {A,C}, {B,C}, {C,D}

3. **Матрица смежности** — квадратная симметричная таблица (матрица), в которой и столбцы, и строки соответствуют вершинам графа, а в ячейках на их пересечении записываются числа, обозначающие наличие или отсутствие связей между соответствующими парами вершин (обычно — количество связей между вершинами).

В простейшем случае, когда граф не имеет кратных рёбер и петель, матрица смежности содержит единицы для ячеек, соответствующих парам вершин, связанных ребром, и нули — для несвязанных вершин.
Пример:



	A	B	C	D
A		3	2	5
B	3		8	
C	2			4
D	5		4	

Разбор типовых задач

Задача 1. Между городами *A, B, C, D, E, F* проложены дороги, протяжённость которых указана в таблице. Отсутствие числа в таблице означает, что дороги между соответствующими пунктами нет.

	A	B	C	D	E	F
A				1	4	3
B			4			5
C		4		2	1	
D	1		2			2
4	4		1			
F	3	5		2		

Определите длину кратчайшего пути между городами *A* и *B* (передвигаться можно только по имеющимся дорогам).

1) 6 2) 7 3) 8 4) 9

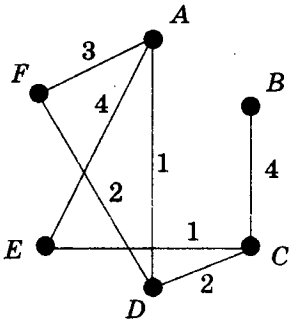
Решение

Это типичная задача на построение графов по заданной матрице смежности. Вершинами искомого графа (карты городов) являются названия городов, обозначенные буквами от *A* до *F*, а рёбра определяются наличием в таблице чисел, указывающих весá этих рёбер.

Построение такого графа является первым шагом решения задачи. Для этого достаточно разметить точки A, B, C, D, E, F и соединить линиями те из них, для которых на пересечении строки и столбца таблицы имеется непустая ячейка. Число, находящееся в этой ячейке, нужно записать над соответствующим ребром.

Кроме того, очевидно, что поскольку граф в данном случае не является ориентированным (в условии не указано, что можно двигаться по дорогам только в одном направлении, значит, возможно и встречное движение), его матрица смежности, зеркально симметрична относительно главной диагонали (выделенной серым фоном ячеек). Поэтому при рисовании графа достаточно просмотреть, например, только ячейки над главной диагональю.

Для данной таблицы (матрицы смежности) получается граф следующего вида:



Остаётся перебрать все возможные пути от вершины A к вершине B:

1) $AFDCB$ — длина пути: $3 + 2 + 2 + 4 = 11$;

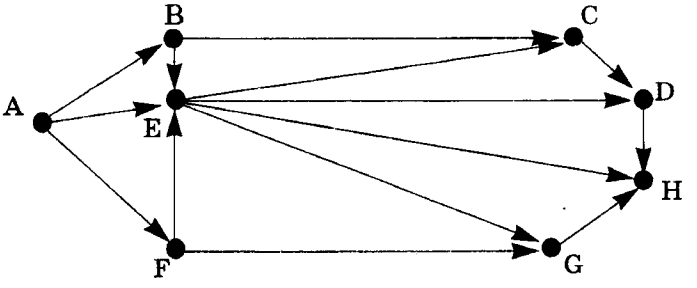
2) $AECB$ — длина пути: $4 + 1 + 4 = 9$;

3) $ADCB$ — длина пути: $1 + 2 + 4 = 7$.

Очевидно, что других путей прямо к вершине A нет. А из существующих самый короткий — путь $ADCB$ длиной 7 единиц.

Ответ: 7 (вариант ответа №2).

Задача 2. На рисунке показана схема дорог между городами A, B, C, D, E, F, G, H. По этим дорогам можно двигаться только в одном направлении, показанном стрелкой. Сколько возможно различных путей из города A в город H?

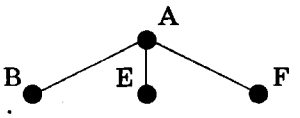


Решение

Предложенная схема дорог представляет собой ориентированный граф. Нужно проследить все возможные пути от вершины A до вершины H. Чтобы при этом не пропустить ни одного возможного варианта, строится второй граф — дерево вариантов. (Такой граф чем-то похож на деревья ходов игроков, которые требуется строить при решении задач С3, посвященных определению выигрышных ходов в играх «в камешки».)

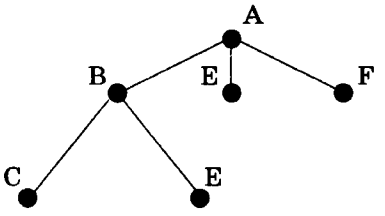
Построение подобного дерева надо начать с вершины A. К каким вершинам ведут рёбра исходного графа от вершины A? Очевидно, к вершинам B, E и F. Тогда вычерчивается в са-

мом начале дерева именно эта структура: «корневая» вершина A и от неё — три ветви к вершинам B, E и F:

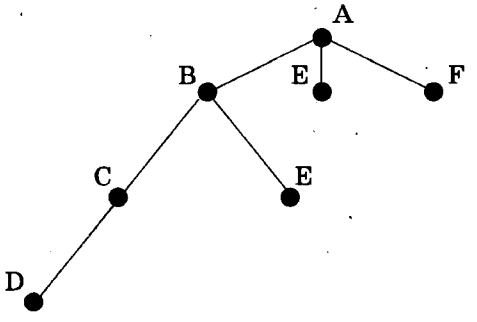


Теперь повторяется та же самая операция — выяснение, к скольким другим вершинам ведут рёбра, и вычерчивание соответствующих ветвей дерева — для каждой из получившихся концевых вершин (B, E и F), а затем — для новых концевых вершин, которые будут получаться на каждом таком новом шаге. При этом не нужно бояться того, что какие-то обозначения (буквы) будут повторяться в разных ветвях дерева. Построение каждой ветви нужно прекратить, если очередной вершиной окажется вершина с буквой H — конечный (по условию задачи) пункт путешествия. Тогда эта вершина для наглядности помечается другим цветом, жирностью шрифта, обводкой и т.п.

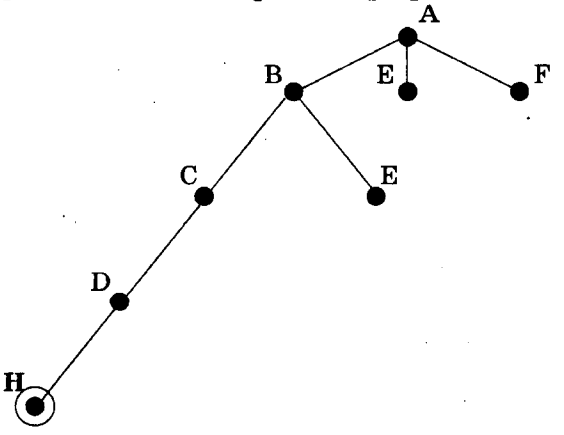
1) От вершины B можно проехать к вершинам C и E, — рисуется от неё две ветви (а ветвь к вершине A уже нарисована ранее):



2) От вершины C можно проехать только к вершине D, — вычерчивается только одна ветвь-продолжение.

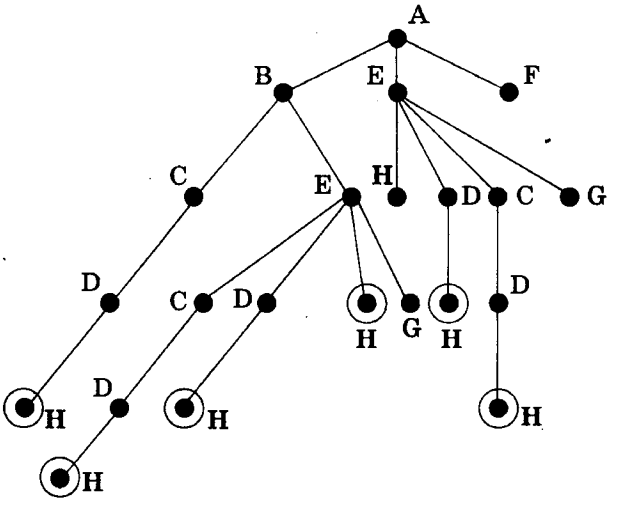


3) От вершины D можно попасть только в вершину H. Она является конечной целью путешествия, поэтому, например, помечается жирным шрифтом и обводится кружочком:

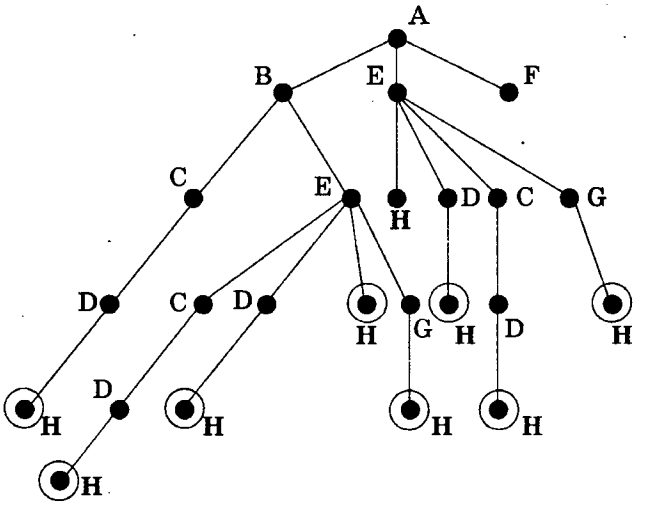


4) Учитывая, что в вершинах C и D дерево не имело ветвлений, возвращаемся к вершине E. От неё, согласно заданной схеме, можно попасть в вершины C, D, H и G. Соответственно, вычерчивается в дереве такой «пучок» ветвей дважды, поскольку вершина E в этом дереве

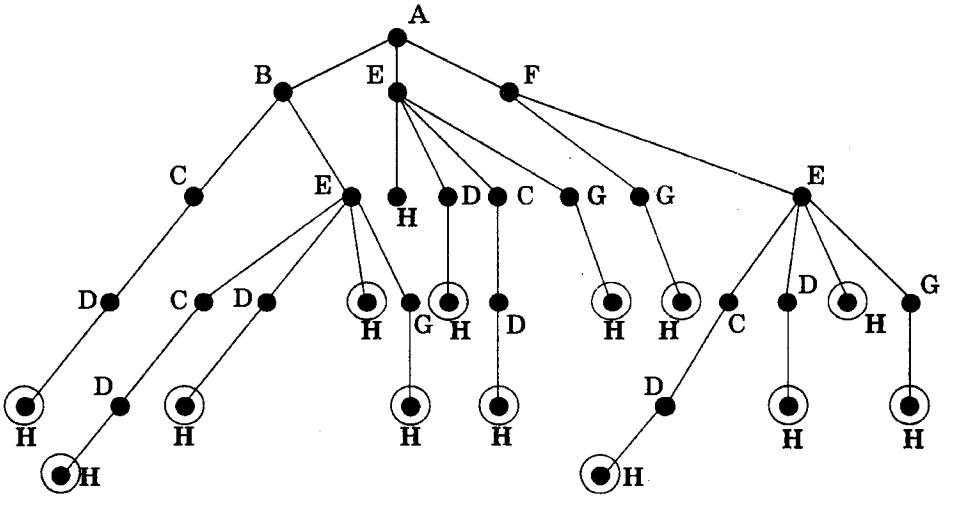
уже повторена дважды. А затем для полученных вершин C и D дочерчиваются ветви к конечной вершине H и везде помечается эта вершина жирным шрифтом и обводится кружочком:



5) Теперь переходим к вершине G. Аналогично D, из неё можно попасть только в H. Дочерчиваются соответствующие ветви:



6) Вернувшись к вершине F. Из неё можно попасть в вершины E и G, — чертятся эти ветви. От G можно перейти только в H, — рисуется эта ветвь. А от вершины E повторяется ещё раз «пучок» возможных путей. Полное дерево возможных путей от вершины A к вершине H тогда имеет вид:



Теперь, когда построенное дерево гарантирует, что не упущен ни один возможный вариант пути от A к H, остаётся только подсчитать количество этих вариантов. Сделать это очень просто: достаточно подсчитать количество «концевых» вершин H в этом дереве, которые заблаговременно выделяется жирным шрифтом. Их 14 штук.

Ответ: 14 возможных вариантов пути.

Построение дерева вариантов — это универсальный метод решения задач с перебором возможных вариантов (действий, решений, ходов и пр.), обладающий высокой наглядностью и существенно уменьшающий риск пропуска возможных вариантов.

Задача 3. Числа, указанные в таблицах, обозначают стоимость проезда между соответствующими соседними станциями, первые буквы названий указаны в соответствующем столбце и строке таблицы. Если число не указано, то это означает, что соответствующие станции не являются соседними.

Стоимость проезда по некоторому маршруту вычисляется как сумма стоимостей проезда между соответствующими соседними станциями.

Для какой таблицы из приведённых ниже выполняется условие: минимальная стоимость проезда по маршруту из D в A не больше 5?

1)

	A	B	C	D	E
A		2	3		6
B	2			3	
C	3			2	
D		3	2		3
E				3	

3)

	A	B	C	D	E
A		2	4		6
B	2			4	
C	4			2	
D		4	2		
E	6				

2)

	A	B	C	D	E
A		3	3		7
B	3			3	
C	3				
D		3			1
E	7			1	

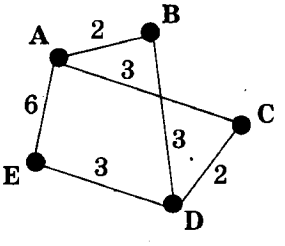
4)

	A	B	C	D	E
A		4	2		7
B	4			2	
C	2			6	
D		2	6		1
E	7			1	

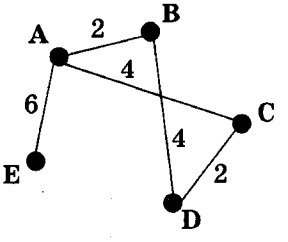
Решение

1. Поскольку графическое представление информации более наглядно, строятся по заданным таблицам соответствующие графы:

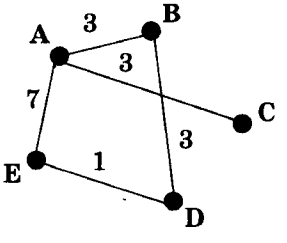
1)



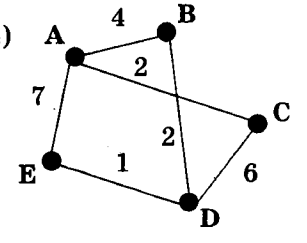
3)



2)



4)



2. Для каждого из четырёх полученных графов перебираются все возможные пути из D в A, подсчитывая суммарную стоимость проезда по каждому маршруту:

1) D-E-A (3 + 6 = 9); D-B-A (3 + 2 = 5); D-C-A (2 + 3 = 5);
2) D-E-A (1 + 7 = 8); D-B-A (3 + 3 = 6);
3) D-B-A (4 + 2 = 6); D-C-A (2 + 4 = 6);
4) D-E-A (1 + 7 = 8); D-B-A (2 + 4 = 6); D-C-A (6 + 2 = 8).

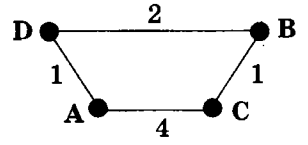
3. Определяется, для какого графа соблюдается условие «минимальная стоимость проезда по маршруту из D в A не больше 5»: очевидно, этому условию соответствует только вариант №1.

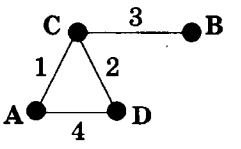
Ответ: вариант ответа №1.

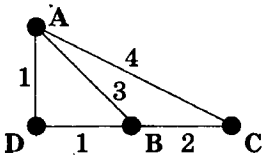
Поскольку матрица смежности симметрична относительно её главной диагонали, достаточно анализировать только одну её часть (например, над главной диагональю).

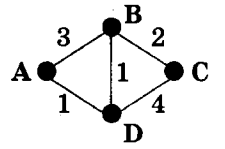
Задача 4. Укажите схему, соответствующую заданной таблице.

	A	B	C	D
A		3		1
B	3		2	1
C		2		4
D	1	1	4	

1)


3)


2)


4)


Решение

Принцип решения данной задачи является «обратным» по отношению к принципу решения предыдущей задачи: для каждого варианта графа нужно построить матрицу смежности и сравнить её с заданной таблицей.

1)

	A	B	C	D
A			4	1
B			1	2
C	4	1		
D	1	2		

3)

	A	B	C	D
A			1	4
B			3	
C	1	3		2
D	4		2	

2)

	A	B	C	D
A		3	4	1
B	3		2	1
C	4	2		
D	1	1		

4)

	A	B	C	D
A		3		1
B	3		2	1
C		2		4
D	1	1	4	

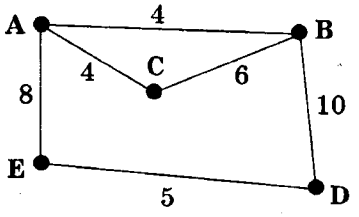
Нетрудно видеть, что полученная матрица смежности совпадает с заданной таблицей для варианта №4.

Ответ: вариант ответа №4.

Поскольку матрица смежности симметрична относительно ее главной диагонали, достаточно заполнять и сравнивать с заданной таблицей только одну ее часть (например, над главной диагональю).

Данная задача — одна из наиболее простых. Для её решения можно не строить матрицы смежности для каждого предложенного варианта графа, а просто сверять каждый вариант графа с заданной таблицей: проверять, имеется ли для каждого ребра графа в таблице на пересечении столбца и строки, соответствующих концевым вершинам ребра, значение, равное весу этого ребра. Вариант графа, для которого замечено первое же несоответствие заданной таблице, сразу отбрасывается (т.е. продолжение его анализа не требуется).

Задача 5. Дана схема дорог между пятью городами A, B, C, D и E с указанием протяжённости этих дорог. Передвигаться можно только по этим дорогам. Укажите два города, дальность поездки между которыми наибольшая. В ответе укажете кратчайшее расстояние между этими городами.



1) 14 2) 15 3) 16 4) 17

Решение

Для решения этой задачи можно построить таблицу, похожую по структуре на матрицу смежности, но содержащую на пересечениях строк и столбцов с названиями городов не веса ребер, а длины путей. При этом если таких путей несколько, то в таблицу записываем длину самого короткого из них (с минимальной суммой весов составляющих его рёбер):

	A	B	C	D	E
A		4	4	13 (AED)	8
B	4		6	10	12 (BAE)
C	4	6		16 (CBD)	12 (CAE)
D	12 (AED)	10	16 (CBD)		5
E	8	12 (BAE)	12 (CAE)	5	

Поскольку в таблицу заносились наименьшие длины соответствующих путей, для получения ответа достаточно найти в построенной таблице наибольшее значение. (Если требуется указать города, то нужно также посмотреть, на пересечении каких строк и столбцов оно находится, — записанные в этих строке и столбце названия городов и есть искомые.)

Ответ: 16 (расстояние между городами C и D по маршруту CBD).

Как и в предыдущих задачах, в силу симметричности получаемой таблицы относительно её главной диагонали, достаточно строить и анализировать только половину таблицы (например, над её главной диагональю).

Задачи для самостоятельного решения

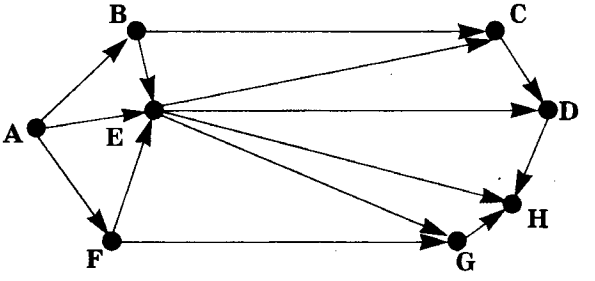
1. Между городами A, B, C, D, E, F проложены дороги, протяжённость которых приведена в таблице. (Отсутствие числа на пересечении соответствующих столбцов и строк указывает, что дороги между этими городами нет.) Передвигаться можно только по имеющимся дорогам.

	A	B	C	D	E	F
A				1	4	3
B			4			5
C		4		2	1	
D	1		2			2
E	4		1			
F	3	5		2		

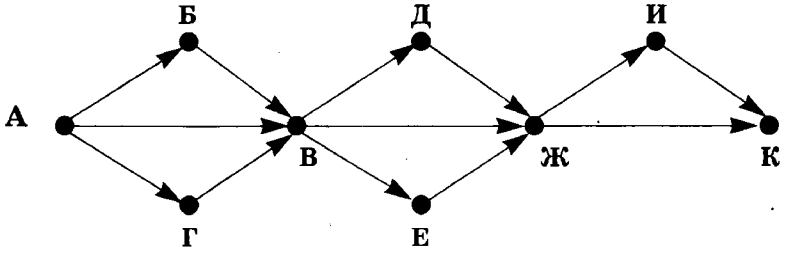
Определите длину кратчайшего пути между городами A и B.

1) 6 2) 7 3) 8 4) 9

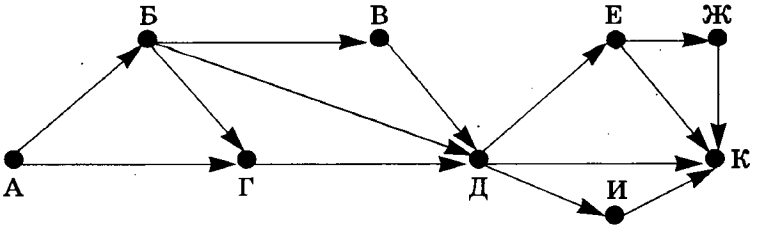
2. Дана схема дорог между городами A, B, C, D, E, F, G, H. По дорогам можно двигаться только в одном направлении, указанном стрелкой. Сколько возможно различных путей из города A в город H?



3. Дана схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, И, К. По дорогам можно двигаться только в одном направлении, указанном стрелкой. Сколько возможно различных путей из города А в город К?



4. Дана схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, И, К. По дорогам можно двигаться только в одном направлении, указанном стрелкой. Сколько возможно различных путей из города А в город К?



5. Числа, стоящие на пересечениях строк и столбцов таблиц, обозначают стоимость проезда между соответствующими соседними станциями. (Если число на пересечении какой-либо строки и столбца не указано, то соответствующие станции не являются соседними.)
Стоимость проезда по некоторому маршруту вычисляется как сумма стоимостей проезда между соответствующими соседними станциями.
Для какой из приведённых ниже таблиц выполняется условие: минимальная стоимость проезда по маршруту из E в B не более 5?

1)

	A	B	C	D	E
A		1	3		6
B	1			3	
C	3			4	
D		3	4		3
E	6			3	

3)

	A	B	C	D	E
A		2	4		6
B	2			4	
C	4			2	
D		4	2		
E	6				

2)

	A	B	C	D	E
A		3	4		7
B	3			4	
C	4				
D		4			1
E	7			1	

4)

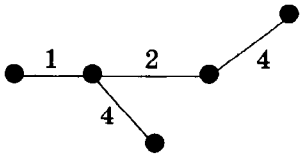
	A	B	C	D	E
A		4	2		7
B	4			3	
C	2			6	
D		3	6		3
E	6			3	

6. Есть четыре города: A, B, C и D.
Между городами A и C проложено три дороги.
Между городами C и B проложено две дороги.
Между городами A и B проложено две дороги.
Между городами C и D проложено две дороги.
Между городами B и D проложено четыре дороги.
По каждой дороге можно ехать в обе стороны. Сколько возможно способов проезда из A в D, посещая каждый город не более одного раза?

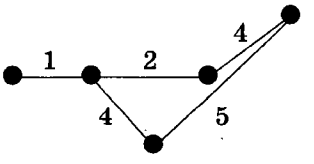
7. В таблице числа, записанные на пересечениях строк и столбцов, обозначают стоимость проезда между соответствующими соседними станциями. Если число не указано, то соответствующие станции не являются соседними.
Какая из приведённых ниже схем соответствует таблице?

	A	B	C	D	E
A		1	4	2	
B	1				
C	4				5
D	2				4
E			5	4	

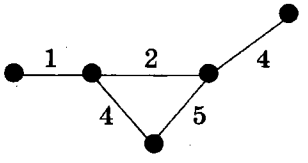
1)



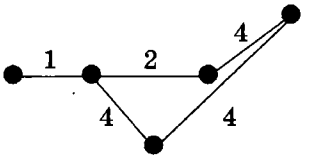
3)



2)

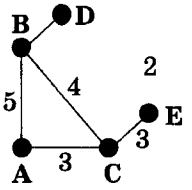


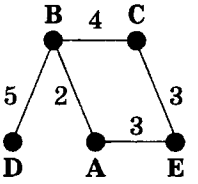
4)

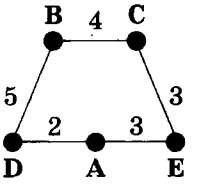


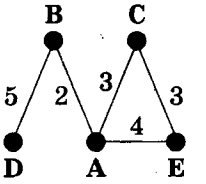
8. В таблице приведена стоимость перевозки пассажиров между соседними населенными пунктами. Укажите схему, соответствующую таблице.

	A	B	C	D	E
A		5	3		
B	5		4	2	
C	3	4			3
D		2			
E			3		

1) 

2) 

3) 

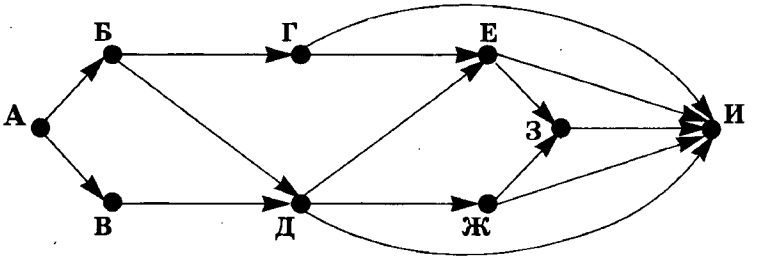
4) 

9. Между населёнными пунктами A, B, C, D, E, F построены дороги, протяжённость которых приведена в таблице. (Отсутствие числа в таблице означает, что прямой дороги между пунктами нет.)

	A	B	C	D	E	F
A		2	4		1	
B	2			5		3
C	4				1	
D		5				2
4	1		1			5
F		3		2	5	

Определите длину кратчайшего пути между пунктами A и F (при условии, что передвигаться можно только по построенным дорогам).

10. На рисунке — схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, З, И. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой.
Сколько существует различных путей из города А в город И?



Ответы для самопроверки

№ задания	Ответ	№ задания	Ответ
1	2	6	46
2	2	7	3
3	18	8	1
4	16	9	5
5	2	10	13

Системы счисления

A1, B8 Двоичная, восьмеричная, шестнадцатеричная системы счисления. Арифметика в указанных системах счисления

Конспект

Система счисления — знаковая система, позволяющая по определённым правилам записывать числа при помощи символов некоторого алфавита (цифр).

Позиционные системы счисления: количественные значения цифр зависят от их позиций (разрядов) в числе, что позволяет при помощи небольшого набора цифр записывать практически любые по величине числа.

Непозиционные системы счисления: значение числа получается путём суммирования (и вычитания) количественных значений цифр, не зависящих от их местоположения в числе. Пример: римская система счисления.

Обычно при записи числа значение основания системы счисления записывается в виде нижнего индекса после последней цифры числа.

Примеры наиболее часто используемых систем счисления:

Система счисления	Основание (p)	Алфавит системы счисления	Пример записи числа
Двоичная	2	0, 1	101101_2
Восьмеричная	8	0, 1, 2, 3, 4, 5, 6, 7	12345_8
Десятичная	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	1234_{10}
Шестнадцатеричная	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (=10), B (=11), C (=12), D (=13), E (=14), F (=15)	$F4D9_{16}$

Формы записи чисел в различных системах счисления

Свёрнутая («обычная») форма записи числа — привычная запись числа как последовательности цифр, стоящих на своих разрядах.

Развёрнутая форма записи числа — запись числа в виде суммы произведений его цифр на основание системы счисления в степени, равной значению разряда той или иной цифры числа (для целого числа нумерация разрядов ведётся с нуля справа налево; для дробного числа нумерация разрядов ведётся от десятичной запятой влево по возрастанию, а вправо — по убыванию, при этом разряду единиц присваивается нулевой номер).

Форма (схема) Горнера — преобразованная запись развёрнутой формы, при которой за счёт использования скобок удаётся избавиться от возведения основания счисления в степени. Схема Горнера предполагает рекуррентные вычисления.

Примеры:

а) для целых чисел:

Формы записи	Примеры чисел			
	двоичное	восьмеричное	десятичное	шестнадцатеричное
Свёрнутая	101101	12345	1234	F4D9
Развёрнутая	$1\cdot2^5+0\cdot2^4+1\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0$	$1\cdot8^4+2\cdot8^3+3\cdot8^2+4\cdot8^1+5\cdot8^0$	$1\cdot10^3+2\cdot10^2+3\cdot10^1+4\cdot10^0$	$F\cdot16^3+4\cdot16^2+D\cdot16^1+9\cdot16^0=(15)\cdot16^3+4\cdot16^2+(13)\cdot16^1+9\cdot16^0$
Схема Горнера	$((((1\cdot2+0)\cdot2+1)\cdot2+1)\times2+0)\cdot2+1$	$(((1\cdot8^4+2)\cdot8+3)\times8+4)\cdot8+5$	$((1\cdot10^3+2)\times10+3)\cdot10+4$	$((F\cdot16^3+4)\cdot16+D)\cdot16+9$

б) для дробных чисел:

Формы записи	Примеры чисел	
	двоичное	восьмеричное
Свёрнутая	1001,11	123,45
Развёрнутая	$1\cdot2^3+0\cdot2^2+0\cdot2^1+1\cdot2^0+1\cdot2^{-1}+1\cdot2^{-2}$	$1\cdot10^2+2\cdot10^1+3\cdot10^0+4\cdot10^{-1}+5\cdot10^{-2}$

Перевод чисел между системами счисления с кратными основаниями. Если основания исходной и конечной системы кратны друг другу, то перевод чисел между этими системами счисления можно выполнять по упрощённой схеме.

1. *Перевод двоичного числа в восьмеричную систему счисления* производится по триадам цифр:

Двоичная триада	000	001	010	011	100	101	110	111
Восьмеричное значение	0	1	2	3	4	5	6	7

2. *Перевод двоичного числа в шестнадцатеричную систему счисления* производится по тетрадам цифр:

Двоичная триада	0000	0001	0010	0011	0100	0101	0110	0111
Шестнадцатеричное значение	0	1	2	3	4	5	6	7
Двоичная триада	1000	1001	1010	1011	1100	1101	1110	1111
Шестнадцатеричное значение	8	9	A	B	C	D	E	F

Таблицы степеней

Существенную помощь при вычислениях, связанных с переводом чисел в десятичную систему счисления, могут оказать таблицы значений степеней оснований системы счисления. В качестве примера приведена такая таблица для основания системы счисления, равного 2 (рекомендуется выучить её наизусть). Аналогично можно составить такие таблицы и для других оснований систем счисления.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2^n	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
Значение	1	2	4	8	16	32	64	128	256	516	1024	2048	4096	8192	16386	32768	65536

Арифметика в недесятичных системах счисления (на примере двоичной арифметики)

Правила выполнения арифметических действий над двоичными числами задаются таблицами двоичных сложения, вычитания и умножения.

Таблица двоичного сложения	Таблица двоичного вычитания	Таблица двоичного умножения
$0+0=0$	$0-0=0$	$0\times0=0$
$0+1=1$	$1-0=1$	$0\times1=0$
$1+0=1$	$1-1=0$	$1\times0=0$
$1+1=10$	$10-1=1$	$1\times1=1$

При сложении двоичных чисел в каждом разряде производится сложение разрядов.

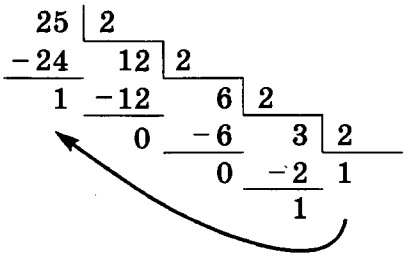
Разбор типовых задач

Задача 1*. Как представлено число 25_{10} в двоичной системе счисления?

1) 1001_2 2) 11001_2 3) 10011_2 4) 11010_2

Решение

Достаточно выполнить перевод заданного числа в двоичную систему счисления путём последовательных делений «в столбик» на 2:



Ответ: $25_{10}=11001_2$ (вариант ответа №2).

Задача 2*. Чему равно количество значащих нулей в двоичной записи десятичного числа 126.

1) 1 2) 2 3) 3 4) 0

Решение

1. Число 126_{10} переводится в двоичную систему счисления: $126_{10}=1111110_2$.
2. Подсчитывается количество значащих нулей в двоичной записи числа: 1 нуль.

Ответ: 1 (вариант ответа №1).

Задача 3*. Сколько единиц в двоичной записи числа 195?

1) 5 2) 2 3) 3 4) 4

Решение

1. Число 195_{10} переводится в двоичную систему счисления: $195_{10}=11000011_2$.
2. Подсчитывается количество единиц в двоичной записи числа: 4 единицы.

Ответ: 4 (вариант ответа №4).

Задача 4*. Сколько единиц в двоичной записи десятичного числа 194,5?

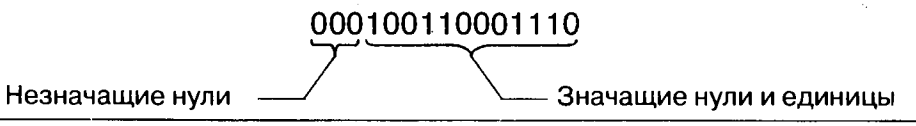
1) 5 2) 2 3) 3 4) 4

Решение

1. Число $194{,}5_{10}$ переводится в двоичную систему счисления:
   - — целая часть числа: $194_{10} = 11000010_2$;
   - — дробная часть числа: $0{,}5_{10} = 0{,}1_2$ ($0{,}5 = {}^1/_2 = 2^{-1}$);
   - — $194{,}5_{10} = 11000010{,}1_2$.
2. Подсчитывается количество единиц в двоичной записи числа: 4 единицы.

Ответ: 4 (вариант ответа №4).

Незначащими в записи числа являются только нули, стоящие слева от числа (которые не влияют на числовое значение и могут быть отброшены). Все остальные нули и единицы в записи числа являются *значащими*!



Задача 5*. Дано $A = A7_{16}$, $B = 251_8$. Какое из чисел C, записанных в двоичной системе, отвечает условию $A < C < B$?

1) 10101100_2 2) 10101010_2 3) 10101011_2 4) 10101000_2

Решение

1. Используя правила перевода восьмеричных и шестнадцатеричных чисел в двоичную систему счисления по триадам и тетрадам, переводятся все заданные числа в одну и ту же систему счисления — двоичную (в которой представлены варианты ответов):
   - — $A = A7_{16} \Rightarrow (A_{16} = 1010_2)\,(7_{16} = 0111_2) \Rightarrow 10100111_2$;
   - — $B = 251_8$ $(2_8 = 010_2)\,(5_8 = 101_2)\,(1_8 = 001_2) \Rightarrow 010101001_2 \Rightarrow 10101001_2$.
2. Записываются полученные двоичные значения в заданный «шаблон» неравенства:

$$\underline{1010}0111_2 < \quad < \underline{1010}1001_2.$$

3. Внимание: первые четыре бита в этих числах одинаковы (выделено подчёркиванием), а последние четыре бита соответствуют в левом (меньшем) числе 0111, а в правом (большем) — 1001. Следовательно, среднее число неравенства (C) должно:
   - — начинаться с тех же четырёх битов 1010;
   - — заканчиваться четвёркой битов 1000 (единственной возможной между 0111 и 1001).

Следовательно, искомое число равно 10101000_2.

Ответ: $C = 10101000_2$ (вариант ответа №4).

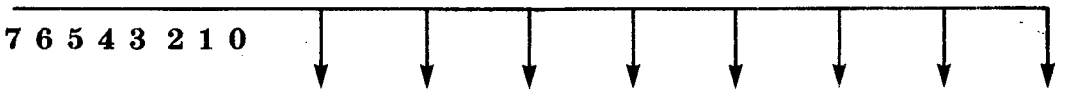
При преобразовании чисел по триадам и тетрадам нужно не забывать дополнять триады (тетрады) незначащими нулями слева до трёх (четырёх) знаков, а также отбрасывать незначащие нули после завершения преобразования.

Задача 6*. Дано: $a = 11100110_2$, $b = 271_8$. Какое из чисел C, записанных в шестнадцатеричной системе счисления, удовлетворяет неравенству $a > C > b$?

1) AA_{16} 2) $B8_{16}$ 3) $D6_{16}$ 4) $F0_{16}$

Решение

Для упрощения расчётов и исключения возможных ошибок при выполнении арифметических операций в недесятичных системах счисления рекомендуется вначале перевести все числа-операнды в десятичную систему счисления, выполнить расчёты в ней, а затем выполнить перевод результата в искомую систему счисления.



$$a = 11100110_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 =$$
$$= 128 + 64 + 32 + 4 + 2 = 230_{10}$$

Необходимо помнить: **любое число в нулевой степени равно 1.**

$2^0=1$

Для быстрого подсчета значений степени 2 надо знать **наизусть!!!**

2^1	2^2	2^3	2^4	2^5	2^6	2^7
2	4	8	16	32	64	128

$$b = 271_8 = 2 \cdot 8^2 + 7 \cdot 8^1 + 1 \cdot 8^0 = 2 \cdot 64 + 56 + 1 = 185_{10}$$

1) $AA_{16} = 10 \cdot 16^1 + 10 \cdot 16^0 = 170$
2) $B8_{16} = 11 \cdot 16^1 + 8 \cdot 16^0 = 176 + 8 = 184$
3) $D6_{16} = 13 \cdot 16^1 + 6 \cdot 16^0 = 208 + 6 = 214$
4) $F0_{16} = 15 \cdot 16^1 + 0 \cdot 16^0 = 240 + 0 = 240$

Надо найти значение между числами 186 и 230.

170 184 (214) 240

Ответ: $D6_{16}$ (вариант ответа №3).

Задача 7*. Вычислите значение суммы $10_2 + 10_8 + 10_{16}$ в двоичной системе счисления.

1) 10100010 2) 11110 3) 11010 4) 10100

Решение

В подобных задачах рекомендуется сразу перевести все числа в десятичную систему счисления, выполнить вычисления в ней, а затем выполнить перевод результата из десятичной системы счисления в требуемую:

— $10_2 = 2$;
— $10_8 = 8$;
— $10_{16} = 16$;
— $2 + 8 + 16 = 26$;
— $26_{10} = 11010_2$.

Ответ: 11010_2 (вариант ответа №3).

Задача 8*. Вычислите сумму чисел X и Y, если, $Y = 135_8$. Результат представьте в двоичном виде.

1) 11010100 2) 10100100 3) 10010011 4) 10010100

Решение

— $X = 110111_2 = 55_{10}$;
— $Y = 135_8 = 93_{10}$;
— $X + Y = 55 + 93 = 148$;
— $148_{10} = 10010100_2$.

Ответ: 10010100_2 (вариант ответа №4).

Задача 9*. В системе счисления с некоторым основанием число 12 записывается в виде 110. Укажите это основание.

Решение

Решение подобных задач основано на развёрнутой форме записи числа в системе счисления с основанием n при его переводе в десятичную систему:

$d_{10} = a_m \cdot n_m + a_{m-1} + \ldots + a_2 \cdot n_2 + a_1 \cdot n_1 + a_0$.

Неизвестное основание системы счисления обозначается буквой n и записывается уравнение на основе имеющихся у нас данных:

$12 = 1 \cdot n^2 + 1 \cdot n + 0$.

В результате получается достаточно простое квадратное уравнение:

$n^2 + n - 12 = 0$,

решить которое можно традиционным способом — через вычисление дискриминанта:

$D = b^2 - 4ac = 1 + 4 \cdot 12 = 49$;

$$n_1 = \frac{-b - \sqrt{d}}{2a} = \frac{-1-7}{2} = -4,\ n_2 = \frac{-b + \sqrt{d}}{2a} = \frac{-1+7}{2} = 3.$$

Очевидно, что основание системы счисления может быть только натуральным числом, поэтому первый (отрицательный) корень уравнения не подходит. Следовательно, ответом является число 3 — троичная система счисления.

Ответ: 3.

Задача 10*. В системе счисления с некоторым основанием десятичное число 49 записывается в виде 100. Укажите это основание.

Решение

Аналогично предыдущей задаче:

$49 = 1 \cdot n^2 + 0 \cdot n + 0 \Rightarrow n^2 - 49 = 0 \Rightarrow n^2 = 49 \Rightarrow n_1 = -7,\ n_2 = 7$.

Первый (отрицательный) корень уравнения не подходит. Следовательно, ответом является число 7 — семиричная система счисления.

Ответ: 7.

Задача 11*. В системе счисления с некоторым основанием десятичное число 18 записывается в виде 30. Укажите это основание.

Решение

В данном случае не требуется даже решение квадратного уравнения.

$18 = 3 \cdot n + 0$; $\Rightarrow 3 \cdot n = 18$; $\Rightarrow n = 6$ (шестеричная система счисления).

Ответ: 6.

Задача 12*. Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись числа 22 оканчивается на 4.

Решение

Решение таких задач по сути аналогично ранее рассмотренному и тоже основано на представлении записи числа в системе счисления с основанием n при его переводе в десятичную систему. Но, кроме того, потребуется вспомнить ещё одну форму записи — формулу Горнера:

$d_{10} = a_m \cdot n^m + a_{m-1} \cdot n^{m-1} + \ldots + a_2 \cdot n^2 + a_1 \cdot n^1 + a_0 =$

$= (a_m \cdot n^{m-1} + a_{m-1} \cdot n^{m-2} + \ldots + a_2 \cdot n^1 + a_1) \cdot n + a_0 =$

$= ((a_m \cdot n^{m-2} + a_{m-1} \cdot n^{m-3} + \ldots + a_2) \cdot n + a_1) \cdot n + a_0 =$

$= \ldots = (((\ldots(a_m \cdot n + a_{m-1}) \cdot n + \ldots + a_2) \cdot n + a_1) \cdot n + a_0$.

Точнее, потребуется только первый шаг преобразования «канонической» записи в формулу Горнера:

$d_{10} = a_m \cdot n^m + a_{m-1} \cdot n^{m-1} + \ldots + a_2 \cdot n^2 + a_1 \cdot n^1 + a_0 =$

$= (a_m \cdot n^{m-1} + a_{m-1} \cdot n^{m-2} + \ldots + a_2 \cdot n^1 + a_1) \cdot n + a_0.$

В этой записи — исходное десятичное число (d); цифра, которой должна заканчиваться запись числа в искомой системе счисления (a_0); основание системы счисления (n) и некоторый сомножитель n, который обозначается, например, как x:

$d_{10} = x \cdot n + a_0.$

Отсюда выражается искомое значение n:

$n = \frac{d_{10} - a_0}{x}.$

При этом неизвестная величина x рассматривается как некоторый параметр, на который наложено условие: x является натуральным числом (т. е. целым и положительным). Другое очевидное условие — получаемое значение n тоже должно быть натуральным числом, причём его значение должно быть больше, чем единственная указанная в условии задачи цифра записи числа в этой системе счисления.

Остаётся последовательно перебирать разные значения x, начиная с 1 и далее по возрастанию, и проверять получаемый результат (n) на соответствие указанным выше условиям. Впрочем, можно догадаться, что значения x должны быть кратны получаемой разности $d_{10} - a_0$.

В задаче дано исходное десятичное число 22 и указано, что его запись должна заканчиваться цифрой 4. Следовательно, n должно быть больше 4. Кроме того, можно записать:

$n = \frac{d_{10} - a_0}{x} = \frac{22 - 4}{x} = \frac{18}{x}$, т. е. требуется найти значения x, кратные 18 (те, на которые число 18 делится нацело). Очевидно, это значения 1, 2, 3, 6, 9 и 18. Но, как уже было сказано выше, значение n должно быть больше 4. Поэтому из всей найденной подборки подходят только значения 6, 9 и 18.

Ответ: 6, 9, 18.

Задача 13. В каких системах счисления запись числа 31 оканчивается на 11? (В качестве ответа нужно записать через запятую в порядке возрастания основания этих систем счисления.)

Решение

Решение этой задачи производится аналогично предыдущей, но формулу Горнера требуется расписать ещё на один шаг «глубже»:

$d_{10} = a_m \cdot n^m + a_{m-1} \cdot n^{m-1} + \ldots + a_2 \cdot n^2 + a_1 \cdot n^1 + a_0 =$

$= (a_m \cdot n^{m-1} + a_{m-1} \cdot n^{m-2} + \ldots + a_2 \cdot n^1 + a_1) \cdot n + a_0 =$

$= ((a_m \cdot n^{m-2} + a_{m-1} \cdot n^{m-3} + \ldots + a_2) \cdot n + a_1) \cdot n + a_0.$

Здесь имеется исходное число (d), две указанные цифры окончания записи (a_0 и a_1) и искомое основание системы счисления (n), а сомножитель ($a_m \cdot n^{m-2} + a_{m-1} \cdot n^{m-3} + \ldots + a_2$), как и ранее, обозначается буквой x и считается за некий параметр.

Для задачи получается запись:

$31 = (x \cdot n) \cdot n + 1$, откуда

$(x \cdot n + 1) \cdot n = 30.$

Следовательно, требуется найти значения n, кратные числу 30. Это: 1, 2, 3, 5, 6, 10, 15 и 30.

Остаётся проверить каждое из этих значений на соответствие условиям: x — целое неотрицательное[1] число; n — натуральное число, большее 1. При этом вспомогательная формула имеет вид:

$$x = \frac{30 - n}{n^2}.$$

n	1	2	3	5	6	10	15	30
x	29	7	3	1	2/3	1/5	1/15	0
Да/нет	нет	да	да	да	нет	нет	нет	да

Ответ: 2, 3, 5 и 30.

Задача 14*. Укажите через запятую в порядке возрастания все десятичные числа, не превосходящие 25, запись которых в системе счисления с основанием четыре оканчивается на 11.

Решение

В отличие от предыдущей задачи, где требовалось найти основание системы счисления, решение данной задачи гораздо проще.

1. В указанную систему счисления переводится заданное предельное десятичное число: $25_{10} = 121_4$.

2. Выписываются все числа в четверичной (!) системе счисления, которые заканчиваются на 11 и меньше, чем число 121_4: очевидно, это числа 11_4 и 111_4 (следующее подобное число 211_4 не подходит, так как оно превышает 121_4).

Эти два числа — и есть ответ к задаче. Осталось перевести их в десятичную форму: $11_4 = 4 + 1 = 5$; $111_4 = 4^2 + 4 + 1 = 21$.

Ответ: 5, 21.

Задача 15. Каково наименьшее основание системы счисления, запись числа 30 в которой состоит ровно из трёх цифр?

Решение

Какая запись числа в системе счисления с некоторым основанием n является трёхзначной? Очевидно, любая в интервале $[100_n, 1000_n]$ (где первое число входит в искомый интервал, а второе — не входит). При этом запись исходного десятичного числа (30) должна для всех искомых оснований систем счисления попадать в этот интервал. Следовательно, можно записать такое двойное неравенство:

$100_n \leqslant$ (*запись исходного числа в системе счисления с основанием* n) $< 1000_n$.

Преобразуем это неравенство в десятичный формат, используя развернутую форму записи числа в системе счисления с основанием n:

$n^2 \leqslant$ (*исходное число*) $< n^3$.

В нашем случае получается неравенство:

$n^2 \leqslant 30 < n^3$.

---

[1] В данном случае нулевое значение допустимо. Тогда получается запись исходного числа в искомой системе счисления в виде 11, — но если всё число состоит их цифр 11, то вполне можно сказать, что это число «оканчивается» на эти цифры!

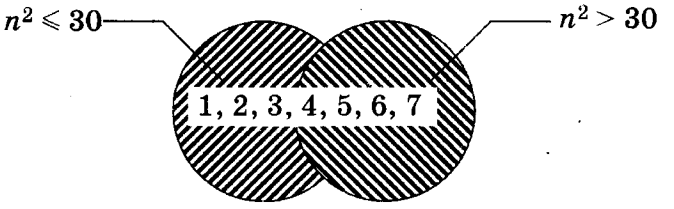
Двойное неравенство для его решения относительно n проще всего представить в виде системы двух отдельных неравенств, каждое из которых сравнительно нетрудно решить простым перебором возможных значений n (причём эти значения должны быть натуральными числами!):

$$\begin{cases} n^2 \leqslant 30 \\ n^3 > 30 \end{cases}$$

Первое неравенство: ищутся все натуральные значения n, квадраты которых не превышают 30. Очевидно, это числа 1, 2, 3, 4 и 5.

Второе неравенство: ищутся все натуральные значения n, кубы которых превышают 30. Очевидно, это числа 4, 5, 6 и т. д. по возрастанию.

Чтобы получить решение системы этих неравенств, нужно найти пересечение найденных множеств значений n: это будут числа 4 и 5.



Поскольку в задаче требуется найти наименьшее основание системы счисления, ответом будет число 4.

Можно проверить правильность решения, переведя исходное число 30 в системы счисления с основаниями 3, 4, 5 и 6 и убедившись, что для оснований 4 и 5 получается трёхзначная запись числа, для основания 6 — только двузначная, а для основания 3 — уже четырёхзначная.

Ответ: 4.

Задача 16. Каково наименьшее основание системы счисления, запись числа 50 в которой состоит ровно из двух цифр?

Решение

Производится аналогично предыдущей задаче, но поскольку требуется получить двузначную запись числа, то получаемое неравенство будет иметь вид:

$n \leqslant$ (*исходное число*) $< n^2$.

То есть получается неравенство:

$n \leqslant 50 < n^2$,

или система неравенств:

$$\begin{cases} n \leqslant 50 \\ n^2 > 50 \end{cases}$$

Решение первого неравенства: $n = 1 \ldots 50$. Решение второго неравенства: $n = 8, 9, 10 \ldots$ Пересечение этих двух множеств (решение системы неравенств): $n = 8 \ldots 50$. Тогда наименьшее основание системы счисления, удовлетворяющее условию задачи, равно 8.

Ответ: 8.

Задача 17. Имеется десятичное число 1104. В некоторой системе счисления это число записывается тремя единицами и тремя нулями. Найти основание этой системы счисления.

Решение (вариант 1)

Основная сложность этой задачи в том, что известны только количества нулей и единиц в искомой записи числа, но не известны их позиции в этой записи. Единственное, что можно

сказать точно, — то, что искомая запись состоит из 6 цифр, из которых три — единицы, а три — нули. Поэтому начать решение нужно с полной шестизначной записи числа 1104_{10} в системе с искомым основанием x:

$$1104_{10} = a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0,$$

где a_5, a_4, a_3, a_2, a_1, a_0 — цифры записи числа в системе счисления с искомым основанием x.

Какие из этих цифр равны 1, а какие — 0? Очевидно, первая цифра a_5 **точно** равна 1 (иначе запись числа не была бы шестизначной). Расположение же двух других единиц неизвестно, поэтому требуется проверять все возможные варианты:

a_5	a_4	a_3	a_2	a_1	a_0
1	1	1	0	0	0
1	1	0	1	0	0
1	1	0	0	1	0
1	1	0	0	0	1
1	0	1	1	0	0
1	0	1	0	1	0
1	0	1	0	0	1
1	0	0	1	1	0
1	0	0	1	0	1
1	0	0	0	1	1

Положение первой единицы определено однозначно ($a_5 = 1$). Остальные возможные варианты расположения двух единиц определяются так. Сначала предполагается, что обе единицы — это a_4 и a_3. Затем «закрепляется» положение первой единицы (a_4) и перебираются все возможные положения второй единицы. Затем меняем положение первой единицы на следующее по порядку a и перебираются все возможные положения второй единицы и т.д.

Реально возможно найти решение для каких-то первых же вариантов, так что перебирать все их не потребуется.

1. Пусть $(a_5, a_4, a_3, a_2, a_1, a_0) = (111000)$. Тогда исходная запись

$1104_{10} = a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0,$

вырождается в запись: $1104_{10} = x^5 + x^4 + x^3$. Чему может быть равен x?

Перебираются возможные значения x, начиная с минимально возможного (2):

- $x = 2$: $2^5 + 2^4 + 2^3 = 32 + 16 + 8 = 56$ (< 1104); равенства нет, значит $x = 2$ — не подходит;
- $x = 3$: $3^5 + 3^4 + 3^3 = 243 + 81 + 27 = 351$ (< 1104); $x = 3$ — также не подходит;
- $x = 4$: $4^5 + 4^4 + 4^3 = 1024 + 256 + 64 = 1344$ (> 1104); $x = 4$ — также не подходит, при этом раз уже получено значение для $x = 4$, большее требуемого, очевидно, что для бóльших значений x будут тоже получаться значения больше требуемого 1104. Значит, данный вариант расположения нулей и единиц в записи числа непригоден при **любом** значении x (или, что то же самое, уравнение $1104_{10} = x^5 + x^4 + x^3$ не имеет решения в натуральных числах больше 1).

2. Пусть $(a_5, a_4, a_3, a_2, a_1, a_0) = (110100)$. Тогда исходная запись

$1104_{10} = a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0,$

вырождается в запись: $1104_{10} = x^5 + x^4 + x^2$. Чему может быть равен x?

Вновь перебираются возможные значения x, начиная с минимально возможного (2):

- $x = 2$: $2^5 + 2^4 + 2^2 = 32 + 16 + 4 = 52$ (< 1104); равенства нет, значит $x = 2$ — не подходит;

• $x = 3$: $3^5 + 3^4 + 3^2 = 243 + 81 + 9 = 333$ (< 1104); $x = 3$ — также не подходит;
• $x = 4$: $4^5 + 4^4 + 4^2 = 1024 + 256 + 8 = 1288$ (> 1104); $x = 4$ — также не подходит, и очевидно, что для бóльших значений x будут тоже получаться значения больше требуемого 1104. Значит, данный вариант непригоден при любом значении x.

3. Пусть $(a_5, a_4, a_3, a_2, a_1, a_0) = (110010)$. Тогда исходная запись

$$1104_{10} = a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0,$$

вырождается в запись: $1104_{10} = x^5 + x^4 + x$.

Перебираются возможные значения x, начиная с минимально возможного (2):
• $x = 2$: $2^5 + 2^4 + 2 = 32 + 16 + 2 = 50$ (< 1104); $x = 2$ — не подходит;
• $x = 3$: $3^5 + 3^4 + 3 = 243 + 81 + 3 = 327$ (< 1104); $x = 3$ — не подходит;
• $x = 4$: $4^5 + 4^4 + 4 = 1024 + 256 + 4 = 1284$ (> 1104); $x = 4$ — не подходит, и для бóльших значений x будут тоже получаться значения больше требуемого 1104. Значит, данный вариант непригоден при любом значении x.

4. Пусть $(a_5, a_4, a_3, a_2, a_1, a_0) = (110001)$. Тогда исходная запись

$1104_{10} = a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$,

вырождается в запись: $1104_{10} = x^5 + x^4 + 1$.

Перебираются возможные значения x, начиная с минимально возможного (2):
• $x = 2$: $2^5 + 2^4 + 1 = 32 + 16 + 1 = 49$ (< 1104); $x = 2$ — не подходит;
• $x = 3$: $3^5 + 3^4 + 1 = 243 + 81 + 1 = 325$ (< 1104); $x = 3$ — не подходит;
• $x = 4$: $4^5 + 4^4 + 1 = 1024 + 256 + 1 = 1281$ (> 1104); $x = 4$ — не подходит, и для бóльших значений x будут тоже получаться значения больше требуемого 1104. Значит, данный вариант непригоден при любом значении x.

5. Пусть $(a_5, a_4, a_3, a_2, a_1, a_0) = (101100)$. Тогда исходная запись

$1104_{10} = a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$,

вырождается в запись: $1104_{10} = x^5 + x^3 + x^2$.

Перебираются возможные значения x, начиная с минимально возможного (2):
• $x = 2$: $2^5 + 2^3 + 2^2 = 32 + 8 + 4 = 44$ (< 1104); $x = 2$ — не подходит;
• $x = 3$: $3^5 + 3^3 + 3^2 = 243 + 27 + 9 = 279$ (< 1104); $x = 3$ — не подходит;
• $x = 4$: $4^5 + 4^3 + 4^2 = 1024 + 64 + 16 = 1104$.

Итак, для данного варианта расположения трёх единиц и трёх нулей в записи числа и для основания системы счисления, равного 4, мы получили требуемое десятичное значение 1104. Следовательно, искомое основание системы счисления равно 4.

Решение (вариант 2)

Можно решить эту задачу и «в лоб», без каких-либо логических рассуждений. Достаточно просто пытаться преобразовывать исходное десятичное число в двоичную, троичную и т.д. систему счисления, в полученном результате подсчитывать количества единиц и нулей и проверять эти количества на соответствие заданному в задаче условию (три единицы, три нуля и больше никаких других цифр), пока для какого-то основания системы не будет найдено такое соответствие. Единственный недостаток такого способа решения — деление (для пересчёта из десятичной системы счисления в искомую) выполняется сложнее и чревато большим количеством ошибок, чем умножение (при вычислении десятичного числа по записи в некоторой системе счисления по первому способу).

Итак:
- основание системы счисления 2: $1104_{10} = 10001010000_2$ — имеются три единицы, но количество нулей слишком велико;
- основание системы счисления 3: $1104_{10} = 1111220_3$ — имеются четыре единицы, только один нуль и «лишние» двойки;
- основание системы счисления 4: $1104_{10} = 101100_4$ — имеются ровно три единицы и ровно три нуля при отсутствии других цифр, следовательно, этот вариант подходит.

Ответ: 4.

Задачи для самостоятельного решения

1. Сколько значащих нулей в двоичной записи числа 2081?
   1) 8 2) 9 3) 10 4) 11
2. Сколько единиц в двоичной записи десятичного числа 317?
   1) 6 2)2 3)3 4) 4
3. В скольких различных системах счисления десятичное число 40 оканчивается нулём?
4. Какой будет запись десятичного числа 48 в шестеричной системе счисления?
5. Определите основание системы счисления, в которой десятичное число 77 записывается как 140.
6. Определите основание системы счисления, в которой десятичное число 80 записывается как 212.
7. Определите основание системы счисления, в которой десятичное число 100 записывается как 55.
8. Определите основание системы счисления, в которой десятичное число 70 записывается как 77.
9. Дано $A = 9D_{16}$, $B = 237_8$. Какое из чисел C, записанных в двоичной системе, отвечает условию $A < C < B$?
   1) 10011010_2 2) 10011110_2 3) 10011111_2 4) 11011110_2
10. Вычислите сумму двоичных чисел x и y, если $x=1010101_2$ $y=1010011_2$
    1) 10100010_2 2) 10101000_2 3) 10100100_2 4) 10111000_2
11. Вычислите сумму чисел x и y, при $x = 1D_{16}$, $y = 72_8$. Результат представьте в двоичной системе счисления.
    1) 10001111_2 2) 1100101_2 3) 101011_2 4) 1010111_2
12. Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись числа 23 оканчивается на 2.

Ответы для самопроверки

№ задания	Ответ
1	2
2	1
3	2, 4, 5, 8, 10, 20, 40
4	120
5	7
6	6
7	19
8	9
9	2
10	2
11	4
12	3, 7, 21

Системы счисления

B4 Задачи на кодирование, решаемые с применением недесятичных систем счисления

Конспект

Система счисления — знаковая система, позволяющая по определённым правилам записывать числа при помощи символов некоторого алфавита.

Как правило, символами алфавита системы счисления являются десятичные цифры. Однако это — лишь результат общепринятой договоренности, формально такими символами могут быть любые знаки, в том числе произвольные буквы (пример — использование латинских букв от A до F в качестве цифр шестнадцатеричной системы счисления.

Примеры наиболее часто используемых систем счисления:

Система счисления	Основание (p)	Алфавит системы счисления	Пример записи числа
Двоичная	2	0, 1	101101_2
Восьмеричная	8	0, 1, 2, 3, 4, 5, 6, 7	12345_8
Десятичная	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	1234_{10}
Шестнадцатеричная	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (=10), B (=11), C (=12), D (=13), E (=14), F (=15)	$F4D9_{16}$

Во многих задачах, в условиях которых не говорится о необходимости преобразования чисел в другую систему счисления или о выполнении арифметических операций в различных системах счисления, можно найти аналогию с той или иной позиционной системой счисления и этим облегчить решение задачи.

Разбор типовых задач

Задача 1. Все 5-буквенные слова, составленные из букв Т, О, Н, записаны в алфавитном порядке. Вот начало списка:

1. ТТТТТ
2. ТТТТО
3. ТТТТН
4. ТТТОТ

.....

Сколько букв Т встречается в слове, стоящем на 101-м месте от начала списка?

Решение

Эту задачу можно решать «в лоб», прослеживая (либо вычисляя по формулам комбинаторики) и записывая количества возможных комбинаций получаемых слов сначала с одной буквой О или Н, затем с двумя буквами ОО, НН, ОН или НО, затем с тремя такими буквами и т.д., и на каком-то шаге (в данном случае — на трёх добавленных буквах О и Н) заметив, что получаемое количество таких комбинаций превышает поставленную в условии задачи границу — 101-ю позицию в списке слов.

Однако, существует гораздо более простое и менее трудоёмкое решение. Буква Т заменяется на цифру 0, буква О — на цифру 1, а буква Н — на цифру 2. Такая замена правомерна, поскольку не меняет сути задачи, а сводится только к изменению используемых «кодов». Тогда условие задачи, по сути, преобразуется в следующий вид:

Все 5-значные числа, составленные из цифр 0, 1, 2, записаны в порядке возрастания. Вот начало списка:

1. 00000
2. 00001
3. 00002
4. 00010
.....

Сколько нулей встречается в числе, стоящем на 101-м месте от начала списка?

Такую задачу решить гораздо проще. Сразу можно заметить, что речь идёт о троичной системе счисления и о последовательности записанных в этой системе чисел, на каждом шаге увеличивающихся на единицу. При записи в десятичной системе счисления эта последовательность чисел эквивалентна ряду целых чисел 0, 1, 2, ...

Чтобы определить число, стоящее в списке на 101-й позиции, нужно учесть «рассогласование» между значениями чисел и их порядковыми номерами в списке:

- последовательность чисел начинается с нуля;
- нумерация чисел в списке начинается с единицы.

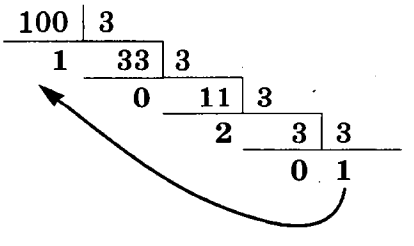
Тогда для определения числа, стоящего в 101-й позиции, нужно составить следующее соответствие:

Порядковый номер	Значение числа
1	0
101	(101 — 1)

Очевидно, что на 101-м месте в списке будет стоять десятичное число 100 (так как значение числа на 1 меньше, чем порядковый номер).

Общее правило: порядковый номер искомого числа надо увеличить на значение разности между значением первого числа в списке и порядковым номером первого числа в списке.

Какова будет пятиразрядная запись этого числа в троичной системе счисления? Это преобразование выполняется традиционным способом — путём последовательного деления «в столбик»:



Получается, что на 101-м месте в списке находится пятиразрядное троичное число **10201**.

Сколько в записи этого числа нулей? Два. Это и есть ответ, поскольку запись числа как слова, составленного из букв Т, О, Н (при возврате к исходному условию задачи), будет такой: **ОТНТО**.

Ответ: 2 буквы Т.

> Этот способ решения позволяет определить не только количество требуемых букв в искомом слове, но и само слово, расположенное на указанном месте в списке.

Задача 2. Все 5-буквенные слова, составленные из букв М, И, Р, записаны в алфавитном порядке. Вот начало списка:

1. ИИИМИ
2. ИИИММ
3. ИИИМР
4. ИИИРИ
5. ИИИРМ

.....

Какое слово стоит на 123-м месте в списке?

Решение

Поскольку слова начинаются с букв И, а в последовательности слов после ИИИММ стоит слово ИИИМР, то сопоставление букв цифрам будет следующим: И—0, М—1, Р—2. В результате исходная задача сводится к следующей:

Все 5-значные числа, составленные из цифр 0, 1, 2, записаны в порядке возрастания. Вот начало списка:

1. 00010
2. 00011
3. 00012
4. 00020
5. 00021

.....

Какое число стоит на 123-м месте в списке?

Чтобы определить число, стоящее в списке на 123-й позиции, нужно учесть «рассогласование» между значениями чисел и их порядковыми номерами в списке:

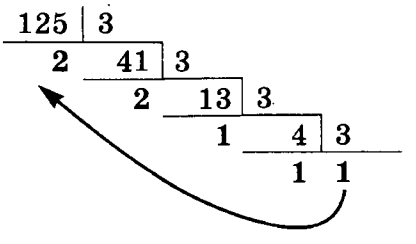
- последовательность чисел начинается с значения $00010_3 = 1 \cdot 3^1 = 3_{10}$;
- нумерация чисел в списке начинается с единицы.

Тогда для определения числа, стоящего в 123-й позиции, нужно составить следующее соответствие:

Порядковый номер	Значение числа
1	3
123	(123 + 2)

Очевидно, что на 123-м месте в списке стоит десятичное число 125 (так как значение числа на 2 больше, чем порядковый номер).

Определяется пятиразрядная запись этого числа в троичной системе счисления.



Получается, что на 123-м месте в списке находится пятиразрядное троичное число **11122**.

Это число преобразуется к записи слова, кодируемого буквами М, И, Р по правилу соответствия: И—0, М—1, Р—2. Получается слово: **МММРР**.

Ответ: МММРР.

Задача 3. Все 5-буквенные слова, составленные из букв Т, О, Н, записаны в алфавитном порядке. Вот начало списка:

1. ТТТТТ
2. ТТТТО
3. ТТТТН
4. ТТТОТ

.....

Укажите номер в списке слова, которое первым начинается с буквы Н.

Решение

Эта задача по смыслу противоположна предыдущим, но суть её решения остаётся та же.

Поскольку слова начинаются с букв Т, а в последовательности слов после ТТТТТ стоят слова ТТТТО и ТТТТН, то сопоставление букв цифрам будет следующим: Т—0, О—1, Н—2. В результате исходная задача сводится к следующей:

Все 5-значные числа, составленные из цифр 0, 1, 2, записаны в порядке возрастания. Вот начало списка:

1. 00000
2. 00001
3. 00002
4. 00010

.....

Укажите номер числа в списке, которое первым начинается с цифры 2.

Очевидно, что речь идёт о числах в троичной системе счисления, тогда искомым первым числом, начинающимся с цифры 2, является троичное число 20000.

Троичное число 20000 преобразуется в десятичную форму. В общем случае это делается традиционным способом поразрядного умножения на соответствующие степени основания системы счисления 3 и суммирования, а в данной конкретной задаче, поскольку все разряды троичного числа, кроме самого старшего (4-го) равны нулю. Это число в десятичном виде равно $2 \cdot 3^4 = 162$.

Чтобы определить номер позиции в списке для числа 162, следует учесть «рассогласование» между значениями чисел и их порядковыми номерами в списке:

- последовательность чисел начинается с нуля;
- нумерация чисел в списке начинается с единицы.

Тогда для определения номера позиции числа 162 составляется следующее соответствие:

Порядковый номер	Значение числа
1	0
(162 + 1)	162

Очевидно, что десятичное число 162 стоит в списке на 163-м месте (так как значение числа на 1 меньше, чем порядковый номер).

Общее правило: значение искомого числа надо уменьшить на значение разности между значением первого числа в списке и порядковым номером первого числа в списке.

Следовательно, искомое число 20000_3, а значит, и искомое слово НТТТТ (первое, начинающееся с буквы Н), будет стоять в списке в 163-й позиции.

Ответ: искомое слово расположено в позиции №163.

Задача 4*. Записано 7 строк, каждая имеет свой номер — от «0» — до «6»-й.

В начальный момент в строке записана цифра 0 (ноль). На каждом из последующих 6 шагов выполняется следующая операция: в очередную строку записывается удвоенная предыдущая строка, а в конец строки приписывается очередная цифра (на i-м шаге приписывается цифра i).

Для удобства в скобках пишется номер строки (начиная с 0).

Ниже показаны первые строки, сформированные по описанному правилу:

(0)0

(1)001

(2)0010012

(3)001001200100123

Какая цифра стоит в последней строке на 123-м месте (считая слева направо)?

Решение

Самое первое, что приходит в голову (особенно если под рукой — компьютер с текстовым редактором, в котором реализованы операции блочного копирования и вставки текста) — просто расписать все указанные в задаче цепочки:

0 — 0

1 — 001

2 — 0010012

3 — 001001200100123

4 — 0010012001001230010012001001234

5 — 001001200100123001001200100123400100120010012300100120010012345

6 — 0010012001001230010012001001234001001200100123001001200100123450010012
0010012300100120010012340010012001001230010012001001234560,

а затем честно отсчитать слева направо нужную (123-ю) цифру. Однако, подобный способ решения крайне нерационален, а на реальном ЕГЭ в условиях ограниченного времени на решение задач и без наличия упомянутого текстового редактора под рукой рекомендовать такое решение и вовсе нельзя.

Попробуем обойтись без выписывания цепочек. Вместо этого научимся «расплетать» предполагаемую цепочку по шагам, чтобы найти в ней нужное.

Сначала следует посмотреть ещё раз на принцип формирования цепочек. Первая по счёту (и нулевая по порядковому номеру) цепочка имеет длину 1. Далее на каждом очередном шаге длина цепочки удваивается, а затем увеличивается ещё на 1 (приписыванием в конце очередной цифры — номера данной цепочки):

№ цепочки (i)	0	1	2	3	4	5	6
Длина цепочки	1	3	7	15	31	63	127

Отсюда можно вывести формулу определения длины цепочки по её номеру i:

$$<\textit{длина цепочки}> = 2^{i+1} - 1$$

Если цепочки нумеровались бы не с нуля, а с единицы при соблюдении всех прочих условий, то в данной формуле в показателе степени для двойки прибавлять единицу было бы не нужно.

Таким образом, искомая цифра (на 123-м месте) отстоит от конца итоговой цепочки на 4 позиции левее. Вспомнив, что на каждом шаге к удваиваемой цепочке каждый раз дописывалась одна цифра, равная порядковому номеру очередной цепочки, нетрудно сообразить, что

итоговая цепочка будет завершаться цифрами: ...0123456, где «шестёрка» стоит на последнем, 127-месте. И, отсчитав 4 позиции влево, получить, что на 123-м месте находится цифра 2.

№ позиции	121	122	123	124	125	126	127
Цифра	0	1	2	3	4	5	6

Ответ: цифра 2.

Такие же сравнительно простые рассуждения нетрудно проводить и в других случаях, когда искомый символ располагается близко к концу цепочки (а точнее — не более чем в i позициях от её конца, где i — порядковый номер цепочки, считая с нуля), либо вблизи от позиции, соответствующей концу какой-либо составляющей её подцепочки, — номера этих «внутренних» позиций поможет определить таблица, построенная по вычисленным согласно вышеприведённой формуле значениям длин подцепочек.

№ цепочки (i)	0	1	2	3	4	5	6
Конечная позиция цепочки	1	3	7	15	31	63	127
Окончание подцепочки	0	←01	←012	←0123	←01234	←012345	←0123456

Стрелка ← условно обозначает предыдущую часть строки, составленную из удвоенных подстрок, полученных на предыдущем шаге.

Однако бывают и задачи, где требуется искать символ (либо символы) в середине цепочек. И тогда уже не обойтись без полноценного «расплетания».

Задача 5*. Строки (цепочки латинских букв) создаются по следующему правилу.

Первая строка состоит из одного символа — латинской буквы «A». Каждая из последующих цепочек создаётся такими действиями: в очередную строку сначала записывается буква, чей порядковый номер в алфавите соответствует номеру строки (на i-м шаге пишется «i»-я буква алфавита), к ней слева дважды подряд приписывается предыдущая строка.

Вот первые 4 строки, созданные по этому правилу:

(1) A

(2) AAB

(3) AABAABC

(4) AABAABCAABAABCD

Латинский алфавит (для справки):

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Запишите шесть символов подряд, стоящие в седьмой строке со 117-го по 122-е место (считая слева направо).

Решение

Сначала тем же самым способом, как и в предыдущей задаче, находится формула для вычисления длины итоговой цепочки, начав выписывать длины первых цепочек, приведённых в условии:

№ цепочки	Цепочка	Длина цепочки
1	A	1
2	AAB	3
3	AABAABC	7
4	AABAABCAABAABCD	15

Можно и сразу вывести искомую формулу: $<\textit{длина цепочки}> = 2^i - 1$.

Прибавление единицы в показателе степени двойки здесь не требуется, так как нумерация цепочек производится не с нуля, а с единицы.

Тогда, очевидно, длина итоговой, 7-й цепочки будет равна $2^7 - 1 = 127$ символов.

Теперь необходимо научиться «расплетать» цепочку по шагам назад. Итак...

1. Согласно правилам, указанным в условии задачи, итоговая, 7-я цепочка имеет формат:

(6)	(6)	G

При этом согласно выведенной формуле 6-я цепочка имеет длину $2^6 - 1 = 63$ символа. Тогда в таблицу, обозначающую формат рассматриваемой строки-цепочки, можно добавить обозначения соответствующих номеров позиций символов[1]:

(6)	(6)	G
1—63	64—126	127

2. Очевидно, что искомые символы располагаются во второй по счёту подстроке (6). Продолжается «расплетание» цепочек ещё на один шаг назад, «раскрывая» эту вторую цепочку (6) и оставляя неизменными остальные части таблицы. При этом учитывается, что длина предыдущей, 5-й цепочки равна $2^5 - 1 = 31$ символу.

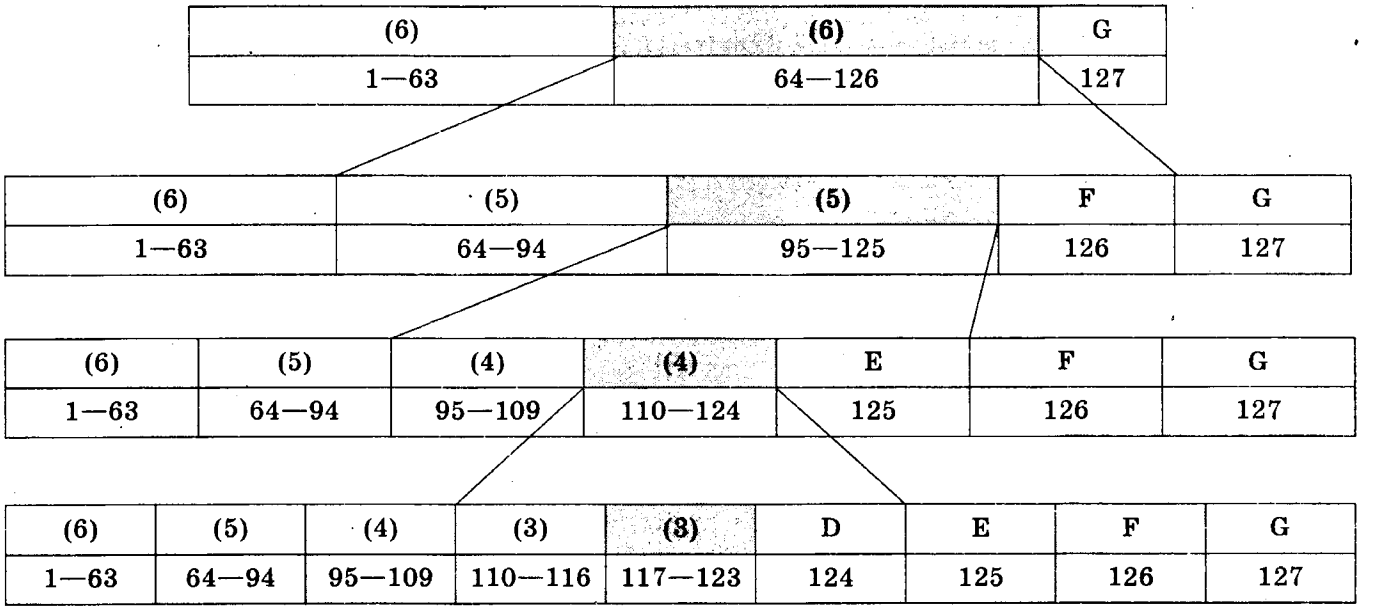
(6)	(6)	G		
1—63	64—126	127		
(6)	(5)	(5)	F	G
1—63	64—94	95—125	126	127

3. Искомые символы находятся во второй «раскрытой» цепочке (5). Она «расплетается» после вычисления длины цепочки (4): $2^4 - 1 = 15$ символов.

(6)	(6)	G				
1—63	64—126	127				
(6)	(5)	(5)	F	G		
1—63	64—94	95—125	126	127		
(6)	(5)	(4)	(4)	E	F	G
1—63	64—94	95—109	110—124	125	126	127

[1] При этом нужно помнить, что отсчёт номеров позиций (как и отсчёт дней по календарю) ведётся по следующим правилам: если предыдущая подцепочка начинается с символа с порядковым номером n и имеет длину d, то номер символа, которым заканчивается эта подцепочка, вычисляется по формуле $(n + d - 1)$, а следующая подцепочка будет начинаться с символа под номером $(n + d)$.

4. Искомые символы находятся во второй «раскрытой» цепочке (4), «расплетается» и она (вычислив, что длина цепочки (3) равна $2^3 - 1 = 7$ символов):



(6)	**(6)**	G
1—63	64—126	127

(6)	(5)	**(5)**	F	G
1—63	64—94	95—125	126	127

(6)	(5)	(4)	**(4)**	E	F	G
1—63	64—94	95—109	110—124	125	126	127

(6)	(5)	(4)	(3)	**(3)**	D	E	F	G
1—63	64—94	95—109	110—116	117—123	124	125	126	127

5. В данном случае: все нужные символы находятся в одной подцепочке (вторая (3)). А её уже нетрудно выписать полностью, «подсмотрев» в условии задачи: AABAABC.

A	**A**	**B**	**A**	**A**	**B**	C
117	**118**	**119**	**120**	**121**	**122**	123

И тут очевидно, что искомые символы, записанные на позициях 117—122, — это AABAAB.

Ответ: искомые символы AABAAB.

Задача 6*. Цепочки символов (строки) создаются по следующему правилу: первая строка состоит из одного символа — цифры «1». Каждая из последующих цепочек создаётся такими действиями: в начало записывается число — номер строки по порядку (для i-й строки ставится число «i»), далее дважды подряд записывается предыдущая строка.

Вот первые 4 строки, созданные по этому правилу:

(1) 1

(2) 211

(3) 3211211

(4) 432112113211211

Сколько раз встречается цифра «1» в первых семи строках (суммарно)?

Решение

Можно попытаться вывести общую формулу, аналогично тому, как ранее определялась формула для вычисления длины цепочек.

№ цепочки	Цепочка	Кол-во единиц
1	1	1
2	211	2
3	3211211	4
4	432112113211211	8

Формула: 2^{i-1}, где i — номер цепочки.

Формула действительна до цепочки с номером 9 включительно; далее надо учитывать, что в начале на каждом шаге должны дописываться двухзначные числа 10, 11, 12, ..., потом — трёхзначные и т. д., но в задачах ЕГЭ пока до таких номеров строк составители заданий не доходили.

Вывод подобных формул представляет дополнительные затруднения. Поэтому для решения таких задач можно предложить более простой табличный метод. Смысл его заключается в том, что та или иная цифра i впервые появляется в цепочке с номером i (что очевидно), а дальше на каждом шаге количество таких цифр удваивается. Легко составить таблицу количеств каждой интересующей нас цифры, а потом в графе, соответствующей требуемой строке, подсчитать сумму количеств этих цифр.

В данном случае требуется подсчитать количество единиц. Единицы начинают появляться сразу с первой же цепочки. Таблица будет выглядеть так:

№ цепочки	1	2	3	4	5	6	7	Суммарно:
Цифра «1»	1	2	4	8	16	32	64	127

Можно не суммировать значения степеней двоек. Вспомнив принципы двоичной арифметики (формулу записи перевода двоичных чисел в десятичный формат) нетрудно догадаться, что указанная сумма может быть записана в двоичном виде как число 1111111_2. А оно, в свою очередь, равно $(10000000 - 1)_2 = 2^8 - 1 = 127$.

Ответ: 127 единиц.

Задача 7. Цепочки символов (строки) создаются по следующему правилу: первая строка состоит из одного символа — цифры «0». Каждая из последующих цепочек создаётся такими действиями: в начале дважды подряд записывается предыдущая строка, а затем записывается число — номер строки по порядку (для i-й строки ставится число «i»).

Вот первые 4 строки, созданные по этому правилу:

(0) 0
(1) 001
(2) 0010012
(3) 001001200100123

и т.д.

Сколько раз встречается цифра «1» в строке с номером 9?

Решение

При решении данной задачи следует учитывать, что в самой первой цепочке (с номером 0) единиц нет вообще, впервые единица появляется в цепочке с номером 1, а далее в таблице записывается все та же последовательность степеней двойки. В результате составленная таблица имеет вид:

№ цепочки	0	1	2	3	4	5	6	7	8	9
Цифра «1»	—	1	2	4	8	16	32	64	128	256

Ответ: 256.

Задача 8*. Цепочки символов (строки) создаются по следующему правилу: первая строка состоит из одного символа — цифры «1». Каждая из последующих цепочек создаётся следующим действием: в очередную строку дважды записывается предыдущая цепочка цифр (одна за другой, подряд), а в конец приписывается ещё одно число — номер строки по порядку (на i-м шаге дописывается число «i»).

Вот первые 4 строки, созданные по этому правилу:
(1) 1
(2) 112
(3) 1121123
(4) 112112311211234
Сколько раз в общей сложности встречаются в восьмой строке четные цифры (2, 4, 6, 8)?

Решение
Можно вывести общую формулу:

№ цепочки	Цепочка	Кол-во четных цифр
1	1	0
2	112	1
3	1121123	2
4	112112311211234	5
5	...	10
6	...	21
7	...	42
8	...	85

Формула (рекуррентная):
$N_i = N_{i-1} \cdot 2 + ((i - 1) \bmod 2)$, при $N_1 = 0$.

Данная формула также действительна до цепочки с номером 9 включительно; далее надо учитывать, что в начале на каждом шаге должны дописываться двухзначные числа 10, 11, 12, ..., потом — трёхзначные и т. д.

Без составления данной формулы можно использовать табличный способ. В данном случае таблица составляется точно так же — практически «механической» записью чисел — степеней двойки, но будет содержать несколько строк — по одной для каждой чётной цифры:

№ цепочки / Цифра	1	2	3	4	5	6	7	8
2	—	1	2	4	8	16	32	64
4	—	—	—	1	2	4	8	16
6	—	—	—	—	—	1	2	4
8	—	—	—	—	—	—	—	1
							Всего четных цифр:	**85**

Ответ: 85 чётных цифр.

Задача 9. Цепочки символов (строки) создаются по следующему правилу: первая строка состоит из цифры «1». В последующих цепочках сначала записывается число — номер строки по порядку (на i-м шаге записывается число i), а после него дважды записывается предыдущая цепочка цифр (одна за другой, подряд).

Вот первые 4 строки, созданные по этому правилу:
(1) 1
(2) 211
(3) 3211211
(4) 432112113211211
Сколько раз в 10-й строке встречаются нечётные цифры (1, 3, 5, 7, 9)?

Решение

Для решения этой задачи составляется таблица, аналогичная предыдущей. Единственное, на что нужно обязательно обратить внимание, — что в 10-й строке кроме единиц, накапливаемых за счёт удвоения предыдущей подцепочки, в начале дописывается новое число 10, которое тоже содержит единицу. Поэтому в последнем столбце в строке, соответствующей цифре «1», к очередной степени двойки (512) надо прибавить 1. А затем подсчитать сумму чисел для всех строк последнего столбца.

№ цепочки / Цифра	1	2	3	4	5	6	7	8	9	10
1	1	2	4	8	16	32	64	128	256	512+1
3	—	—	1	2	4	8	16	32	64	128
5	—	—	—	—	1	2	4	8	16	32
7	—	—	—	—	—	—	1	2	4	8
9	—	—	—	—	—	—	—	—	1	2
									Всего нечетных цифр:	**683**

Очевидно, что если в задаче будет ставиться вопрос о количестве цифр в цепочках с номерами больше 10, то для 11-й и т.д. строк надо учитывать как удвоение этой дополнительной единицы на каждом шаге, так и добавление одной или двух (для 11-й цепочки) новых единиц. То же самое делать для других цифр, когда они будут появляться в соответствующем номере цепочки.

Ответ: 683 нечётные цифры.

Задача 10. Из цифр формируются цепочки (строки) по следующему правилу:

- первая строка состоит из одной цифры 1;
- вторая строка состоит из двух цифр — 23;
- далее любая цепочка с номером n составляется из двух цепочек — сначала переписывается цепочка с номером $(n - 2)$, а потом к ней дописывается цепочка с номером $(n - 1)$.

Вот первые несколько таких строк:

(1) 1
(2) 23
(3) 123
(4) 23123
...

Сколько нечётных и сколько чётных цифр будет в строке с номером 7?

Решение

Сначала следует проанализировать состав получаемых цепочек и постараться вывести общую формулу. Либо, как альтернативный вариант при наличии достаточного времени, — просто «честно» расписать все получающиеся цепочки.

№ цепочки (n)	Цепочка	Кол-во чётных цифр	Кол-во нечётных цифр
1	1	0	1
2	23	1	1
3	123	1	2
4	23123	2	3
5	12323123	3	5
6	2312312323123	5	8
7	123231232312312323123	8	13

Проанализировав получаемые последовательности количеств чётных и нечётных цифр, можно прийти к выводу, что это не что иное как числа Фибоначчи! Только для количества чётных цифр последовательность Фибоначчи начинается с пары чисел 0 и 1, а для нечётных — с пары чисел 1 и 1.

Почему получились именно числа Фибоначчи, — понять нетрудно. Согласно правилу записи цепочек, в каждую n-ю цепочку копируются две предыдущих, а значит, получаемые количества чётных и нечётных цифр в n-й цепочке есть сумма количеств таких цифр в предыдущих двух цепочках. А это — как раз и есть принцип формирования последовательности Фибоначчи.

Теперь становится понятно, что любые подобные задачи на вычисление количеств указанных символов (чисел или букв) в цепочках, получаемых переписыванием двух предыдущих — $(n - 2)$ и $(n - 1)$, — можно легко решать, даже не расписывая сами цепочки. Достаточно подсчитать количество заданных символов в первых двух «стартовых» цепочках, а потом выстроить на базе этих двух чисел последовательность Фибоначчи.

Ответ: 8, 13.

Задачи для самостоятельного решения

1. Все 5-буквенные слова, составленные из букв Т, О, Н, записаны в алфавитном порядке. Вот начало списка:
   1. ТТТТТ
   2. ТТТТО
   3. ТТТТН
   4. ТТТОТ

   Сколько букв Т встречается в слове, стоящем на 101-м месте от начала списка.
2. Все 5-буквенные слова, составленные из букв D, O, M, записаны в алфавитном порядке и пронумерованы. Вот начало списка:
   1. DDDDD
   2. DDDDO
   3. DDDDM
   4. DDDOD

   Какое слово будет стоять в списке под номером 242?
3. Все 5-буквенные слова, составленные из букв Е, Г, Э, записаны в алфавитном порядке и пронумерованы. Вот начало списка:
   1. ЭЭЭЭЭ
   2. ЭЭЭЭЕ
   3. ЭЭЭЭГ
   4. ЭЭЭЕЭ
   5. ЭЭЭЕЕ

   Какое слово будет стоять в списке под номером 241?
4. Все 5-буквенные слова, составленные из букв К, О, Т, записаны в алфавитном порядке. Вот начало списка:
   1. ТТТОК
   2. ТТТКТ
   3. ТТТКО
   4. ТТТКК
   5. ТТОТТ

   .....

   Есть ли в слове, стоящем на 50 месте, хотя бы одна буква К?

5. Все 5-буквенные слова, составленные из букв О, Т, П, записаны в алфавитном порядке. Вот фрагмент списка:

.....
4. ОООПП
5. ОООПТ
6. ОООТО
7. ОООТП
8. ОООТТ
.....

Сколько слов в списке находится между словами ПОТОП и ТОПОТ

6. Все 5-буквенные слова, составленные из букв К, Р, О, Т записаны в алфавитном порядке и пронумерованы.

Вот начало списка:

1. РРРРР
2. РРРРТ
3. РРРРО
4. РРРРК
5. РРРТР
.....

Запишите слово, которое стоит под номером 1024.

7. Строки из латинских букв создаются по следующему правилу:

— первая строка состоит из буквы «А»;

— в каждой из последующих строк сначала дважды подряд записывается предыдущая строка, а затем справа приписывается буква, порядковый номер которой в алфавите соответствует номеру строки (на *i*-*м* шаге пишется *i*-я буква алфавита).

Вот первые 4 строки, созданные по этому правилу:

(1) A
(2) AAB
(3) AABAABC
(4) AABAABCAABAABCD
.....

Латинский алфавит (для справки): ABCDEFGHIJKLMNOPQRSTUVWXYZ

Запишите шесть символов подряд, которые записаны в восьмой строке с 100-го по 105-е место (считая слева направо).

8. Строки (цепочки символов латинских букв) создаются по следующему правилу:

— первая строка состоит из одного символа — цифры 1;

— каждая из последующих цепочек создаётся такими действиями: в очередную строку сначала записывается цифра, соответствующая номеру строки (на *i*-м шаге пишется цифра *i*), к ней слева дважды подряд приписывается предыдущая строка.

Вот первые 4 строки, созданные по этому правилу:

(1) 1
(2) 112
(3) 1121123
(4) 112112311211234
...

Запишите шесть цифр подряд, стоящих в седьмой строке со 116-го по 121-е место (считая слева направо).

9. Строки (цепочки цифр) создаются по следующему правилу:

— первая строка состоит из одного символа — цифры 1;

— каждая из последующих цепочек создаётся такими действиями: в очередную строку дважды подряд записывается предыдущая строка, а в конец приписывается ещё одна цифра, соответствующая номеру строки (на *i*-м шаге дописывается цифра *i*).

Вот первые 4 строки, созданные по этому правилу:

(1) 1
(2) 112
(3) 1121123
(4) 112112311211234
...
Сколько раз в седьмой строке встретится цифра 1?

10. Строки (цепочки цифр) создаются по следующему правилу:
— первая строка состоит из одного символа — цифры 1;
— каждая из последующих цепочек создается следующим действием: в очередную строку дважды записывается предыдущая строка (цепочка за цепочкой, подряд), а в конец приписывается ещё один символ — цифра, соответствующая номеру строки (на i-м шаге дописывается цифра i).
Вот первые 4 строки, созданные по этому правилу:
(1) 1
(2) 112
(3) 1121123
(4) 112112311211234
Сколько раз в общей сложности встречаются в седьмой строке цифры 1, 6 и 7?

Ответы для самопроверки

№ задания	Ответ
1	2 буквы «А»
2	РРРРО
3	ИИИИА
4	да 1 буква «К»
5	72 слова
6	КККККК
7	ВСААВА
8	311211
9	64
10	67

Основы логики

A3, A10 Таблицы истинности. Законы алгебры логики. Задачи, решаемые с использованием таблиц истинности

Конспект

В алгебре логики изучаются логические операции, производимые над высказываниями. Высказывания могут быть истинными или ложными. Применяя к простым высказываниям логические операции, можно строить составные высказывания.

Основными логическими операциями являются:

Отрицание (инверсия, логическое НЕ)

Смысл операции: результат меняется на противоположный (вместо истины — ложь, вместо лжи — истина).

Обозначение: ¬

Таблица истинности:

A	¬A
0	1
1	0

Логическое сложение (дизъюнкция, логическое ИЛИ)

Смысл операции: результат — истина, если хотя бы один операнд — истина (операндом называется то значение или та переменная, над которым (которой) осуществляется операция).

Обозначение: $\vee$ или +

Таблица истинности:

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

Логическое умножение (конъюнкция, логическое И)

Смысл операции: результат — истина, если оба операнда — истина.

Обозначение: $\wedge$ или &

Таблица истинности:

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

Следование (импликация)

Смысл операции: из лжи может следовать что угодно, а из истины — только истина.

Обозначение: $\rightarrow$

Таблица истинности:

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

Равносильность (эквиваленция)

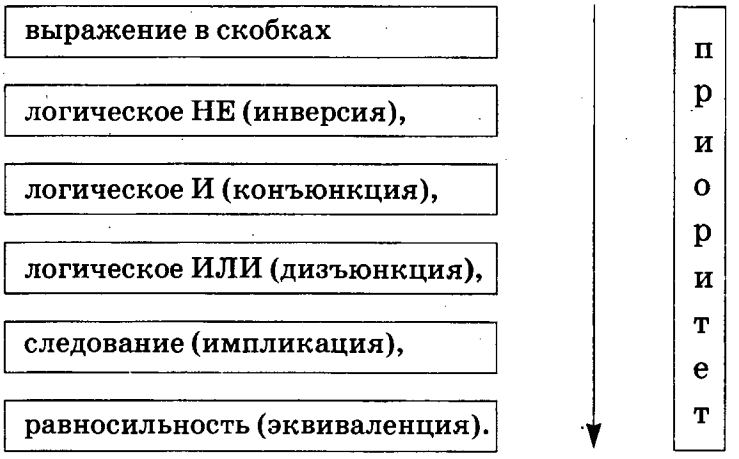
Смысл операции: результат — истина, если операнды одинаковые.

Обозначение: $\equiv$ или $\leftrightarrow$

Таблица истинности:

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

Если в логическом выражении используется несколько логических операций, их порядок определяется *приоритетами логических операций:*



Операцию «импликация» можно выразить через «ИЛИ» и «НЕ»:

$A \rightarrow B = \neg A \vee B$

Операцию «эквиваленция» также можно выразить через «ИЛИ» и «НЕ»:

$A \equiv B = \neg A \wedge \neg B \vee A \wedge B$

Основные законы алгебры логики

Название закона	Формулировка
Переместительный закон	$A \vee B = B \vee A$ $A \wedge B = B \wedge A$
Сочетательный закон	$(A \vee B) \vee C = A \vee (B \vee C)$ $(A \wedge B) \wedge C = A \wedge (B \wedge C)$

Окончание табл.

Название закона	Формулировка
Распределительный закон	$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$
Закон непротиворечия. Этот закон выражает тот факт, что высказывание не может быть одновременно истинным и ложным	$A \wedge \neg A = 0$
Закон исключённого третьего. Этот закон означает, что либо высказывание, либо его отрицание должно быть истинным	$A \vee \neg A = 1$
Закон двойного отрицания	$\neg(\neg A) = A$
Законы де Моргана	$\neg(A \vee B) = \neg A \wedge \neg B$ $\neg(A \wedge B) = \neg A \vee \neg B$
Законы переменной с самой собой	$A \vee A = A$ $A \wedge A = A$
Законы нуля и единицы	$A \wedge 0 = 0$ $A \wedge 1 = A$ $A \vee 0 = A$ $A \vee 1 = 1$
Законы поглощения	$A \vee (A \wedge B) = A$ $A \wedge (A \vee B) = A$ $A \vee (\neg A \wedge B) = A \vee B$

Разбор типовых задач

Задача 1*. Какое логическое выражение равносильно выражению $\neg(\neg A \vee B) \vee \neg C$?

1) $(A \vee \neg B) \vee \neg C$
2) $\neg A \vee B \vee \neg C$
3) $A \vee \neg B \vee \neg C$
4) $(\neg A \wedge B) \vee \neg C$

Решение

Исходное выражение преобразовывается, используя законы алгебры логики:
$\neg(\neg A \vee B) \vee \neg C$ = {закон де Моргана} = $(A \wedge \neg B) \vee \neg C$.
Полученное выражение совпадает с вариантом ответа №1.
Ответ: $(A \wedge \neg B) \vee \neg C$ (вариант ответа №1).

Задача 2*. Символом F обозначено одно из указанных ниже логических выражений от трёх фрагментов: X, Y, Z.

Дан фрагмент таблицы истинности выражения F:

X	Y	Z	F
0	1	1	0
1	1	1	1
0	0	1	1

Какое выражение соответствует F?

1) X ∧ BY ∧ BZ
2) BX ∧ BY ∧ Z
3) BX ∨ BY ∨ Z
4) X ∨ BY ∨ BZ

Решение

Обычно в таких задачах даётся только фрагмент таблицы истинности, поэтому пытаться решать задачу «в лоб», выводя соответствующее таблице логическое выражение и сравнивая его с вариантами ответов, бессмысленно. Лучше всего просто проверять предлагаемые ответы один за другим, поочерёдно подставляя в соответствующее логическое выражение значения переменных из каждой строки таблицы и проверяя, получается ли в результате указанное в той же строке таблицы требуемое значение F.

Если для какой-то из строк таблицы получается неправильный результат, то можно прервать проверку данного варианта ответа и сразу перейти к следующему варианту.

Проверка первого варианта ответа: X ∧ ¬Y ∧ ¬Z. Операция И даёт значение 1 только когда все значения переменных равны 1. (Цветом отмечены строки таблицы, в которых результат вычисления выражения не совпадает с заданным значением F.)

X	Y	Z	X ∧ ¬Y ∧ ¬Z	F
0	1	1	0	0
1	1	1	0	1
0	0	1		1

Проверка второго варианта ответа: ¬X ∧ ¬Y ∧ Z.

X	Y	Z	¬X ∧ ¬Y ∧ Z	F
0	1	1	0	0
1	1	1	0	1
0	0	1		1

Проверка третьего варианта ответа: ¬X ∨ ¬Y ∨ Z. Операция ИЛИ даёт значение 1, если значение хотя бы одной переменной равно 1.

X	Y	Z	¬X ∨ ¬Y ∨ Z	F
0	1	1	1	0
1	1	1		1
0	0	1		1

Проверка четвёртого варианта ответа: X ∨ ¬Y ∨ ¬Z.

X	Y	Z	X ∨ ¬Y ∨ ¬Z	F
0	1	1	0	0
1	1	1	1	1
0	0	1	1	1

Таким образом, правильным является четвёртый вариант ответа.

Ответ: вариант ответа №4.

Задача 3. Дан фрагмент таблицы истинности выражения F:

z_1	z_2	z_3	z_4	z_5	z_6	z_7	F
0	1	0	1	1	1	0	0
1	0	1	1	0	0	1	0
0	1	0	1	1	0	1	0

Каким выражением может быть F?

1) $z_1 \to (z_2 \wedge z_3 \vee z_4 \wedge z_5 \vee z_6 \wedge z_7)$
2) $z_2 \to (z_1 \wedge z_3 \vee z_4 \wedge z_5 \vee z_6 \wedge z_7)$
3) $z_3 \to (z_1 \wedge z_2 \vee z_4 \wedge z_5 \vee z_6 \wedge z_7)$
4) $z_4 \to (z_1 \wedge z_2 \vee z_3 \wedge z_5 \vee z_6 \wedge z_7)$

Решение

Общий принцип решения — тот же, что и для предыдущей задачи, — только логические выражения более сложные.

Проверка первого варианта ответа: $z_1 \to (z_2 \wedge z_3 \vee z_4 \wedge z_5 \vee z_6 \wedge z_7)$.

Необходимо помнить приоритеты логических операций и таблицу истинности логической операции следования. (Цветом отмечены строки таблицы, в которых результат вычисления выражения не совпадает с заданным значением F. При обнаружении строки, в которой значение F не совпадает с результатом вычисления выражения, анализ дальнейших строк таблицы не производится.)

z_1	z_2	z_3	z_4	z_5	z_6	z_7	F
0	1	0	1	1	1	0	0
1	0	1	1	0	0	1	0
0	1	0	1	1	0	1	0

$0 \to (1 \wedge 0 \vee 1 \wedge 1 \vee 1 \wedge 0) =$
$= 0 \to (0 \vee 1 \vee 0) = 0 \to 1 = 1$

Проверка второго варианта ответа: $z_2 \to (z_1 \wedge z_3 \vee z_4 \wedge z_5 \vee z_6 \wedge z_7)$.

z_1	z_2	z_3	z_4	z_5	z_6	z_7	F
0	1	0	1	1	1	0	0
1	0	1	1	0	0	1	0
0	1	0	1	1	0	1	0

$1 \to (0 \wedge 0 \vee 1 \wedge 1 \vee 1 \wedge 0) =$
$= 1 \to (0 \vee 1 \vee 0) = 1 \to 1 = 1$

Проверка третьего варианта ответа: $z_3 \to (z_1 \wedge z_2 \vee z_4 \wedge z_5 \vee z_6 \wedge z_7)$.

z_1	z_2	z_3	z_4	z_5	z_6	z_7	F
0	1	0	1	1	1	0	0
1	0	1	1	0	0	1	0
0	1	0	1	1	0	1	0

$0 \to (0 \wedge 1 \vee 1 \wedge 1 \vee 1 \wedge 0) =$
$= 0 \to (0 \vee 1 \vee 0) = 0 \to 1 = 1$

Проверка четвёртого варианта ответа: $z_4 \to (z_1 \wedge z_2 \vee z_3 \wedge z_5 \vee z_6 \wedge z_7)$.

z_1	z_2	z_3	z_4	z_5	z_6	z_7	F
0	1	0	1	1	1	0	0
1	0	1	1	0	0	1	0
0	1	0	1	1	0	1	0

$1 \to (0 \wedge 1 \vee 0 \wedge 1 \vee 1 \wedge 0) =$
$= 1 \to (0 \vee 0 \vee 0) = 1 \to 0 = 0;$
$1 \to (1 \wedge 0 \vee 1 \wedge 0 \vee 0 \wedge 1) =$
$= 1 \to (0 \vee 0 \vee 0) = 1 \to 0 = 0;$
$1 \to (0 \wedge 1 \vee 0 \wedge 1 \vee 0 \wedge 1) =$
$= 1 \to (0 \vee 0 \vee 0) = 1 \to 0 = 0$

Таким образом, правильным является четвёртый вариант ответа.

Ответ: вариант ответа №4.

Задача 4*. Какое из приведённых имён удовлетворяет логическому условию:
¬(последняя буква гласная → первая буква согласная) ∧ вторая буква согласная.
1) ИРИНА
2) АРТЁМ
3) СТЕПАН
4) МАРИЯ

Решение

«Хитрость» решения данной задачи состоит в том, что нужно сначала преобразовать запись логического выражения в более привычную таблицу двоичных значений 0 и 1. Строки этой таблицы соответствуют вариантам ответов.

Составляется таблица. Для справки: операция И даёт значение 1 только когда все значения переменных равны 1, а операция следования (→) даёт результат 0 только в одном случае — когда из 1 следует 0.

Имя	X1: последняя буква гласная	X2: первая буква согласная	X3: вторая буква согласная	X4: X1 X2	X5: ¬X4	Результат: X5 ∧ X3
ИРИНА	1	0	1	0	1	1
АРТЁМ	0	0	1	1	0	0
СТЕПАН	0	1	1	1	0	0
МАРИЯ	1	1	0	1	0	0

В таблице выделена строка, соответствующая правильному ответу (первому).

Обнаружив, что условие выполняется для какого-то варианта, остальные варианты ответа можно не проверять.

Ответ: ИРИНА (вариант ответа №1).

Задача 5*. Для какого из указанных значений X истинно высказывание
¬((X > 2) → (X > 3))?
1) 1 2) 2 3) 3 4) 4

Решение

Такие задачи (определение числа, для которого истинно логическое высказывание) «родственны» рассмотренной выше и решаются аналогично. Только в данном случае вместо условий, накладываемых на отдельные буквы имён, рассматриваются логические условия типа «больше» / «меньше».

Составляется таблица. Для справки: операция следования (→) даёт результат 0 только в одном случае — когда из 1 следует 0.

Значение X	Y1: X > 2	Y2: X > 3	Y3: Y1 Y2	Результат: ¬Y3
1	0	0	1	0
2	0	0	1	0
3	1	0	0	1
4	1	1	1	0

Ответ: число 3 (вариант ответа №3).

Задача 6*. Каково наибольшее целое число X, при котором истинно высказывание $(50 < X \cdot X) \rightarrow (50 > (X + 1) \cdot (X + 1))$?

Решение

Задача является более сложной: здесь требуется полностью решить задачу, а не проверять четыре предложенных варианта ответа. Зато в этой задаче добавлено ещё одно дополнительное условие: искомое число должно быть целым и наибольшим. Такое условие означает, что в результате решения логического уравнения получить диапазон значений X, который распространяется от требуемого наибольшего значения в сторону уменьшения. Решать задачу лучше не табличным способом, а при помощи графической схемы, которая представляет собой пересекающиеся интервалы на числовой прямой.

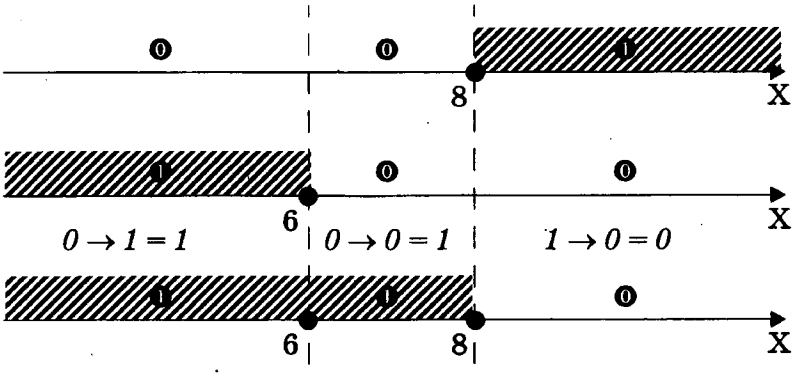
Способ 1.

1. Строится интервал для условия $50 < X \cdot X$. При этом данное условие преобразуется в эквивалентное условие $X \cdot X \geqslant 50$, помня, что речь идёт только о целых числах, о значениях $X \geqslant 8$.

2. Строится интервал для условия $50 > (X + 1) \cdot (X + 1)$, преобразовывая его в эквивалентное условие $(X + 1) \cdot (X + 1) \leqslant 50$, откуда $X \leqslant 6$.

3. Построенные интервалы располагаются один под другим. Заштрихованные значения соответствуют логической единице, а не заштрихованные — нулю.

Операция следования «направлена сверху вниз». Она даёт значение 0 только когда из 1 следует 0, а в остальных случаях получается значение 1 (истина). Поэтому, разбив числовую ось на соответствующие интервалы, легко построить требуемый «результирующий» интервал.



Важно учитывать, что в исходных интервалах граничные точки принадлежат соответствующим интервалам и потому на эти точки полностью распространяется заданная логическая операция. Следовательно в результирующем интервале граничная точка 8 (для которой выполняется равенство $1 \rightarrow 0 = 0$) уже не будет входить в результирующий «единичный» интервал. Следовательно, наибольшее целое значение X, удовлетворяющее условию задачи, равно 7.

Способ 2.

1. Наоборот, сначала рассматривается, в каких ситуациях истинна операция следования. Известно, что результат её выполнения является ложным в единственном случае — когда из

1 следует 0, и истинным в остальных случаях. Тогда исходное выражение можно представить как три возможных варианта систем неравенств:

$$(50 < X \cdot X) \to (50 > (X+1) \cdot (X+1))$$

$$\begin{cases} 50 < X \cdot X, & \text{❶} \\ 50 > (X+1) \cdot (X+1); & \text{❶} \end{cases} \qquad \begin{cases} 50 < X \cdot X, & \text{⓪} \\ 50 > (X+1) \cdot (X+1); & \text{❶} \end{cases} \qquad \begin{cases} 50 < X \cdot X, & \text{⓪} \\ 50 > (X+1) \cdot (X+1); & \text{⓪} \end{cases}$$

Для поиска решений этих неравенств нужно в тех случаях, когда неравенство ложно (логическое значение 0), изменить знак неравенства на противоположный, причём строгое неравенство преобразуется в нестрогое и наоборот:

$$\begin{cases} 50 < X \cdot X, \\ 50 > (X+1) \cdot (X+1); \end{cases} \qquad \begin{cases} 50 \geqslant X \cdot X, \\ 50 > (X+1) \cdot (X+1); \end{cases} \qquad \begin{cases} 50 \geqslant X \cdot X, \\ 50 \leqslant (X+1) \cdot (X+1); \end{cases}$$

$$\Downarrow \qquad\qquad \Downarrow \qquad\qquad \Downarrow$$

$$\begin{cases} X \cdot X > 50, \\ (X+1) \cdot (X+1) < 50. \end{cases} \qquad \begin{cases} X \cdot X \leqslant 50, \\ (X+1) \cdot (X+1) < 50. \end{cases} \qquad \begin{cases} X \cdot X \leqslant 50, \\ (X+1) \cdot (X+1) \geqslant 50. \end{cases}$$

Очевидно, что значение $(X + 1) \cdot (X + 1)$ заведомо больше, чем значение $X \cdot X$. Поэтому первая система неравенств не имеет решения.

Решение (в целых числах) второй системы неравенств — интервал значений $[-\infty,6]$. Он определяется по второму неравенству этой системы:

$(X + 1) \cdot (X + 1) < 50 \Rightarrow X + 1 < \sqrt{50} \Rightarrow X + 1 \leqslant 7 \Rightarrow X \leqslant 6$.

Решение (в целых числах) третьей системы неравенств — пересечение интервалов значений $[-\infty, 7]$ и $[7, +\infty]$, которое равно числу 7.

Из найденных двух чисел — решений систем уравнений требуется наибольшее. Оно равно 7.

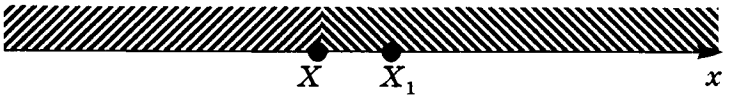
Способ 3.

Операция следования даёт результат «Истина» в трёх случаях из четырёх возможных, а результат «Ложь» — только в одном случае из четырёх, когда из 1 («Истина») следует 0 («Ложь»). Поэтому удобнее решать предложенную задачу в «обратной» формулировке:

Каково наименьшее целое число, при котором ложно высказывание $(50 < X \cdot X) \to (50 > (X + 1) \cdot (X + 1))$?

Прежде всего следует доказать правомерность такой замены формулировки задачи.

Предполагается, что искомое значение X в первоначальной задаче найдено. Раз в исходной задаче оно названо наибольшим целым числом, то это означает, что заданное выражение истинно на диапазоне значений $[-\infty, X]$. Тогда на оставшейся части числовой оси, т.е. в диапазоне $[X, +\infty]$ данное выражение будет ложно. А значит, определив наименьшее целое значение X_1, при котором заданное выражение ложно, легко получить искомое наибольшее X, при котором это выражение истинно: $X = X_1 - 1$.



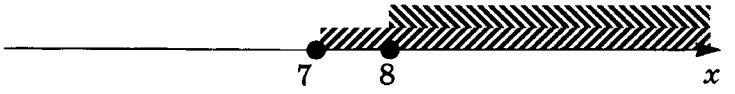
Решение задачи по предложенному новому способу.

1) Выражение $(50 < X \cdot X) \to (50 > (X + 1) \cdot (X + 1))$ ложно, когда

$$\begin{cases} 50 < X \cdot X \text{ — } \textit{истинно;} \\ 50 > (X+1) \cdot (X+1) \text{ — } \textit{ложно.} \end{cases}$$

2) Первой операции сравнения соответствует диапазон целых чисел [8, +∞]. Второй операции сравнения (учитывая, что она должна быть ложной, т. е. истинно сравнение $50 \leqslant (X + 1) \cdot (X + 1)$) соответствует диапазон целых чисел [7, +∞].

3) Графическое представление решения «модифицированной» задачи.



4) То, что оба условия сравнения объединены в систему, означает, что они должны выполняться одновременно. Следовательно, решением этой системы уравнений является *пересечение* построенных интервалов (пересечение множества составляющих их целых чисел). Этот интервал ложности операции следования – [8, +∞].

5) Наименьшее целое значение (X_1), при котором заданное выражение ложно, равно 8.

Искомое значение X в первоначальной задаче (наибольшее целое число, при котором исходное логическое выражение истинно) на 1 меньше найденного значения X_1. Следовательно для исходной задачи ответ — число 7.

Ответ: 7

Задача 7. X, Y и Z — целые числа, для которых истинно высказывание:

$\neg(X = Y) \wedge ((X > Y) \to (Y > Z)) \wedge ((Y > X) \to (Z > Y))$.

Чему равно Y, если $X = 45$ и $Z = 43$?

Решение

Вместо того, чтобы сразу пытаться преобразовывать исходное выражение или пытаться интуитивно определить предполагаемый правильный ответ, заданные значения переменных X и Z подставляются в исходное выражение:

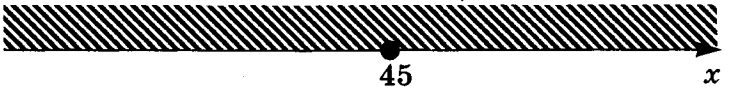
$\neg(45 = Y) \wedge ((45 > Y) \to (Y > 43)) \wedge ((Y > 45) \to (43 > Y))$

Теперь аналогия этой задачи с ранее рассмотренной (найти наименьшее или наибольшее целое значение, при котором истинно заданное выражение) становится очевидной. Соответственно, аналогичным может быть и её решение при помощи интервалов.

1) Выражение состоит из трёх компонентов, соединённых операцией «И». Значит, чтобы это выражение было истинным, нужно обеспечить истинность всех трёх этих компонентов:

$$\begin{cases} \neg(45 = Y) \text{ — } \textit{истинно;} \\ (45 > Y) \to (Y > 43) \text{ — } \textit{истинно;} \\ (Y > 45) \to (43 > Y) \text{ — } \textit{истинно.} \end{cases}$$

2) Для первого выражения этой системы получается интервал истинности (в целых числах): $[-\infty, 45] \cup [45, +\infty]$ (такая запись с объединением двух интервалов фактически означает всю числовую прямую, кроме числа 45).



3) Второй компонент системы: $(45 > Y) \to (Y > 43)$. Он будет истинным в трёх случаях из четырёх, поэтому проще решить «обратную» задачу — найти, при каких значениях Y это выражение ложно (один случай из четырёх), а потом взять значения на числовой прямой, которые *не входят* в найденный интервал ложности этого выражения.

Указанная операция следования ложна, если $\begin{cases} 45 > Y \text{ — } \textit{истинно;} \\ Y > 43 \text{ — } \textit{ложно.} \end{cases}$

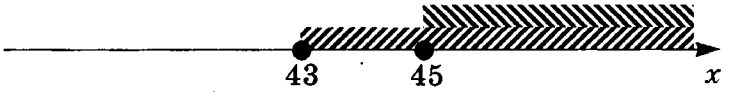
Первой части соответствует интервал истинности $[-\infty, 45]$, а второй — интервал истинности $[-\infty, 43]$. Тогда интервал ложности рассматриваемой операции следования: $[-\infty, 43]$.

Отсюда интересующий интервал истинности второго компонента исходного выражения: $[43, +\infty]$.

4) Аналогично рассматривается третий компонент системы: $(Y > 45) \to (43 > Y)$, применяя приём замены поиска интервала истинности выражения поиском интервала его ложности.

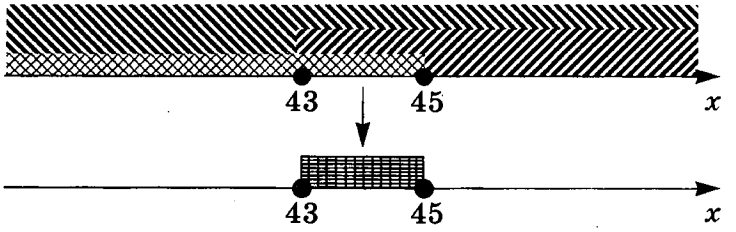
Указанная операция следования ложна, если $\begin{cases} Y > 45 \text{ — } \textit{истинно;} \\ 43 > Y \text{ — } \textit{ложно.} \end{cases}$

Первой части соответствует интервал истинности $[45,+\infty]$, а второй — интервал истинности $[43,+\infty]$. Тогда интервал ложности рассматриваемой операции следования: $[45,+\infty]$.



Отсюда интересующий интервал истинности третьего компонента исходного выражения: $[-\infty,45]$.

5) Возвратившись к системе этих трёх компонентов, определяется пересечение полученных для них интервалов истинности: $([-\infty, 45] \cup [45, +\infty]) \cap [43, +\infty] \cap [-\infty,45] = [43,45]$.



В итоге, интервал истинности исходного выражения — от 43 до 45, *не включая эти граничные точки*. Единственное возможное целочисленное решение этой задачи — число 44.

Ответ: число 44.

Графическое решение несколько длиннее, чем решение путем логических рассуждений. Однако оно выполняется быстрее, поскольку все операции выполняются «механически» (надо только быть внимательным при обмене местами левой и правой части в записи неравенств и при переходе от истинности к ложности операций сравнения), и более понятно благодаря его наглядности. Кроме того, графический способ универсален и приводит к правильному решению для любого исходного выражения.

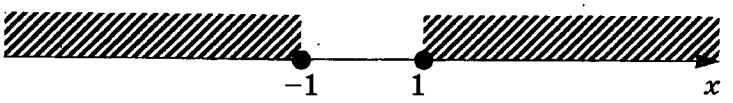
Задача 8. Сколько существует целых значений K, при которых ложно высказывание: $(|K| \geqslant 5) \vee (|K| < 1)$.

Решение

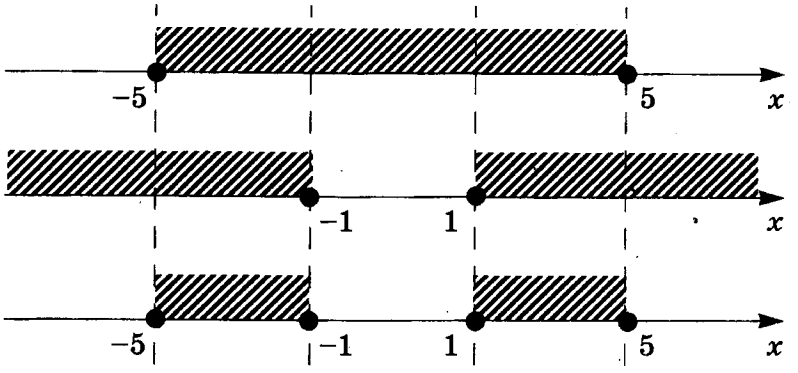
1) Два компонента данного выражения связаны операцией «ИЛИ», которая может быть ложной в одном случае из четырёх возможных — когда оба эти компонента ложны. Тогда это выражение эквивалентно системе: $\begin{cases} |K| \geqslant 5 \text{ — } \textit{ложно;} \\ |K| < 1 \text{ — } \textit{ложно.} \end{cases}$

2) Рассмотривая первый компонент, очевидно (если вспомнить понятие модуля), что его интервал истинности имеет вид: $[-\infty,-5] \cup [5,+\infty]$, а интервал ложности составляет всю оставшуюся часть числовой оси: $[-5,5]$.

3) Рассматривая второй компонент, получается, что его интервал истинности имеет вид: [−1, 1]. Тогда интервал ложности составляет: $[-\infty,-1] \cup [1,+\infty]$.



4) Строится пересечение полученных ранее интервалов для первого и второго компонентов:



5) В пределах полученного составного интервала $[-5,-1] \cup [1, 5]$, учитывая, что краевые значения −5 и 5 в него не входят, расположены следующие целые числа: −4, −3, −2, −1, 1, 2, 3, 4. Всего — 8 значений переменной *K*.

Ответ: 8 значений переменной.

Правила, которыми нужно руководствоваться при решении логических задач с интервалами:
1) при обмене местами левой и правой частей неравенства его знак меняется на противоположный;
2) если неравенство является ложным, то эквивалентное ему истинное неравенство не только имеет противоположный знак, но и становится из строгого нестрогим и наоборот; то же самое происходит при замене истинного неравенства эквивалентным ему ложным;
3) соединение компонентов логического выражения операцией «И» соответствует *пересечению* интервалов их истинности (интервалов значений, при которых эти компоненты истинны); соединение компонентов логического выражения операцией «ИЛИ» соответствует *объединению* интервалов их истинности;
4) интервал ложности представляет собой всю часть числовой прямой, кроме интервала истинности этого выражения — производится *вычитание* интервала истинности из числовой прямой; аналогично определяется и интервал истинности по интервалу ложности.
Кроме того, при решении задач с интервалами надо внимательно читать текст условия: если в вопросе фигурирует, например, «наибольшее натуральное число *X*» или «наибольшее целое положительное число *X*», то это означает добавление дополнительного условия — $X > 0$.

Задача 9*. Укажите значения переменных K, L, M, N, при которых логическое выражение $(\neg K \vee M) \rightarrow (\neg L \vee M \vee N)$ **ложно**. Ответ запишите в виде строки из четырех символов: значений переменных K, L, M и N (в указанном порядке). Так, например, строка 1101 соответствует тому, что K = 1, L = 1, M = 0, N = 1.

Решение

Способ 1.

Операция следования (→) при вычислении заданного выражения должна выполняться последней. Эта операция даёт результат 0 (требуемое нами «ложно») только когда из 1 следует 0. Поэтому необходимо, чтобы одновременно выполнялись условия:
$\neg K \vee M = 1$ и $\neg L \vee M \vee N = 0$.

После этого лучше всего рассмотреть второе из этих условий, поскольку операция ИЛИ даёт в результате нуль только в одном-единственном случае: когда все три «составляющие»

логического выражения равны нулю. Поэтому требуемые значения переменных равны: L = 1, M = 0, N = 0.

Из первого условия, если M = 0, то для его выполнения требуется, чтобы K было равно 0.

Остаётся только записать значения переменных в требуемом порядке — K, L, M, N.

Способ 2.

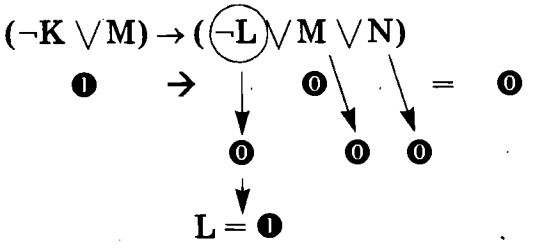
Возможно, для кого-то из учащихся более наглядными окажутся не отвлеченные рассуждения, а более наглядное «графическое» решение.

1. Операция следования даёт результат «ложно» в единственном случае: 1→ 0.

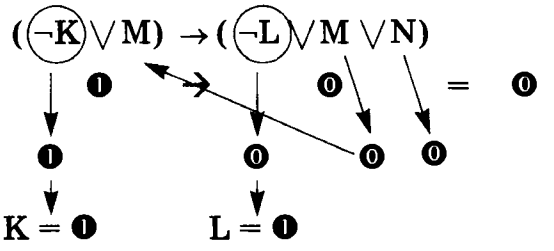
$$(\neg K \vee M) \to (\neg L \vee M \vee N)$$

$$1 \quad \to \quad 0 \quad = \quad 0$$

2. Стоящее справа логическое выражение с использованием операций ИЛИ ложно тоже только в одном случае — когда все его аргументы равны 0:



3. Таким образом определилось единственное возможное значение переменной M = 0. Подставив его в левое логическое выражение для получения в нём результата «истина», остаётся единственный возможный вариант значения ¬K = 1:



4. Остаётся раскрыть операции НЕ (т. е. инвертировать полученные для них значения 1 / 0), а затем записать полученные значения переменных в требуемом порядке: K = 0, L = 1, M = 0, N = 0.

Ответ: 0100.

Задача 10*. Укажите значения логических переменных K, L, M, N, при которых логическое выражение (K ∨ M) → (M ∨ ¬L ∨ N) **ложно.**

Ответ запишите в виде строки из четырёх символов: значений переменных K, L, M и N (в указанном порядке). Так, например, строка 0101 соответствует тому, что K = 0, L = 1, M = 0, N = 1.

Решение

Снова решение начинается с операции следования (→), которая при вычислении выражения должна выполняться последней. Вспомнив, что эта операция даёт результат 0 только когда из 1 следует 0, получается, что для этого одновременно выполнялись условия: K ∨ M = 1 и M ∨ ¬L ∨ N = 0.

Теперь рассматривается второе из этих условий, которое предполагает единственный набор значений переменных: M = 0, L = 1, N = 0.

Вернувшись к первому условию, нетрудно определить, что для его выполнения (при M = 0), требуется, чтобы K было равно 1.

Остаётся записать значения переменных в требуемом порядке — K, L, M, N.

Ответ: 1100.

Задачи для самостоятельного решения

1. Дан фрагмент таблицы истинности выражения D:

A	B	C	D
1	0	0	0
0	1	1	1
1	0	1	0

Каким выражением может быть D?

1) $(A \vee B) \to \neg C$
2) $A \wedge B \wedge \neg C$
3) $\neg A \vee B \vee C$
4) $\neg A \wedge B \wedge C$

2. Дан фрагмент таблицы истинности выражения F:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	F
0	1	0	1	1	1	1	1
1	0	1	0	1	1	0	0
0	1	0	1	1	0	1	1

Каким выражением может быть F?[13]

1) $x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4 \wedge x_5 \wedge x_6 \wedge \neg x_7$
2) $\neg x_1 \vee x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5 \vee \neg x_6 \vee x_7$
3) $\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4 \wedge x_5 \wedge x_6 \wedge x_7$
4) $x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4 \vee \neg x_5 \vee \neg x_6 \vee \neg x_7$

3. Для какого из приведённых чисел Z логическое условие истинно;
$(Z < 5) \to (Z < 3) \vee ((Z < 2) \to (Z > 1))$
1) 1 2) 2 3) 3 4) 4

4. Дан фрагмент таблицы истинности выражения F:

a_1	a_2	a_3	a_4	a_5	a_6	a_7	F
0	1	0	1	1	1	0	0
1	0	1	1	0	0	1	0
0	1	0	1	1	0	1	0

Каким выражением может быть F?

1) $a_1 \to (a_2 \wedge a_3 \vee a_4 \wedge a_5 \vee a_6 \wedge a_7)$
2) $a_2 \to (a_1 \wedge a_3 \vee a_4 \wedge a_5 \vee a_6 \wedge a_7)$
3) $a_3 \to (a_1 \wedge a_2 \vee a_4 \wedge a_5 \vee a_6 \wedge a_7)$
4) $a_4 \to (a_1 \wedge a_2 \vee a_3 \wedge a_5 \vee a_6 \wedge a_7)$

5. Какое из приведённых слов удовлетворяет логическому условию: (первая буква согласная → вторая буква согласная) ∧ (последняя буква гласная → предпоследняя буква гласная)? Если таких слов несколько, укажите самое длинное из них.

1) ИГРА
2) МАФИЯ
3) ОЗОН
4) ТРЕНАЖ

6. Какое из приведённых чисел Z удовлетворяет логическому условию
(Z кратно 4 $\vee$ Z кратно 6) $\rightarrow$ Z кратно 5.
1) 12 2) 7 3) 6 4) 4
7. Ниже приведены имена и фамилии четырёх участников соревнований. Укажите участника, чьи имя и фамилия НЕ удовлетворяют такому условию:
(первая буква имени согласная $\rightarrow$ последняя буква имени согласная) $\wedge$ (последняя буква фамилии согласная $\rightarrow$ первая буква фамилии согласная).
Если таких участников несколько, укажите того из них, у которого самая длинная фамилия.
1) АННА АННЕНКОВА 2) МАРИЯ МИХАЙЛОВА
3) ОЛЕГ ОРЛОВ 4) СТЕПАН САРГСЯН
8. Каково наибольшее целое число Z, при котором ложно высказывание?
$(8Z - 6 < 75) \rightarrow (Z \cdot (Z - 1) > 65)$
9. Сколько различных решений имеет уравнение
$\neg D \wedge B \wedge \neg E \wedge \neg A \wedge (C \vee \neg C) = 0$,
где A, B, C, D, E — логические переменные?
В ответе нужно указать только количество наборов значений A, B, C, D и E , при которых выполнено равенство.
10. Дано логическое выражение:
$(A \rightarrow \neg C) \vee (\neg B \wedge C \wedge A) \vee \neg D$.
Укажите значения переменных A, B, C, D, при которых логическое выражение **ложно.**
Ответ запишите в виде строки из четырёх двоичных цифр: значений переменных A, B, C и D (именно в таком порядке порядке). Например, для значений A = 0, B = 1, C = 0, D = 1 ответ должен иметь вид 0101.

Ответы для самопроверки

№ задания	Ответ
1	4
2	2
3	2
4	4
5	4
6	2
7	2
8	8
9	30
10	1111

Основы логики

B15 Решение логических уравнений. Решение систем логических уравнений

Конспект

В алгебре логики изучаются логические операции, производимые над высказываниями. Высказывания могут быть истинными или ложными.

Применяя к простым высказываниям логические операции, можно строить составные высказывания.

Основные логические операции

Отрицание (инверсия, логическое НЕ)
Результат меняется на противоположный (вместо истины — ложь, вместо лжи — истина).
Обозначение: ¬
Таблица истинности:

A	¬A
0	1
1	0

Логическое сложение (дизъюнкция, логическое ИЛИ)
Результат — истина, если хотя бы один операнд имеет значение «истина».
Обозначения: $\vee$ или +
Таблица истинности:

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

Логическое умножение (конъюнкция, логическое И)
Результат — истина, если оба операнда имеют значение «истина».
Обозначения: $\wedge$ или &
Таблица истинности:

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

Следование (импликация)
Из лжи может следовать что угодно, а из истины — только истина.
Обозначение: →
Таблица истинности:

A	B	A → B
0	0	1
0	1	1
1	0	0
1	1	1

Равносильность (эквиваленция)
Результат — истина, если операнды одинаковы.
Обозначения: ≡ или ↔
Таблица истинности:

A	B	A ↔ B
0	0	1
0	1	0
1	0	0
1	1	1

Если в логическом выражении используется несколько логических операций, то их порядок определяется *приоритетами логических операций:*

Равносильные преобразования

Операцию «импликация» можно выразить через «ИЛИ» и «НЕ»:
$A \to B = \neg A \vee B$
Операцию «эквиваленция» также можно выразить через «ИЛИ» и «НЕ»:
$A \to B = \neg A \wedge \neg B \vee A \wedge B$

Основные законы алгебры логики

Название закона	Формулировка
Переместительный закон	$A \vee B = B \vee A$ $A \wedge B = B \wedge A$
Сочетательный закон	$(A \vee B) \vee C = A \vee (B \vee C)$ $(A \wedge B) \wedge C = A \wedge (B \wedge C)$

Окончание табл.

Название закона	Формулировка
Распределительный закон	$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$
Закон непротиворечия (высказывание не может быть одновременно истинным и ложным)	$A \wedge \neg A = 0$
Закон исключенного третьего (либо высказывание, либо его отрицание должно быть истинным)	$A \vee \neg A = 1$
Закон двойного отрицания	$\neg(\neg A) = A$
Законы де Моргана	$\neg(A \vee B) = \neg A \wedge \neg B$ $\neg(A \wedge B) = \neg A \vee \neg B$
Законы конъюнкции и дизъюнкции переменной с самой собой	$A \vee A = A$ $A \wedge A = A$
Законы конъюнкции и дизъюнкции с нулём и единицей	$A \wedge 0 = 0$ $A \wedge 1 = A$ $A \vee 0 = A$ $A \vee 1 = 1$
Законы поглощения	$A \vee (A \wedge B) = A$ $A \wedge (A \vee B) = A$ $A \vee (\neg A \wedge B) = A \vee B$

Разбор типовых задач

Задача 1*. Сколько различных решений имеет уравнение

$(K \wedge L \wedge M) \vee (\neg L \wedge \neg M \wedge N) = 1,$

где K, L, M, N — логические переменные?

В ответе **не нужно** перечислять все различные наборы значений K, L, M и N, при которых выполнено данное равенство. В качестве ответа вам нужно указать только количество таких наборов.

Решение

Способ 1.

Когда результат логической операции ИЛИ равен 1? Очевидно, для этого достаточно, чтобы хотя бы один из операндов был равен 1. Следовательно, исходное логическое уравнение «распадается» на два следующих уравнения, из которых достаточно, чтобы выполнялось хотя бы одно любое:

$K \wedge L \wedge M = 1; \quad \neg L \wedge \neg M \wedge \rightarrow N = 1.$

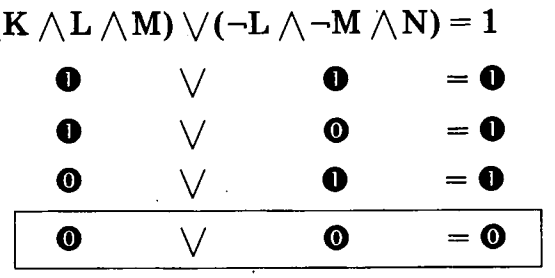
В первом уравнении результат логической операции И равен 1 только в одном-единственном случае — если все операнды равны 1. Значит, в этом случае $K = L = M = 1$. Но тогда второе уравнение не выполняется независимо от значения N, которое может быть любым ($N = 0$ или $N = 1$). Получаются 2 возможных решения исходного уравнения (1110 и 1111).

Аналогично, если требовать выполнения второго уравнения, то это равенство справедливо тоже только в одном случае — если $L = M = 0$ и $N = 1$. Но в этом случае уже не выполняется первое уравнение, причём независимо от значения переменной K. Значит, получаются ещё 2 возможных решения исходного уравнения (0001 и 1001).

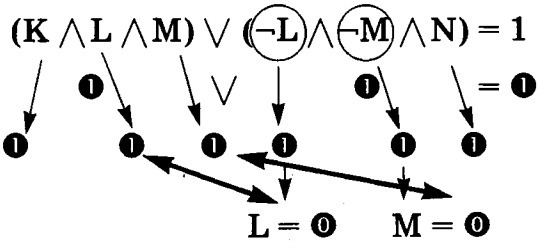
Следовательно, общее количество возможных (различных!) решений равно 4.

Способ 2 (наглядно-«графический»).

1. Результат выполнения операции ИЛИ является ложным в единственном случае — когда оба операнда равны нулю, и истинным — в трёх остальных случаях:

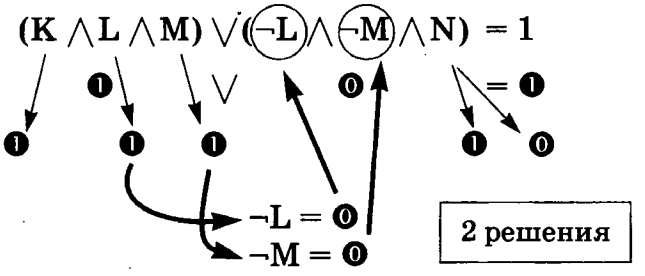


2. Первый возможный вариант:



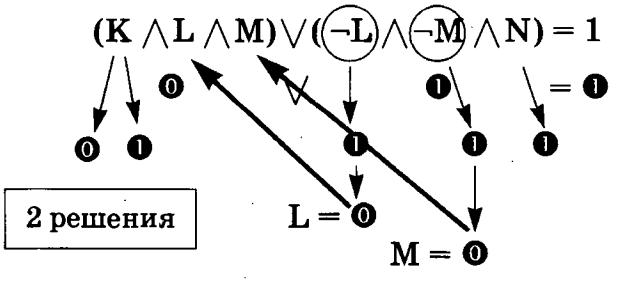
Полученные решения для левого и правого выражений противоречат друг другу. Следовательно, для данного варианта решений не существует.

3. Второй возможный вариант (начиная с левого выражения, так как операция И даёт результат 1 в единственном случае — когда все входящие в него переменные имеют значения 1):



Из левого выражения можно получить единственно возможные значения переменных L и M (а также K) и, инвертировав их (поскольку речь идёт об операции НЕ), подставить в правое выражение. В этом случае значение переменной N, очевидно, может быть любым, так как от него ничего не зависит: нулевого значения любого из двух других аргументов уже достаточно, чтобы обеспечить нулевое значение правого выражения. Получаются два решения (хотя в задаче их не требуется записывать в качестве ответа, почему бы не записать их «для себя»): (K, L, M, N) = (1111) и (1110).

4. Третий возможный вариант (начиная с правого выражения — того, где возможен единственный вариант решения):



Теперь уже из правого выражения можно получить единственно возможные значения переменных L и M (а также N) и, инвертировав их (раскрыв операцию НЕ), подставить в левое выражение. В этом случае значение переменной K также может быть любым, так как от него ничего не зависит: нулевого значения любого из двух других аргументов достаточно, чтобы обеспечить нулевое значение левого выражения. Следовательно, опять получаются два решения: (K, L, M, N) = (0001) и (1001).

5. Таким образом, в трёх рассмотренных вариантах получаются следующие результаты:

- в первом варианте решений нет;
- во втором варианте два решения;
- в третьем варианте ещё два решения (причём, что важно, не совпадающие с вторым вариантом!).

Следовательно, общее количество решений в данной задаче равно 4.

Ответ: 4.

Задача 2*. Сколько различных решений имеет уравнение $((K \vee L) \rightarrow (L \wedge M \wedge N)) = 0$, где K, L, M, N — логические переменные?

В ответе **не нужно** перечислять все различные наборы значений K, L, M и N, при которых выполнено данное равенство. В качестве ответа Вам нужно указать количество таких наборов.

Решение

Известно, что результат выполнения логической операции следования равен нулю только в одном-единственном случае: когда из 1 следует 0. Значит, исходное уравнение превращается в систему двух уравнений:

$$\begin{cases} K \vee L = 1; \\ L \wedge M \wedge N = 0. \end{cases}$$

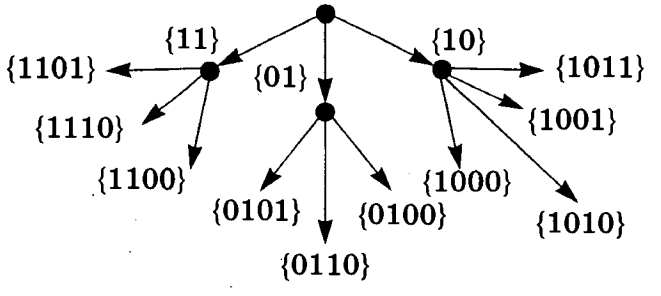
Логическая операция ИЛИ (в первом уравнении) даёт 1 в трёх случаях: {K = L = 1}, {K = 0, L = 1} и {K = 1, L = 0}.

Возможные решения второго уравнения во всех этих случаях:

1) поскольку переменная K во второе уравнение не входит, то в первых двух случаях ({K = L = 1} и {K = 0, L = 1}); второе уравнение выполняется в трёх случаях: {M = 0, N = 1}, {M = 1, N = 0} и {M = N = 0};
2) в третьем случае ({K = 1, L = 0}) второе уравнение выполняется всегда, при любых значениях переменных M и N, а таких вариантов может быть четыре: {M = N = 0}, {M = N = 1}, {M = 1, N = 0} и {M = 0, N = 1}.

Сколько всего тогда возможно вариантов сочетаний значений переменных?

В первом случае — три, плюс во втором случае — три, плюс в третьем случае — четыре. Всего — десять. А для большей наглядности можно построить «дерево вариантов»:



Ответ: 10.

Задача 3*. Сколько различных решений имеет уравнение

$J \wedge \neg K \wedge L \wedge \neg M \wedge (N \vee \neg N) = 0$, где J, K, L, M, N — логические переменные?

В ответе **не нужно** перечислять все различные наборы значений J, K, L, M и N, при которых выполнено данное равенство. В качестве ответа Вам нужно указать количество таких наборов.

Решение

Данная задача выглядит сложной. Однако, решается она ненамного сложнее, чем предыдущие.

Во-первых, комбинация $(N \vee \neg N)$ **всегда** равна 1, а это означает, что переменная N принципиально может быть любой и никак не определяет выполнение уравнения. Поэтому переменную N можно пока временно исключить из рассмотрения.

Во-вторых, операция И даёт в результате 0, если хотя бы один операнд равен 0.

В-третьих, «активных» переменных-операндов (значения которых могут влиять на результат выполнения логического выражения) четыре.

Сколько неповторяющихся комбинаций можно составить из n элементов, каждый из которых может иметь значение 0 или 1? Очевидно[1], их будет 2^n. Соответственно, четыре переменные могут дать $2^4 = 16$ возможных комбинаций. Для решения годятся все, кроме одной — когда все эти операнды равны 1. Значит, пригодных комбинаций будет 15.

Переменная N может принимать любое из двух значений (0 или 1), не влияя на результат вычисления логического выражения. Следовательно, с учётом N, возможно $2 \cdot 15 = 30$ возможных комбинаций пяти «участвующих» в уравнении логических переменных, при которых приведённое в условии логическое уравнение верно.

Ответ: 30.

Задача 4*. Сколько различных решений имеет уравнение

$((J \to K) \to (M \wedge N \wedge L)) \wedge ((J \wedge \neg K) \to \neg(M \wedge N \wedge L)) \wedge (M \to J) = 1$,

где J, K, L, M, N — лгические переменные?

В ответе **не нужно** перечислять все различные наборы значений J, K, L, M и N, при которых выполнено данное равенство. В качестве ответа нужно указать количество таких наборов.

Решение

Так же как и раньше, «расплетается» заданное логическое уравнение.

Самая последняя по порядку выполнения логическая операция здесь — операция И. Её результат равен 1 только в одном-единственном случае — когда все операнды равны 1. Следовательно, взамен исходного уравнения получается система уравнений:

$$\begin{cases} (J \to K) \to (M \wedge N \wedge L) = 1 \\ (J \wedge \neg K) \to \neg(M \wedge N \wedge L) = 1; \\ M \to J = 1. \end{cases}$$

Из третьего уровнения можно определить однозначно, что значение переменной J обязательно должно быть равно 1, а вот значение M может быть любым (2 возможных варианта).

Операция следования даёт 0 в случае, когда из 1 следует 0, и 1 — во всех остальных случаях. Во втором уравнении тот же самый компонент, что и в первом уравнении, взят с отрицанием.

---

[1] Ответить на этот вопрос очень просто, если вспомнить, что такими неповторяющимися комбинациями являются значения битов двоичного числа из соответствующего количества разрядов (в данном случае 4) при полном переборе числовых значений от 0 до максимально возможного значения (т. е. в нашем случае — от 0000 до 1111). Очевидно, что количество всех возможных неповторяющихся комбинаций значений четырёх двоичных переменных равно $1111_2 + 1 = 16$ (где добавляемая ещё одна комбинация — та самая из одних нулей).

Теперь аккуратно рассматривается каждый из возможных вариантов.

1. Пусть верно первое уравнение: $(J \to K) \to (M \wedge N \wedge L) = 1$. Тогда обязательное условие его выполнения — $M \wedge N \wedge L = 1$. Это возможно только в одном случае — когда все три эти переменные равны 1. А вот значение $J \to K$ может быть любым: 0 или 1. Однако, если $M \wedge N \wedge L = 1$, то второе уравнение будет истинным только в одном-единственном случае: когда $J \wedge \neg K = 0$: ведь $\neg(M \wedge N \wedge L)$ тогда равно 0, а операция следования даёт единицу при нулевом втором операнде, только если нулю равен и первый операнд. Причём J может быть равно только 1. Значит, чтобы $J \wedge \neg K$ было равно 0, нужно, чтобы $\neg K$ было равно 0, т. е. чтобы K было равно 1. Итого по результатам анализа этой ситуации получается, что в данном случае возможен только один вариант значений переменных: $J = K = M = N = L = 1$.

2. Пусть теперь верно второе уравнение: $(J \wedge \neg K) \to \neg(M \wedge N \wedge L) = 1$. Тогда для его выполнения нужно, чтобы $\neg(M \wedge N \wedge L)$ было равно 1, т. е. $M \wedge N \wedge L = 0$. Это возможно, если хотя бы одна переменная имеет значение 0. Переменных три, следовательно, всех возможных комбинаций их значений будет $2^3 = 8$. Из них одна комбинация (когда все три переменных равны 1) не подходит, следовательно, пригодных вариантов остаётся 7. Однако, если $M \wedge N \wedge$ $\wedge L = 0$, то первое уравнение системы выполняется только в одном случае: при $J \to K = 0$. А это возможно только в одном случае: когда $J = 1$, а $K = 0$. В этом случае возможно 7 вариантов значений переменных, при которых исходное уравнение верно.

Итого для обоих рассмотренных случаев возможных вариантов значений переменных будет $7 + 1 = 8$.

Ответ: 8.

Подобные логические рассуждения могут быть по силам не всем учащимся, да и хорошо знакомые с логикой школьники могут запутаться и допустить ошибку. Поэтому для таких задач на ЕГЭ лучше придерживаться следующего правила: если у вас остаётся достаточно времени, то лучше решать такую задачу традиционным способом — составляя полную таблицу истинности и подсчитывая в ней количество «правильных» строк. А если время в дефиците, то можно попытаться сократить себе «рутинную» работу, поискав требуемую цепь логических рассуждений.

При подготовке к экзамену для получения правильного ответа для проверки своего решения можно составить компьютерную программу, содержащую некоторое количество (по числу используемых в логическом уравнении переменных) вложенных циклов, в которых соответствующая цикловая переменная меняется от 0 до 1. Например[1]:

```
program z2005_b2;
  var k, l, m, n: boolean;
      x: byte;
  begin
    x := 0;
    for k := false to true do
      for l := false to true do
        for m := false to true do
          for n := false to true do
          if (k and l and m) or (not(l) and not(m) and n) then x := x + 1;
    writeln ('кол-во решений :',x);
  end.
```

[1] На Паскале вместо значений 0 и 1 нужно использовать логические (тип boolean) значения true и false, однако, в операторе цикла for эти значения можно использовать так же, как константы 1 и 0 соответственно, так как логический тип тоже является перечислимым. Кроме того, поскольку нужно проверять равенство заданного логического выражения значению 1 (т.е. true), в условном операторе нет необходимости записывать операцию сравнения: ветвь then выполняется только в этом случае.

Задача 5. Определить, сколько различных решений имеет система уравнений:

$$\begin{cases} (x_1 \equiv x_2) \vee (x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) = 1; \\ (x_1 \equiv x_3) \vee (x_3 \wedge x_4) \vee (\neg x_3 \wedge \neg x_4) = 1; \\ (x_1 \equiv x_4) \vee (x_4 \wedge x_5) \vee (\neg x_4 \wedge \neg x_5) = 1; \\ x_1 \equiv x_5 = 1. \end{cases}$$

Нужно указать именно количество решений системы уравнения, а не записывать сами эти решения.

Решение

В задаче используется достаточно «редкая» для школьного курса булевой логики операция «тождество» ($\equiv$). Её результат является истинным тогда и только тогда, когда обе связанные ею логические переменные равны между собой (обе равны 0 либо обе равны 1).

Из последнего уравнения системы следует, что переменные x_1 и x_5 должны быть равны друг другу.

Далее нужно проанализировать остальные уравнения (эта задача несколько облегчается тем, что все они — «типовые»).

1) Уравнение $(x_1 \equiv x_2) \vee (x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) = 1$.

Результат логической операции ИЛИ ($\vee$) равен 1, если хотя бы один операнд равен 1. Значит, достаточно, чтобы $x_1 \equiv x_2 = 1$, или $x_2 \wedge x_3 = 1$, или $\neg x_2 \to \neg x_3 = 1$.

Из первой части уравнения следует, что x_1 и x_2 либо обе равны 0, либо обе равны 1.

Из второй части уравнения следует, что обе переменные — x_2 и x_3 — должны быть равны 1.

Из третьей части уравнения следует, что обе переменные — x_2 и x_3 — должны быть равны 0.

Тогда уравнение (1) даёт следующие возможные комбинации значений переменных (x_1, x_2, x_3):

- (000) или (001) или (110) или (111) — если истинна операция тождества, то результат остальных операций безразличен, т. е. x_3 может быть любой;
- (011) или (111) — если истинна вторая часть уравнения, то третья часть ложна (но это нам не важно), а результат операции тождества нам также безразличен, и значение x_1 может быть любым;
- (000) или (100) — если истинна третья часть уравнения, то вторая часть ложна (и это нам не важно), а результат операции тождества также безразличен, и значение x_1 может быть любым.

По уравнению (1) получаются **шесть** различных вариантов значений переменных (x_1, x_2, x_3): (000), (001), (011), (110), (100) и (111).

2) Уравнение $(x_1 \equiv x_3) \vee (x_3 \wedge x_4) \vee (\neg x_3 \wedge \neg x_4) = 1$.

Результат логической операции ИЛИ ($\vee$) равен 1, если хотя бы один операнд равен 1. Значит, достаточно, чтобы $x_1 \equiv x = 1$ или $x_3 \wedge x_4 = 1$ или $\neg x_3 \to \neg x_4 = 1$.

Рассуждения для него аналогичны предыдущим.

Из первой части уравнения следует, что x_1 и x_3 либо обе равны 0, либо обе равны 1.

Из второй части уравнения следует, что обе переменные — x_3 и x_4 — должны быть равны 1.

Из третьей части уравнения следует, что обе переменные — x_3 и x_4 — должны быть равны 0.

Тогда уравнение (2) даёт следующие возможные комбинации значений переменных (x_1, x_3, x_4):

- (000) или (001) или (110) или (111);

• (011) или (111);
• (000) или (100).

Итого по уравнению (2) получаются тоже **шесть** различных вариантов значений переменных (x_1, x_3, x_4): (000), (001), (011), (110), (100) и (111).

3) Уравнение $(x_1 \equiv x_4) \vee (x_4 \wedge x_5) \vee (\neg x_4 \wedge \neg x_5) = 1$.

Уравнение (3) даёт следующие **шесть** возможных различных вариантов значений переменных (x_1, x_4, x_5): (000), (001), (011), (110), (100) и (111).

Поскольку все уравнения системы, кроме последнего, — типовые, то тройки значений соответствующих каждому уравнению переменных всегда будут одни и те же, сколько бы ни было таких уравнений.

4) Для последнего уравнения возможны две комбинации значений переменных (x_1, x_5): (00) и (11).

Теперь нужно «свести» результаты, полученные для каждого уравнения в отдельности, воедино, вспомнив, что объединение уравнений в систему означает, что **все** эти уравнения должны выполняться одновременно. Тогда:

- учитывая результаты для последнего уравнения, надо в предпоследнем уравнении оставить только те решения, в которых одинаковы значения x_1 и x_5 (первое и последнее в каждой триаде): (000) и (111);
- сочетание последнего и предпоследнего уравнений выполняется только тогда, когда все три переменные — x_1, x_4 и x_5 — имеют одинаковые значения;
- в найденных наборах значений переменных для пред-предпоследнего уравнения (оно числится под номером 2) следует оставить только те варианты, в которых одинаковы значения x_1 и x_4 (первое и последнее в каждой триаде): (000), (111);
- три переменные — x_1, x_3 и x_4 — тоже должны иметь одинаковые значения;
- тогда в найденных наборах значений переменных для самого первого уравнения тоже надо оставить только варианты, в которых одинаковы значения x_1 и x_3 (первое и последнее в каждой триаде): (000), (111);
- три переменные — x_1, x_2 и x_3 — также должны иметь одинаковые значения.

Таким образом, начиная с последнего уравнения, которое однозначно задаёт два возможных варианта комбинаций значений соответствующих переменных, «расплетается» система уравнений «снизу вверх». На каждом шаге подтверждается, что все три задействованные в текущем анализируемом уравнении переменные должны быть равными, и тем самым однозначно определяются только два возможных варианта решений в уравнении, предыдущем текущему. Такой «цикл типовых рассуждений» повторяется многократно. На каждом его «шаге» (до самого первого уравнения) выявляется ещё одна переменная и количество уравнений в системе на рассуждения практически не влияет — лишь увеличивается количество «проходов цикла».

Все вышесказанное означает, что, сколько бы ни было таких «типовых» уравнений в системе, рано или поздно следует вывод: в решениях этой системы **все** её переменные должны быть тождественны. А значит, эта система имеет только два возможных решения: (00000) и (11111).

Ответ: 2 решения.

Задача 6. Сколько различных решений имеет система уравнений

$$\begin{cases} \neg(x_1 \equiv x_2) \wedge \neg(x_2 \equiv x_3) = 1, \\ \neg(x_2 \equiv x_3) \wedge \neg(x_3 \equiv x_4) = 1, \\ \ldots \\ \neg(x_8 \equiv x_9) \wedge \neg(x_9 \equiv x_{10}) = 1, \end{cases}$$

где x_1, x_2, ..., x_{10} — логические переменные?

В ответе не нужно перечислять все различные наборы значений x_1, x_2, ..., x_{10}, при которых выполнена данная система равенств. В качестве ответа вам нужно указать количество таких наборов.

Решение

1) Анализируется первое уравнение этой системы:

$\neg(x_1 \equiv x_2) \wedge \neg(x_2 \equiv x_3) = 1$.

Символ $\rightarrow$ обозначает логическую операцию тождества:

A	B	A ≡ B
0	0	1
0	1	0
1	0	0
1	1	1

То есть операция тождества истинна, если оба операнда одинаковы (оба равны 1 или оба равны 0), и ложна, если они различны.

Операция, выполняемая в этом уравнении последней, — **И**. Следовательно, чтобы уравнение было верным, нужно, чтобы были истинными обе его составляющие:

$\neg(x_1 \equiv x_2) = 1$ и $\neg(x_2 \equiv x_3) = 1$,

либо, учитывая операции отрицания:

$(x_1 \equiv x_2) = 0$ и $(x_2 \equiv x_3) = 0$,

Какие значения переменных x_1, x_2, x_3 могут этому соответствовать? Если $x_1 = 0$, то x_2 должно быть равно 1 (только тогда тождество x_1 и x_2 ложно). Но тогда x_3 должно равняться 0. И наоборот, если $x_1 = 1$, то $x_2 = 0$, а $x_3 = 1$. Другими словами, x_1 должно быть тождественно равно x_3. Следовательно, первое уравнение даёт нам 2 варианта задействованных в нём переменных: (0, 1, 0) и (1, 0, 1).

2) Учитывая эти два единственно допустимых варианта значений переменных x_1, x_2, x_3, можно приступить к анализу второго уравнения системы:

$\neg(x_2 \equiv x_3) \wedge \neg(x_3 \equiv x_4) = 1$

Рассуждая аналогично предыдущему шагу решения:

$(x_2 \equiv x_3) = 0$ и $(x_3 \equiv x_4) = 0$, откуда $x_2 \rightarrow x_4$.

При этом, возможные значения x_2 и x_3 жёстко заданы (2 варианта). Тогда с учётом данного уравнения по-прежнему будет оставаться 2 возможных варианта значений переменных: (0, 1, 0, 1) и (1, 0, 1, 0).

3) Поскольку все уравнения системы точно такие же, эта тенденция будет сохраняться для каждого из них: предыдущие уравнения жёстко определяют два возможных варианта значений переменных, и текущее уравнение, добавляя к ним «свою» переменную, никак не меняет количество этих вариантов. Так — вплоть до последнего уравнения.

Данная система уравнений в целом имеет только два варианта решений, причём нетрудно указать и сами эти варианты, хотя в условии это не требуется:

(0, 1, 0, 1, 0, 1, 0, 1, 0, 1) и (1, 0, 1, 0, 1, 0, 1, 0, 1, 0).

Ответ: 2 варианта решений (наборов переменных).

Задача 7*. Сколько различных решений имеет система уравнений

$$\begin{cases} ((x_1 \equiv x_2) \vee (x_3 \equiv x_4)) \wedge (\neg(x_1 \equiv x_2) \vee \neg(x_3 \equiv x_4)) = 1, \\ ((x_3 \equiv x_4) \vee (x_5 \equiv x_6)) \wedge (\neg(x_3 \equiv x_4) \vee \neg(x_5 \equiv x_6)) = 1, \\ \ldots \\ ((x_7 \equiv x_8) \vee (x_9 \equiv x_{10})) \wedge (\neg(x_7 \equiv x_8) \vee \neg(x_9 \equiv x_{10})) = 1, \end{cases}$$

где x_1, x_2, ..., x_{10} — логические переменные?

В ответе не нужно перечислять все различные наборы значений x_1, x_2, ..., x_{10}, при которых выполнена данная система равенств. В качестве ответа вам нужно указать количество таких наборов.

Решение

Такая система уравнений выглядит пугающе, но вполне решаема!

1) Анализируя первое уравнение:

$((x_1 \equiv x_2) \vee (x_3 \equiv x_4)) \wedge (\neg(x_1 \equiv x_2) \vee \neg(x_3 \equiv x_4)) = 1.$

Последняя выполняемая операция здесь — **И**, поэтому:

$((x_1 \equiv x_2) \vee (x_3 \equiv x_4)) = 1$ и $(\neg(x_1 \equiv x_2) \vee \neg(x_3 \equiv x_4)) = 1.$

В обеих частях записаны одни и те же тождества, только в первом случае они записаны «как есть», а во втором — с отрицаниями. Тогда, если $(x_1 \equiv x_2) = 1$ и $(x_3 \equiv x_4) = 1$, то первая запись будет истинной, но тогда $\neg(x_1 \equiv x_2)$ и $\neg(x_3 \equiv x_4)$ оба будут ложными, и вторая запись ложна. И наоборот, при $(x_1 \equiv x_2) = 0$ и $(x_3 \equiv x_4) = 0$ первая запись будет ложной, а вторая (с отрицаниями) — истинной. Не подходит ни тот, ни другой вариант. «Спасает положение» то, что тождества в обоих записях соединены операцией **ИЛИ**, т. е. оба раза достаточно, чтобы единице было равно хотя бы одно из этих тождеств.

Вывод: чтобы первое уравнение нашей системы было равно 1, нужно, чтобы либо $(x_1 \equiv x_2) = 1$ и $(x_3 \equiv x_4) = 0$, либо, наоборот, $(x_1 \equiv x_2) = 0$ и $(x_3 \equiv x_4) = 1$.

Первое из этих «либо» даёт такие варианты значений переменных, когда x_1 и x_2 одинаковы, а x_3 и x_4 различны:

(0, 0, 1, 0)
(0, 0, 0, 1)
(1, 1, 1, 0)
(1, 1, 0, 1)

Второе «либо» аналогично даёт варианты, в которых, наоборот, x_1 и x_2 различны, а x_3 и x_4 одинаковы:

(1, 0, 0, 0)
(0, 1, 0, 0)
(1, 0, 1, 1)
(0, 1, 1, 1)

Всего — 8 вариантов.

2) В анализ добавляется второе уравнение:

$((x_3 \equiv x_4) \vee (x_5 \equiv x_6)) \wedge (\neg(x_3 \equiv x_4) \vee \neg(x_5 \equiv x_6)) = 1$

Рассуждая аналогично и учитывая, что для x_3 и x_4 возможные варианты «унаследованы» от предыдущего уравнения, получается, что в вариантах значений x_5, x_6, добавленных этим вторым уравнением, для одинаковых значений x_3 и x_4 должны быть разными значения x_5 и x_6, а для различных значений x_3 и x_4 — одинаковые значения x_5 и x_6:

(1, 0, 0, 0)	→	(1, 0, 0, 0, 1, 0)
		(1, 0, 0, 0, 0, 1)
(0, 1, 0, 0)		(0, 1, 0, 0, 1, 0)
		(0, 1, 0, 0, 0, 1)
(1, 0, 1, 1)		(1, 0, 1, 1, 1, 0)
		(1, 0, 1, 1, 0, 1)
(0, 1, 1, 1)		(0, 1, 1, 1, 1, 0)
		(0, 1, 1, 1, 0, 1)

(0, 0, 1, 0)		(0, 0, 1, 0, 0, 0) (0, 0, 1, 0, 1, 1)
(0, 0, 0, 1)		(0, 0, 0, 1, 0, 0) (0, 0, 0, 1, 1, 1)
(1, 1, 1, 0)		(1, 1, 1, 0, 0, 0) (1, 1, 1, 0, 1, 1)
(1, 1, 0, 1)		(1, 1, 0, 1, 0, 0) (1, 1, 0, 1, 1, 1)

Итого из 8 предыдущих вариантов благодаря второму уравнению получается 16 (вдвое больше).

3) Очевидно, такая тенденция сохранится и дальше, ведь уравнения системы — типовые. Значит, добавление в рассмотрение третьего уравнения, пропущенного в записи системы и использующего переменные x_5, x_6, x_7, x_8, снова удвоит количество вариантов значений переменных: из 16 их получится 32.

Аналогично последнее, четвёртое уравнение системы (переменные x_7, x_8, x_9, x_{10}) снова удвоит количество вариантов, «унаследованное» от предыдущего уравнения. В итоге для всей системы уравнений получается 64 возможных варианта значений переменных x_1 — x_{10}.

Ответ: 64 варианта значений переменных.

Задача 8. Сколько существует различных наборов значений логических переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, которые удовлетворяют всем перечисленным ниже условиям?

$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1$;

$(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1$;

$y_5 \to x_5 = 1$

В ответе не нужно перечислять все различные наборы значений переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, при которых выполнена данная система равенств. В качестве ответа Вам нужно указать количество таких наборов.

Решение

Как всегда при решении задач с системами логических уравнений нужно сначала проанализировать каждое уравнение в отдельности. При этом первое и второе уравнения заданной системы практически идентичны (с точностью до имён переменных — «игреки» вместо «иксов»), и это существенно облегчит работу.

Анализируется первое уравнение:

$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1$;

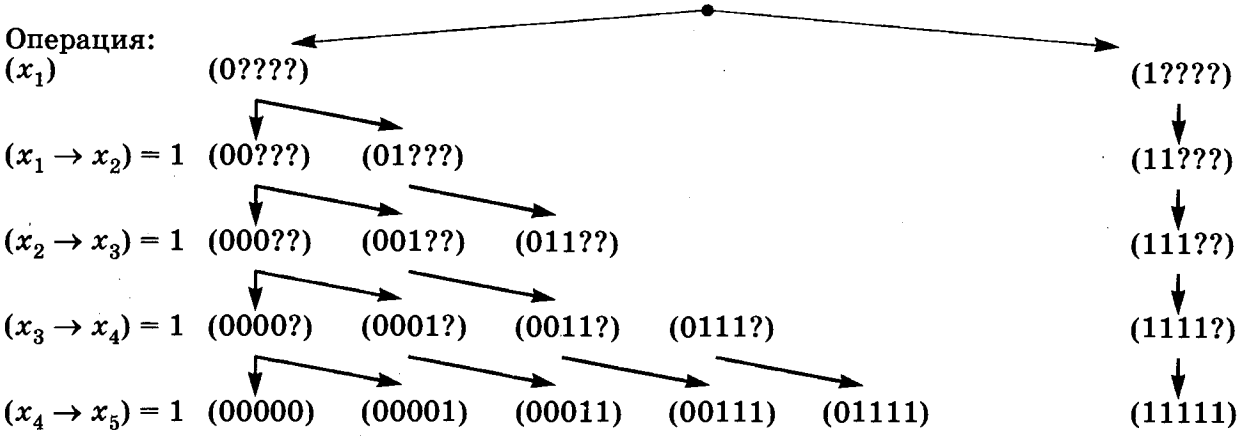
Наполняется таблица истинности логической операции следования: единственная ситуация, при которой её результат равен нулю, — когда из единицы следует нуль, а во всех других случаях эта операция возвращает единицу:

a	b	$a \to b$
0	0	1
0	1	1
1	0	0
1	1	1

Кроме того, поскольку все отдельные операции следования в первом уравнении соединены операцией И, для выполнения заданного в нём равенства требуется, чтобы *все* операции следования давали в результате единицу.

Чтобы найти все возможные комбинации значений переменных, задействованных в первом уравнении, удобнее всего выполнить построение дерева решений: это позволит не запу-

таться и не пропустить какие-то варианты. При построении дерева на каждом его очередном шаге анализируется очередная пара переменных и для каждой имеющейся ветви определяются дальнейшие варианты ветвления. Слева указаны логические операции следования, которые и анализируются на соответствующих шагах (уровнях дерева). Ключевым моментом при построении дерева является уже отмеченный выше факт, что для получения единичного результата из нуля может следовать любое значение второй переменной, а из единицы — *только* единица.



Основную идею при построении данного дерева можно условно выразить фразой: «*размножаются только нули*». То есть, если имеется начало набора значений пяти переменных, которое на данный момент завершается нулём, то продолжить его можно как нулём, так и единицей (в дереве имеется ветвление), но если текущая последовательность заканчивается единицей, то продолжать её можно *только* единицей, и в дереве не будет никакого ветвления, а только продолжение уже существующей ветви.

Полный набор возможных значений переменных, удовлетворяющих первому уравнению, содержится в самой нижней строке построенного дерева (в его «листьях»): $(x_1x_2x_3x_4x_5)$ = = (00000), (00001), (00011), (00111), (01111), (11111).

Второе уравнение, как уже было ранее отмечено, по структуре полностью совпадает с первым. Поэтому анализировать его уже нет необходимости, и можно сразу записать набор возможных для него значений переменных: $(y_1y_2y_3y_4y_5)$ = (00000), (00001), (00011), (00111), (01111), (11111).

Если бы в условии задачи присутствовали только рассмотренные два уравнения, то, поскольку в них нет общих переменных, решением этой системы уравнений были бы *все* возможные попарные сочетания найденных наборов значений «иксов» и «игреков». Именно третье уравнение, в котором одной логической операцией связаны один из «иксов» и один из «игреков», является «*ключом*», определяющим выбор: какие из найденных комбинаций наборов значений $(x_1x_2x_3x_4x_5)$ и $(y_1y_2y_3y_4y_5)$ годятся, а какие — нет.

Запись третьего уравнения: $y_5 \to x_5 = 1$.

Согласно ему, из всех найденных пар наборов значений «иксов» и «игреков» подходят только такие, в которых значения указанных переменных соответствуют истинности заданной логической операции, т. е.:

- когда в наборе значений $(y_1y_2y_3y_4y_5)$ пятая цифра равна нулю, в пару с ним годятся *любые* наборы значений $(x_1x_2x_3x_4x_5)$. Что бы в них ни стояло в пятой позиции (0 или 1), результат операции $y_5 \to x_5 = 1$ будет равен 1 в любом случае (см. таблицу истинности для этой операции);
- когда в наборе значений $(y_1y_2y_3y_4y_5)$ пятая цифра равна единице, в пару с ним годятся *только такие* наборы значений $(x_1x_2x_3x_4x_5)$, в которых пятая цифра равна 1.

Удобнее и нагляднее всего расписать все получаемые комбинации значений x и y в виде таблицы («*матрицы решений*»). Анализируемые цифры в ней подчеркнуты.

«Матрица решений»:

$(y_1y_2y_3y_4y_5)$	$(x_1x_2x_3x_4x_5)$						*Кол-во вариантов (пар)*
	(00000)	(00001)	(00011)	(00111)	(01111)	(11111)	
(00000)	+	+	+	+	+	+	6
(00001)	–	+	+	+	+	+	5
(00011)	–	+	+	+	+	+	5
(00111)	–	+	+	+	+	+	5
(01111)	–	+	+	+	+	+	5
(11111)	–	+	+	+	+	+	5
Всего возможных вариантов (пар) наборов значений $(x_1x_2x_3x_4x_5)$ ***и*** $(y_1y_2y_3y_4y_5)$***:***							**31**

Ответ: заданная система уравнений имеет 31 решение.

В данной задаче система логических уравнений состоит из двух чётко обособленных частей: первые два уравнения представляют собой «генераторы наборов значений переменных», а последнее уравнение — это «селектор», «фильтр», осуществляющий отбор из всех возможных попарных комбинаций наборов «иксов» и «игреков».

Задача 9. Сколько существует различных наборов значений логических переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, которые удовлетворяют всем перечисленным ниже условиям?

$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1$;

$(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1$;

$x_1 \to y_1 = 1$

В ответе не нужно перечислять все различные наборы значений переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, при которых выполнена данная система равенств. В качестве ответа Вам нужно указать количество таких наборов.

Решение

Первая часть решения этой задачи аналогична только что рассмотренной, поскольку первые два уравнения («генераторы») те же самые. Из анализа этих уравнений выделяются два набора значений соответствующих переменных:

$(x_1x_2x_3x_4x_5)$ = (00000), (00001), (00011), (00111), (01111), (11111);

$(y_1y_2y_3y_4y_5)$ = (00000), (00001), (00011), (00111), (01111), (11111).

Третье уравнение требует анализа первых значений в каждом таком наборе.

«Матрица решений»:

$(x_1x_2x_3x_4x_5)$	$(y_1y_2y_3y_4y_5)$						*Кол-во вариантов (пар)*
	(00000)	(00001)	(00011)	(00111)	(01111)	(11111)	
(00000)	+	+	+	+	+	+	6
(00001)	+	+	+	+	+	+	6
(00011)	+	+	+	+	+	+	6
(00111)	+	+	+	+	+	+	6
(01111)	+	+	+	+	+	+	6
(11111)	–	–	–	–	–	+	1
Всего возможных вариантов (пар) наборов значений $(x_1x_2x_3x_4x_5)$ ***и*** $(y_1y_2y_3y_4y_5)$***:***							**31**

Ответ: заданная система уравнений также имеет 31 решение.

Задача 10. Сколько существует различных наборов значений логических переменных $x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5$, которые удовлетворяют всем перечисленным ниже условиям?

$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1;$

$(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1;$

$x_1 \vee y_1 = 1$

В ответе не нужно перечислять все различные наборы значений переменных $x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4$, при которых выполнена данная система равенств. В качестве ответа Вам нужно указать количество таких наборов.

Решение

Аналогично двум предыдущим задачам, из анализа первых двух уравнений имеются два набора значений соответствующих переменных:

$(x_1x_2x_3x_4x_5) = (00000), (00001), (00011), (00111), (01111), (11111);$

$(y_1y_2y_3y_4y_5) = (00000), (00001), (00011), (00111), (01111), (11111).$

Третье уравнение требует анализа первых значений в каждом таком наборе. При этом, поскольку в нём записана операция ИЛИ, правило отбора значений будет следующим:

- когда в наборе значений $(x_1x_2x_3x_4x_5)$ первая цифра равна единице, в пару с ним годятся *любые* наборы значений $(y_1y_2y_3y_4y_5)$;
- когда в наборе значений $(x_1x_2x_3x_4x_5)$ первая цифра равна единице, в пару с ним годятся *только такие* наборы значений $(y_1y_2y_3y_4y_5)$, в которых первая цифра равна 1.

«Матрица решений»:

$(x_1x_2x_3x_4x_5)$	$(y_1y_2y_3y_4y_5)$						*Кол-во вариантов (пар)*
	(00000)	(00001)	(00011)	(00111)	(01111)	(11111)	
(00000)	–	–	–	–	–	+	1
(00001)	–	–	–	–	–	+	1
(00011)	–	–	–	–	–	+	1
(00111)	–	–	–	–	–	+	1
(01111)	–	–	–	–	–	+	1
(11111)	+	+	+	+	+	+	6
Всего возможных вариантов (пар) наборов значений $(x_1x_2x_3x_4x_5)$ ***и*** $(y_1y_2y_3y_4y_5)$***:***							**11**

Ответ: заданная система уравнений имеет 11 решений.

Задача 11. Сколько существует различных наборов значений логических переменных $x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5$, которые удовлетворяют всем перечисленным ниже условиям?

$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1;$

$(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1;$

$x_1 \vee y_1 = 0$

В ответе не нужно перечислять все различные наборы значений переменных $x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5$, при которых выполнена данная система равенств. В качестве ответа Вам нужно указать количество таких наборов.

Решение

Из анализа первых двух уравнений имеются два набора значений соответствующих переменных:

$(x_1x_2x_3x_4x_5) = (00000), (00001), (00011), (00111), (01111), (11111);$

$(y_1y_2y_3y_4y_5) = (00000), (00001), (00011), (00111), (01111), (11111).$

Третье уравнение требует анализа первых значений в каждом таком наборе. При этом, поскольку в нём записана операция ИЛИ, результат которой должен быть равен нулю, правило отбора значений будет следующим:

- если в наборе значений $(x_1x_2x_3x_4x_5)$ первая цифра равна нулю, в пару с ним годятся *только такие* наборы значений $(y_1y_2y_3y_4y_5)$, в которых первая цифра тоже равна нулю;
- если в наборе значений $(x_1x_2x_3x_4x_5)$ первая цифра равна единице, то уже никакие наборы значений $(y_1y_2y_3y_4y_5)$ не обеспечивают истинности третьего равенства, поэтому такие комбинации наборов значений «иксов» и «игреков» необходимо исключить вовсе.

«Матрица решений»:

$(x_1x_2x_3x_4x_5)$	$(y_1y_2y_3y_4y_5)$						*Кол-во вариантов (пар)*
	(00000)	(00001)	(00011)	(00111)	(01111)	(11111)	
(00000)	+	+	+	+	+	–	5
(00001)	+	+	+	+	+	–	5
(00011)	+	+	+	+	+	–	5
(00111)	+	+	+	+	+	–	5
(01111)	+	+	+	+	+	–	5
(11111)	–	–	–	–	–	–	0
Всего возможных вариантов (пар) наборов значений $(x_1x_2x_3x_4x_5)$ ***и*** $(y_1y_2y_3y_4y_5)$***:***							**25**

Ответ: заданная система уравнений имеет 25 решений.

Задача 12. Сколько существует различных наборов значений логических переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, которые удовлетворяют всем перечисленным ниже условиям?

$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1;$

$(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1;$

$x_3 \oplus y_3 = 1$

В ответе не нужно перечислять все различные наборы значений переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, при которых выполнена данная система равенств. В качестве ответа Вам нужно указать количество таких наборов.

Решение

Из анализа первых двух уравнений имеются два набора значений соответствующих переменных:

$(x_1x_2x_3x_4x_5)$ = (00000), (00001), (00011), (00111), (01111), (11111);

$(y_1y_2y_3y_4y_5)$= (00000), (00001), (00011), (00111), (01111), (11111).

Третье уравнение требует анализа третьих по счёту значений в каждом таком наборе. Кроме того, поскольку в нём записана операция ИСКЛЮЧАЮЩЕЕ ИЛИ, результат которой должен быть равен единице, правило отбора значений будет следующим:

- когда в наборе значений $(x_1x_2x_3x_4x_5)$ третья цифра равна нулю, в пару с ним годятся такие наборы значений $(y_1y_2y_3y_4y_5)$, в которых первая цифра равна единице;
- наоборот, когда в наборе значений $(x_1x_2x_3x_4x_5)$ третья цифра равна единице, в пару с ним годятся такие наборы значений $(y_1y_2y_3y_4y_5)$, в которых первая цифра равна нулю.

Или, иными словами, нужны такие пары наборов «иксов» и «игреков», в которых третьи по счёту цифры *различны*.

«Матрица решений»:

$(x_1x_2x_3x_4x_5)$	$(y_1y_2y_3y_4y_5)$						*Кол-во вариантов (пар)*
	(00000)	(00001)	(00011)	(00111)	(01111)	(11111)	
(00000)	–	–	–	+	+	+	3
(00001)	–	–	–	+	+	+	3
(00011)	–	–	–	+	+	+	3
(00111)	+	+	+	–	–	–	3
(01111)	+	+	+	–	–	–	3
(11111)	+	+	+	–	–	–	3
Всего возможных вариантов (пар) наборов значений $(x_1x_2x_3x_4x_5)$ *и* $(y_1y_2y_3y_4y_5)$*:*							18

Ответ: заданная система уравнений имеет 18 решений.

Задача 13. Сколько существует различных наборов значений логических переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, которые удовлетворяют всем перечисленным ниже условиям?

$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1$;
$(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1$;
$(x_1 \to y_1) \vee (x_5 \oplus x_5) = 0$

В ответе не нужно перечислять все различные наборы значений переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, при которых выполнена данная система равенств. В качестве ответа Вам нужно указать количество таких наборов.

Решение

В «ключевом» уравнении задачи присутствуют сразу две логические операции (СЛЕДОВАНИЕ и ИСКЛЮЧАЮЩЕЕ ИЛИ) для двух пар «иксов» и «игреков», связанные третьей операцией — ИЛИ. Однако не стоит пугаться: принципы её решения — те же, что и для ранее рассмотренных.

Из анализа первых двух уравнений имеются два набора значений соответствующих переменных:

$(x_1x_2x_3x_4x_5)$ = (00000), (00001), (00011), (00111), (01111), (11111);
$(y_1y_2y_3y_4y_5)$ = (00000), (00001), (00011), (00111), (01111), (11111).

Третье уравнение требует совместного попарного анализа первых и пятых значений в каждом таком наборе.

Во-первых, операция ИЛИ даёт нулевой результат, если *оба* связываемых ею значения нулевые.

Во-вторых, операция СЛЕДОВАНИЯ даёт нуль только в случае «$1 \to 0$».

В-третьих, операция ИСКЛЮЧАЮЩЕЕ ИЛИ даёт нуль, когда соответствующие значения переменных *одинаковы* (оба нули или оба единицы).

Тогда правило отбора значений будет следующим: годятся только такие пары наборов значений «иксов» и «игреков», в которых первая единица в наборе $(x_1x_2x_3x_4x_5)$ соответствует первому нулю в наборе $(y_1y_2y_3y_4y_5)$ и одновременно пятые цифры одинаковы.

«Матрица решений»:

$(x_1x_2x_3x_4x_5)$	$(y_1y_2y_3y_4y_5)$						*Кол-во вариантов (пар)*
	(00000)	(00001)	(00011)	(00111)	(01111)	(11111)	
(00000)	–	–	–	–	–	–	0
(00001)	–	–	–	–	–	–	0
(00011)	–	–	–	–	–	–	0
(00111)	–	–	–	–	–	–	0
(01111)	–	–	–	–	–	–	0
(11111)	–	+	+	+	+	–	4
Всего возможных вариантов (пар) наборов значений $(x_1x_2x_3x_4x_5)$ *и* $(y_1y_2y_3y_4y_5)$*:*							4

Ответ: заданная система уравнений имеет 4 решения.

Задача 14. Сколько существует различных наборов значений логических переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, которые удовлетворяют всем перечисленным ниже условиям?

$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1$;

$(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1$;

$(x_1 \to y_1) \wedge (x_5 \vee y_5) = 0$

В ответе не нужно перечислять все различные наборы значений переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, при которых выполнена данная система равенств. В качестве ответа Вам нужно указать количество таких наборов.

Решение

В «ключевом» уравнении присутствуют две логические операции (СЛЕДОВАНИЕ и ИЛИ), связанные третьей операцией — И.

Из анализа первых двух уравнений имеются два набора значений соответствующих переменных:

$(x_1x_2x_3x_4x_5)$ = (00000), (00001), (00011), (00111), (01111), (11111);

$(y_1y_2y_3y_4y_5)$ = (00000), (00001), (00011), (00111), (01111), (11111).

Третье уравнение требует совместного попарного анализа первых и пятых значений в каждом таком наборе.

Во-первых, операция следования даёт нуль только в случае «1 → 0», а во всех остальных случаях она равна единице.

Во-вторых, операция ИЛИ даёт результат «единица», если *хотя бы одно* из связываемых ею значений равно единице.

В-третьих, операция И даёт единичный результат только тогда, когда оба связываемых ею значения единичные.

Правило отбора значений будет следующим: годятся все такие пары наборов значений «иксов» и «игреков», в которых последние цифры обоих наборов — не одновременно нулевые, и при этом первая цифра в наборе $(x_1x_2x_3x_4x_5)$ либо нулевая, либо единичная, которой в наборе $(y_1y_2y_3y_4y_5)$ соответствует тоже единица.

«Матрица решений»:

$(x_1x_2x_3x_4x_5)$	$(y_1y_2y_3y_4y_5)$						*Кол-во вариантов (пар)*
	(00000)	(00001)	(00011)	(00111)	(01111)	(11111)	
(00000)	–	+	+	+	+	+	5
(00001)	+	+	+	+	+	+	6
(00011)	+	+	+	+	+	+	6
(00111)	+	+	+	+	+	+	6
(01111)	+	+	+	+	+	+	6
(11111)	–	–	–	–	–	+	1
Всего возможных вариантов (пар) наборов значений $(x_1x_2x_3x_4x_5)$ *и* $(y_1y_2y_3y_4y_5)$*:*							30

Ответ: заданная система уравнений имеет 30 решений.

Задача 15. Сколько существует различных наборов значений логических переменных a_1, a_2, ..., a_9, a_{10}, которые удовлетворяют всем перечисленным ниже условиям?

$(a_1 \wedge \neg a_2) \vee (a_3 \wedge \neg a_4) = 0$
$(a_3 \wedge \neg a_4) \vee (a_5 \wedge \neg a_6) = 0$
$(a_5 \wedge \neg a_6) \vee (a_7 \wedge \neg a_8) = 0$
$(a_7 \wedge \neg a_8) \vee (a_9 \wedge \neg a_{10}) = 0$

В ответе не нужно перечислять все различные наборы значений a_1, a_2, ..., a_9, a_{10}, при которых выполнена данная система равенств. В качестве ответа вам нужно указать количество таких наборов.

Решение

1. Начать анализ следует с последнего уравнения: $(a_7 \wedge \neg a_8) \vee (a_9 \to \neg a_{10}) = 0$.

Это уравнение состоит из двух операндов (скобок), соединённых логической операцией ИЛИ. Результат выполнения операции ИЛИ равен нулю в одном-единственном случае — если оба операнда равны нулю. Тогда:

$$\begin{cases} (a_7 \wedge \neg a_8) = 0 \\ (a_9 \wedge \neg a_{10}) = 0 \end{cases}$$

Анализируются все возможные варианты значений переменных a_7, a_8, a_9, a_{10}, удовлетворяющие этим условиям, и составляется таблица. При этом учитывается, что логическая операция И даёт нулевой результат в трёх случаях — когда хотя бы одна переменная равна нулю или обе переменные равны нулю, и для каждого такого случая для первой пары переменных возможно по три случая для второй пары переменных. Кроме того, нужно учитывать, что переменные a_8 и a_{10} записаны с отрицаниями, поэтому в таблицу записываются противоположные значения.

Таблица 1.1

$(a_7 \wedge \neg a_8)$	$(a_9 \wedge \neg a_{10})$
0	0

Таблица 1.2

a_7	a_8	a_9	a_{10}
0	1 (¬0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (0)
0	0 (¬1)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)
1	1 (0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)

2. Аналогично анализируется предпоследнее уравнение: $(a_5 \wedge \neg a_6) \vee (a_7 \to \neg a_8) = 0$.

Это уравнение также состоит из двух операндов, соединенных логической операцией ИЛИ, которая дает нулевой результат, если оба операнда равны нулю:

$$\begin{cases} (a_5 \wedge \neg a_6) = 0, \\ (a_7 \wedge \neg a_8) = 0. \end{cases}$$

Так же как для последнего уравнения, записываются в таблицу все возможные варианты значений переменных a_5, a_6, a_7, a_8, удовлетворяющие этим условиям. Очевидно, что поскольку структура обоих рассмотренных уравнений одинакова, заполнение таблицы будет точно таким же.

Таблица 2.1

$(a_5 \wedge \neg a_6)$	$(a_7 \wedge \neg a_8)$
0	0

Таблица 2.2

a_5	a_6	a_7	a_8
0	1 (¬0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)
0	0 (¬1)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)
1	1 (0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)

Для всех четырёх уравнений структура и заполнение таблицы возможных значений четырёх используемых в каждом уравнении переменных одна и та же. Смысл решения задачи проявляется во взаимосвязи между этими таблицами, отражающими взаимосвязь между уравнениями.

Переменные a_7 и a_8 используются в обоих рассмотренных уравнениях. Поэтому для каждого варианта (набора значений) переменных a_7 и a_8, приведённого в таблице 2.2, нужно определить количество наборов переменных a_7, a_8, a_9, a_{10}, имеющихся в таблице 1.2.

Таблица 2.2 (дополненная)

a_5	a_6	a_7	a_8	Кол-во вариантов с учётом a_9, a_{10}
0	1 (¬0)	0	1 (¬0)	3 ❶
		0	0 (¬1)	3 ❷
		1	1 (¬0)	3 ❸
0	0 (¬1)	0	1 (¬0)	3 ❶
		0	0 (¬1)	3 ❷
		1	1 (¬0)	3 ❸
1	1 (¬0)	0	1 (¬0)	3 ❶
		0	0 (¬1)	3 ❷
		1	1 (¬0)	3 ❸

Таблица 1.2

a_7	a_8	a_9	a_{10}
0 ❶	1 (¬0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)
0 ❷	0 (¬1)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)
1 ❸	1 (¬0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)

3. Аналогично анализируется второе по счёту уравнение: $(a_3 \wedge \neg a_4) \vee (a_5 \wedge \neg a_6) = 0$.

Таблица 3.1

$(a_3 \wedge \neg a_4)$	$(a_5 \wedge \neg a_6)$
0	0

Таблица 3.2

a_3	a_4	a_5	a_6
0	1 (¬0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)
0	0 (¬1)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)
1	1 (¬0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)

Переменные a_5 и a_6 используются в двух рассмотренных уравнениях (втором и третьем), поэтому для каждого варианта (набора значений) переменных a_5 и a_6 в таблице 3.2 нужно определить количество наборов переменных a_5, a_6, a_7, a_8, имеющихся в таблице 2.2 (с учётом ранее определённых количеств вариантов для a_7, a_8).

Таблица 3.2 (дополненная)

a_3	a_4	a_5	a_6	Кол-во вариантов с учётом a_7, a_8, a_9, a_{10}
0	1 (¬0)	0	1 (¬0)	9 (3+3+3) ❶
		0	0 (¬1)	9 (3+3+3) ❷
		1	1 (¬0)	9 (3+3+3) ❸
0	0 (¬1)	0	1 (¬0)	9 (3+3+3) ❶
		0	0 (¬1)	9 (3+3+3) ❷
		1	1 (¬0)	9 (3+3+3) ❸
1	1 (¬0)	0	1 (¬0)	9 (3+3+3) ❶
		0	0 (¬1)	9 (3+3+3) ❷
		1	1 (¬0)	9 (3+3+3) ❸

Таблица 2.2

a_5	a_6	a_7	a_8	Кол-во вариантов
0 ❶	1 (¬0)	0	1 (¬0)	3
		0	0 (¬1)	3
		1	1 (¬0)	3
0 ❷	0 (¬1)	0	1 (¬0)	3
		0	0 (¬1)	3
		1	1 (¬0)	3
1 ❸	1 (¬0)	0	1 (¬0)	3
		0	0 (¬1)	3
		1	1 (¬0)	3

4. Аналогично анализируется первое уравнение: $(a_1 \wedge \neg a_2) \vee (a_3 \wedge \neg a_4) = 0$.

Таблица 4.1

$(a_1 \wedge \neg a_2)$	$(a_3 \wedge \neg a_4)$
0	0

Таблица 4.2

a_1	a_2	a_3	a_4
0	1 (¬0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)
0	0 (¬1)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)
1	1 (¬0)	0	1 (¬0)
		0	0 (¬1)
		1	1 (¬0)

Переменные a_3 и a_4 используются в двух уравнениях (первом и втором), поэтому для каждого варианта (набора значений) переменных a_3 и a_4 в таблице 4.2 нужно определить количество наборов переменных a_3, a_4, a_5, a_6, имеющихся в таблице 3.2 (с учётом ранее определённых количеств вариантов для a_5, a_6).

Таблица 4.2 (дополненная)

a_1	a_2	a_3	a_4	Кол-во вариантов с учётом a_5, a_6, a_7, a_8, a_9, a_{10}
0	1 (¬0)	0	1 (¬0)	27 (9+9+9) ❶
		0	0 (¬1)	27 (9+9+9) ❷
		1	1 (¬0)	27 (9+9+9) ❸
0	0 (¬1)	0	1 (¬0)	27 (9+9+9) ❶
		0	0 (¬1)	27 (9+9+9) ❷
		1	1 (¬0)	27 (9+9+9) ❸
1	1 (¬0)	0	1 (¬0)	27 (9+9+9) ❶
		0	0 (¬1)	27 (9+9+9) ❷
		1	1 (¬0)	27 (9+9+9) ❸
ИТОГО				$9 \times 27 = 243$ варианта

Таблица 3.2

a_3	a_4	a_5	a_6	Кол-во вариантов
0 ❶	1 (¬0)	0	1 (¬0)	9
		0	0 (¬1)	9
		1	1 (¬0)	9
0 ❷	0 (¬1)	0	1 (¬0)	9
		0	0 (¬1)	9
		1	1 (¬0)	9
1 ❸	1 (¬0)	0	1 (¬0)	9
		0	0 (¬1)	9
		1	1 (¬0)	9

Ответ: 243 варианта.

Задача 16. Сколько существует различных наборов значений логических переменных a_1, a_2, …, a_9, a_{10}, которые удовлетворяют всем перечисленным ниже условиям?

$(a_1 \vee a_2) \wedge (\neg a_3 \vee \neg a_4) = 0$

$(a_3 \vee a_4) \wedge (\neg a_5 \vee \neg a_6) = 0$

$(a_5 \vee a_6) \wedge (\neg a_7 \vee \neg a_8) = 0$

$(a_7 \vee a_8) \wedge (\neg a_9 \vee \neg a_{10}) = 0.$

В ответе не нужно перечислять все различные наборы значений a_1, a_2, …, a_9, a_{10}, при которых выполнена данная система равенств. В качестве ответа вам нужно указать количество таких наборов.

Решение

1. Анализ лучше начать с последнего уравнения: $(a_7 \vee a_8) \wedge (\neg a_9 \wedge \neg a_{10}) = 0$.

Это уравнение состоит из двух операндов (скобок), соединённых логической операцией И. Результат выполнения операции И равен нулю в трёх случаях — если хотя бы один операнд равен нулю, т.е. $(a_7 \vee a_8) = 0$ или $(\neg a_9 \vee \neg a_{10}) = 0$.

Таблица 1

$(a_7 \vee a_8)$	$(\neg a_9 \vee \neg a_{10})$
0	0
0	1
1	0

Анализируются все возможные варианты значений переменных a_7, a_8, a_9, a_{10}, удовлетворяющие этим условиям, и составляется таблица.

При этом учитывается, что логическая операция ИЛИ даёт нулевой результат в одном случае из четырёх (если обе переменные равны нулю) и единичный результат — в трёх остальных случаях. Поэтому для второй строки таблицы 1.1 получается, что единственному варианту значений a_7, a_8 соответствуют три варианта значений a_9, a_{10}, и наоборот, для третьей строки таблицы 1.1 — три варианта значений a_7, a_8 соответствуют единственному варианту значений a_9, a_{10} (в котором необходимо учесть отрицания).

Таблица 1.1

$(a_7 \vee a_8)$	$(\neg a_9 \vee \neg a_{10})$	
0	0	❶
0	1	❷
1	0	❸

Таблица 1.2

	a_7	a_8	a_9	a_{10}
❶	0	0	1 ($\neg$0)	1 ($\neg$0)
❷	0	0	0 ($\neg$1)	0 ($\neg$1)
			0 ($\neg$1)	1 ($\neg$0)
			1 ($\neg$0)	0 ($\neg$1)
❸	1	1	1 ($\neg$0)	1 ($\neg$0)
	0	1		
	1	0		

2. Аналогично анализируется предпоследнее уравнение: $(a_5 \vee a_6) \wedge (\neg a_7 \vee \neg a_8) = 0$.

Таблица 2

$(a_5 \vee a_6)$	$(\neg a_7 \vee \neg a_8)$
0	0
0	1
1	0

Таблица 2.1

$(a_5 \vee a_6)$	$(\neg a_7 \vee \neg a_8)$	
0	0	❶
0	1	❷
1	0	❸

Таблица 2.2

	a_5	a_6	a_7	a_8
❶	0	0	1 ($\neg$0)	1 ($\neg$0)
❷	0	0	0 ($\neg$1)	0 ($\neg$1)
			0 ($\neg$1)	1 ($\neg$0)
			1 ($\neg$0)	0 ($\neg$1)
❸	1	1	1 ($\neg$0)	1 ($\neg$0)
	0	1		
	1	0		

Для всех четырёх уравнений структура и заполнение таблицы возможных значений четырёх используемых в каждом уравнении переменных одна и та же. Смысл решения задачи проявляется во взаимосвязи между этими таблицами, отражающими взаимосвязь между уравнениями.

Переменные a_7 и a_8 используются в обоих рассмотренных уравнениях, поэтому для каждого варианта (набора значений) переменных a_7 и a_8, приведённого в таблице 2.2, нужно определить количество наборов переменных a_7, a_8, a_9, a_{10}, имеющихся в таблице 1.2.

Таблица 2.2 (дополненная)

a_5	a_6	a_7	a_8	Кол-во вариантов с учетом a_9, a_{10}
0	0	1 (¬0)	1 (¬0)	1 ❶
0	0	0 (¬1)	0 (¬1)	4 ❷
		0 (¬1)	1 (¬0)	1 ❸
		1 (¬0)	0 (¬1)	1 ❹
1	1	1 (¬0)	1 (¬0)	1 ❶
0	1			1 ❶
1	0			1 ❶

Таблица 1.2

a_7	a_8	a_9	a_{10}
0	0	1 (¬0)	1 (¬0)
0	0	0 (¬1)	0 (¬1)
		0 (¬1)	1 (¬0)
		1 (¬0)	0 (¬1)
1	1	1 (¬0)	1 (¬0)
0	1		
1	0		

3. Аналогично анализируется второе по счёту уравнение: $(a_3 \vee a_4) \wedge (\neg a_5 \vee \neg a_6) = 0$.

Таблица 3

$(a_3 \vee a_4)$	$(\neg a_5 \vee \neg a_6)$
0	0
0	1
1	0

Таблица 3.1

$(a_3 \vee a_4)$	$(\neg a_5 \vee \neg a_6)$
0	0 ❶
0	1 ❷
1	0 ❸

Таблица 3.2

	a_3	a_4	a_5	a_6
❶	0	0	1 (¬0)	1 (¬0)
❷	0	0	0 (¬1)	0 (¬1)
			0 (¬1)	1 (¬0)
			1 (¬0)	0 (¬1)
❸	1	1	1 (¬0)	1 (¬0)
	0	1		
	1	0		

Так как переменные a_5 и a_6 используются в двух рассмотренных уравнениях (втором и третьем), для каждого варианта (набора значений) переменных a_5 и a_6, приведённого в таблице 3.2, нужно определить количество наборов переменных a_5, a_6, a_7, a_8, имеющихся в таблице 2.2.

При подсчёте количества вариантов нужно быть внимательным:

- для некоторых сочетаний a_5 и a_6, присутствующих в таблице 3.2, в «ответной» таблице 2.2 может суммироваться несколько строк (как, например, для сочетания a_5, $a_6 = (0,0)$);

- для некоторых сочетаний в таблице 3.2 подсчёт количеств сочетаний в таблице 2.2 дублируется (как, например, для сочетания a_5, $a_6 = (1,1)$);
- возможна и ситуация, когда какие-то строки таблицы 2.2 в подсчёт для таблицы 3.2 не войдут вовсе.

Таблица 3.2 (дополненная)

a_3	a_4	a_5	a_6	Кол-во вариантов с учётом a_7, a_8
0	0	1 (¬0)	1 (¬0)	1 ❶
0	0	0 (¬1)	0 (¬1)	7 (4+1+1+1) ❷
		0 (¬1)	1 (¬0)	1 ❸
		1 (¬0)	0 (¬1)	1 ❹
1	1	1 (¬0)	1 (¬0)	1 ❶
0	1			1 ❸
1	0			1 ❹

Таблица 2.2

	a_5	a_6	a_7	a_8	Кол-во вариантов
❷	0	0	1 (¬0)	1 (¬0)	1
	0	0	0 (¬1)	0 (¬1)	4
			0 (¬1)	1 (¬0)	1
			1 (¬0)	0 (¬1)	1
❶	1	1			1
❸	0	1	1 (¬0)	1 (¬0)	1
❹	1	0			1

4. Аналогично анализируется первое уравнение: $(a_1 \vee a_2) \wedge (\neg a_3 \vee \neg a_4) = 0$.

Таблица 4

$(a_1 \vee a_2)$	$(\neg a_3 \vee \neg a_4)$
0	0
0	1
1	0

Таблица 4.1

$(a_1 \vee a_2)$	$(\neg a_3 \vee \neg a_4)$
0	0 ❶
0	1 ❷
1	0 ❸

Таблица 4.2

	a_1	a_2	a_3	a_4
❶	0	0	1 (¬0)	1 (¬0)
❷	0	0	0 (¬1)	0 (¬1)
			0 (¬1)	1 (¬0)
			1 (¬0)	0 (¬1)
❸	1	1	1 (¬0)	1 (¬0)
	0	1		
	1	0		

Так как переменные a_3 и a_4 используются в двух рассмотренных уравнениях (первом и втором), для каждого варианта (набора значений) переменных a_3 и a_4, приведённого в таблице 4.2, нужно определить количество наборов переменных a_3, a_4, a_5, a_6, имеющихся в таблице 3.2.

Таблица 4.2 (дополненная)

a_1	a_2	a_3	a_4	Кол-во вариантов с учётом a_5, a_6
0	0	1 (¬0)	1 (¬0)	1 ❶
0	0	0 (¬1)	0 (¬1)	10 (7+1+1+1) ❷
		0 (¬1)	1 (¬0)	1 ❸
		1 (¬0)	0 (¬1)	1 ❹
1	1	1 (¬0)	1 (¬0)	1 ❶
0	1			1 ❸
1	0			1 ❹
ИТОГО				10 + 6 = 16 вариантов

Таблица 3.2

	a_3	a_4	a_5	a_6	Кол-во вариантов
❷	0	0	1 (¬0)	1 (¬0)	1
	0	0	0 (¬1)	0 (¬1)	7
			0 (¬1)	1 (¬0)	1
			1 (¬0)	0 (¬1)	1
❶	1	1	1 (¬0)	1 (¬0)	1
❸	0	1			1
❹	1	0			1

Ответ: 16 вариантов.

Задача 17. Сколько различных решений имеет система уравнений?

$a_1 \vee \neg a_2 \vee \neg a_3 \wedge a_4 = 1$

$a_3 \vee \neg a_4 \vee \neg a_5 \wedge a_6 = 1$

$a_5 \vee \neg a_6 \vee \neg a_7 \wedge a_8 = 1$

$a_7 \vee \neg a_8 \vee \neg a_9 \wedge a_{10} = 1$,

где $a_1, a_2, \ldots, a_{10}$ — логические переменные? В ответе не нужно перечислять все различные наборы значений переменных, при которых выполнено данное равенство. В качестве ответа нужно указать количество таких наборов.

Решение

Эта задача похожа на две предыдущие и решается аналогично с некоторыми отличиями.

1. Анализ начинается с последнего уравнения: $a_7 \vee \neg a_8 \vee \neg a_9 \wedge a_{10} = 1$.

При разборе структуры этого уравнения (равно как и последующих) нужно вспомнить о приоритетах выполнения логических операций. В данном случае уравнение содержит операции И и ИЛИ и с учётом приоритетов может быть переписано так: $a_7 \vee \neg a_8 \vee (\neg a_9 \wedge a_{10}) = 1$. Теперь в нём имеются три операнда, соединённых логической операцией ИЛИ, причём два первых операнда представляют собой простые переменные, а третий — результат выполнения логической операции И.

Результат выполнения операции ИЛИ равен единице в трёх случаях — если хотя бы один операнд равен единице, т. е. $a_7 = 1$ или $a_8 = 1$ или $(\neg a_9 \wedge a_{10}) = 1$.

Таблица 1

a_7	$\neg a_8$	$(\neg a_9 \wedge a_{10})$
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Учитывая, что операция И (для третьего операнда рассмотренной выше логической конструкции) даёт результат, равный единице, только в случае, когда соединенные ей переменные $\neg a_9$ и a_{10} обе равны единице, и равный нулю — в остальных трёх случаях, составляется таблица возможных значений для всех четырёх переменных (учитывая отрицания).

Таблица 1.1

a_7	$\neg a_8$	$(\neg a_9 \wedge a_{10})$	
0	0	1	❶
0	1	0	❷
0	1	1	❸
1	0	0	❹
1	0	1	❺
1	1	0	❻
1	1	1	❼

Таблица 1.2

	a_7	a_8	a_9	a_{10}
❶	0	1 ($\neg$0)	0 ($\neg$1)	1
❷	0	0 ($\neg$1)	1 ($\neg$0)	0
			1 ($\neg$0)	1
			0 ($\neg$1)	0
❸	0	0 ($\neg$1)	0 ($\neg$1)	1
❹	1	1 ($\neg$0)	1 ($\neg$0)	0
			1 ($\neg$0)	1
			0 ($\neg$1)	0
❺	1	1 ($\neg$0)	0 ($\neg$1)	1
❻	1	0 ($\neg$1)	1 ($\neg$0)	0
			1 ($\neg$0)	1
			0 ($\neg$1)	0
❼	1	0 ($\neg$1)	0 ($\neg$1)	1

2. Аналогичным образом анализируется предпоследнее уравнение: $a_5 \vee \neg a_6 \vee \neg a_7 \wedge a_8 = 1$. Поскольку его структура совпадает с рассмотренной выше, можно опустить сделанные ранее рассуждения и сразу записать соответствующие таблицы.

Таблица 2.1

a_5	$\neg a_6$	$(\neg a_7 \wedge a_8)$	
0	0	1	❶
0	1	0	❷
0	1	1	❸
1	0	0	❹
1	0	1	❺
1	1	0	❻
1	1	1	❼

Таблица 2.2

	a_5	a_6	a_7	a_8
❶	0	1 ($\neg$0)	0 ($\neg$1)	1
❷	0	0 ($\neg$1)	1 ($\neg$0)	0
			1 ($\neg$0)	1
			0 ($\neg$1)	0
❸	0	0 ($\neg$1)	0 ($\neg$1)	1
❹	1	1 ($\neg$0)	1 ($\neg$0)	0
			1 ($\neg$0)	1
			0 ($\neg$1)	0
❺	1	1 ($\neg$0)	0 ($\neg$1)	1
❻	1	0 ($\neg$1)	1 ($\neg$0)	0
			1 ($\neg$0)	1
			0 ($\neg$1)	0
❼	1	0 ($\neg$1)	0 ($\neg$1)	1

Так как переменные a_7 и a_8 используются в обоих рассмотренных уравнениях, для каждого варианта (набора значений) переменных a_7 и a_8, приведённого в таблице 2.2, нужно определить количество наборов переменных a_7, a_8, a_9, a_{10}, имеющихся в таблице 1.2.

Таблица 2.2 (дополненная)

a_5	a_6	a_7	a_8	Кол-во вариантов
0	1 (¬0)	0 (¬1)	1	1 ❶
0	0 (¬1)	1 (¬0)	0	4 ❹
		1 (¬0)	1	4 ❸
		0 (¬1)	0	4 ❷
0	0 (¬1)	0 (¬1)	1	1 ❶
1	1 (¬0)	1 (¬0)	0	4 ❹
		1 (¬0)	1	4 ❸
		0 (¬1)	0	4 ❷
1	1 (¬0)	0 (¬1)	1	1 ❶
1	0 (¬1)	1 (¬0)	0	4 ❹
		1 (¬0)	1	4 ❸
		0 (¬1)	0	4 ❷
1	0 (¬1)	0 (¬1)	1	1 ❶

Таблица 1.2

a_7	a_8	a_9	a_{10}
0	1 (¬0)	0 (¬1)	1 ❶
0	0 (¬1)	1 (¬0)	0
		1 (¬0)	1 ❷
		0 (¬1)	0
0	0 (¬1)	0 (¬1)	1 ❶
1	1 (¬0)	1 (¬0)	0
		1 (¬0)	1 ❸
		0 (¬1)	0
1	1 (¬0)	0 (¬1)	1 ❶
1	0 (¬1)	1 (¬0)	0
		1 (¬0)	1 ❹
		0 (¬1)	0
1	0 (¬1)	0 (¬1)	1 ❶

3. Анализируется второе по счёту уравнение: $a_3 \vee \neg a_4 \vee \neg a_5 \wedge a_6 = 1$.

Таблица 3.1

a_3	$\neg a_4$	$(\neg a_5 \wedge a_6)$
0	0	1 ❶
0	1	0 ❷
0	1	1 ❸
1	0	0 ❹
1	0	1 ❺
1	1	0 ❻
1	1	1 ❼

Таблица 3.2

	a_3	a_4	a_5	a_6
❶	0	1 (¬0)	0 (¬1)	1
❷	0	0 (¬1)	1 (¬0)	0
			1 (¬0)	1
			0 (¬1)	0
❸	0	0 (¬1)	0 (¬1)	1
❹	1	1 (¬0)	1 (¬0)	0
			1 (¬0)	1
			0 (¬1)	0
❺	1	1 (¬0)	0 (¬1)	1
❻	1	0 (¬1)	1 (¬0)	0
			1 (¬0)	1
			0 (¬1)	0
❼	1	0 (¬1)	0 (¬1)	1

Переменные a_5 и a_6 используются в двух уравнениях (втором и третьем), поэтому для каждого варианта (набора значений) переменных a_5 и a_6, приведённого в таблице 3.2, нужно определить количество наборов переменных a_5, a_6, a_7, a_8, имеющихся в таблице 2.2 (с учётом количества уже подсчитанных в ней вариантов).

Таблица 3.2 (дополненная)

a_3	a_4	a_5	a_6	Кол-во вариантов	
0	1 (¬0)	0 (¬1)	1	1	❶
0	0 (¬1)	1 (¬0)	0	13	❹
		1 (¬0)	1	13	❸
		0 (¬1)	0	13	❷
0	0 (¬1)	0 (¬1)	1	1	❶
1	1 (¬0)	1 (¬0)	0	13	❹
		1 (¬0)	1	13	❸
		0 (¬1)	0	13	❷
1	1 (¬0)	0 (¬1)	1	1	❶
1	0 (¬1)	1 (¬0)	0	13	❹
		1 (¬0)	1	13	❸
		0 (¬1)	0	13	❷
1	0 (¬1)	0 (¬1)	1	1	❶

Таблица 2.2

a_5	a_6	a_7	a_8	Кол-во вариантов	
0	1 (¬0)	0 (¬1)	1	1	❶
0	0 (¬1)	1 (¬0)	0	4	
		1 (¬0)	1	4	
		0 (¬1)	0	4	
0	0 (¬1)	0 (¬1)	1	1	❷
1	1 (¬0)	1 (¬0)	0	4	
		1 (¬0)	1	4	
		0 (¬1)	0	4	
1	1 (¬0)	0 (¬1)	1	1	❸
1	0 (¬1)	1 (¬0)	0	4	
		1 (¬0)	1	4	
		0 (¬1)	0	4	
1	0 (¬1)	0 (¬1)	1	1	❹

4. Анализируется первое уравнение: $a_1 \vee \neg a_2 \vee \neg a_3 \wedge a_4 = 1$.

Таблица 4.1

a_1	$\neg a_2$	$(\neg a_3 \wedge a_4)$	
0	0	1	❶
0	1	0	❷
0	1	1	❸
1	0	0	❹
1	0	1	❺
1	1	0	❻
1	1	1	❼

Таблица 4.2

	a_1	a_2	a_3	a_4
❶	0	1 (¬0)	0 (¬1)	1
❷	0	0 (¬1)	1 (¬0)	0
			1 (¬0)	1
			0 (¬1)	0
❸	0	0 (¬1)	0 (¬1)	1
❹	1	1 (¬0)	1 (¬0)	0
			1 (¬0)	1
			0 (¬1)	0
❺	1	1 (¬0)	0 (¬1)	1
❻	1	0 (¬1)	1 (¬0)	0
			1 (¬0)	1
			0 (¬1)	0
❼	1	0 (¬1)	0 (¬1)	1

Переменные a_3 и a_4 используются в двух уравнениях (первом и втором), поэтому для каждого варианта (набора значений) переменных a_3 и a_4, приведенного в таблице 4.2, нужно определить количество наборов переменных a_3, a_4, a_5, a_6, имеющихся в таблице 3.2 (с учётом количества уже подсчитанных в ней вариантов).

Таблица 4.2 (дополненная)

a_1	a_2	a_3	a_4	Кол-во вариантов	
0	1 (¬0)	0 (¬1)	1	1	❶
0	0 (¬1)	1 (¬0)	0	40	❹
		1 (¬0)	1	40	❸
		0 (¬1)	0	40	❷
0	0 (¬1)	0 (¬1)	1	1	❶
1	1 (¬0)	1 (¬0)	0	40	❹
		1 (¬0)	1	40	❸
		0 (¬1)	0	40	❷
1	1 (¬0)	0 (¬1)	1	1	❶
1	0 (¬1)	1 (¬0)	0	40	❹
		1 (¬0)	1	40	❸
		0 (¬1)	0	40	❷
1	0 (¬1)	0 (¬1)	1	1	❶
ИТОГО				$9 \times 40 + 4 \times 1 = 364$ варианта	

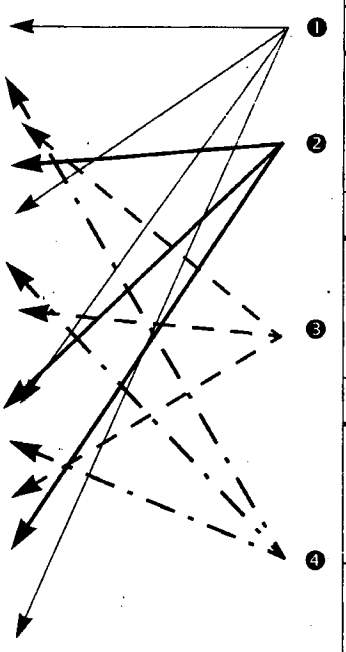


Таблица 3.2

a_3	a_4	a_5	a_6	Кол-во вариантов
0	1 (¬0)	0 (¬1)	1	1
0	0 (¬1)	1 (¬0)	0	13
		1 (¬0)	1	13
		0 (¬1)	0	13
0	0 (¬1)	0 (¬1)	1	1
1	1 (¬0)	1 (¬0)	0	13
		1 (¬0)	1	13
		0 (¬1)	0	13
1	1 (¬0)	0 (¬1)	1	1
1	0 (¬1)	1 (¬0)	0	13
		1 (¬0)	1	13
		0 (¬1)	0	13
1	0 (¬1)	0 (¬1)	1	1

Ответ: 364 варианта.

Задача 18. Сколько различных решений имеет система уравнений

$a_1 \to a_2 \to a_3 \to a_4 \to a_5 = 1,$

$b_1 \to b_2 \to b_3 \to b_4 \to b_5 = 0.$

где $a_1, a_2, \ldots, a_5, b_1, b_2, \ldots, b_5$ — логические переменные? В ответе не нужно перечислять все различные наборы значений переменных, при которых выполнено данное равенство. В качестве ответа нужно указать количество таких наборов.

Решение

Данная система уравнений отличается от ранее рассмотренных, поэтому её решение будет несколько иным (хотя и тоже основанным на таблицах).

1. Первое уравнение: $a_1 \to a_2 \to a_3 \to a_4 \to a_5 = 1$. В нём имеется пять операций следования, приоритет выполнения которых поочерёдно слева направо. Поэтому можно вести анализ данного уравнения, последовательно «раскладывая» его на операции (в чём-то аналогично преобразованию развёрнутой формы записи числа в некоторой системе счисления в схему Горнера).

1.1. Объединяются первые четыре операции следования в скобках (что правомерно, поскольку не меняет порядка выполнения операций) и обозначается содержимое этой скобки как α:

$$a_1 \to a_2 \to a_3 \to a_4 \to a_5 = 1 \Rightarrow \underbrace{(a_1 \to a_2 \to a_3 \to a_4)}_{\alpha} \to a_5 = 1 \Rightarrow \alpha \to a_5 = 1.$$

Вспомнив таблицу истинности логической операции следования, нетрудно составить таблицу возможных вариантов значений переменных α и a_5, при которых результат выполнения этой операции будет равен 1:

Таблица 1.1

α	a_5
0	0
0	1
1	1

1.2. Продолжая «расплетание» логического уравнения, можно записать его часть, которая обозначена как α, так:

$$a_1 \to a_2 \to a_3 \to a_4 = \alpha \Rightarrow \underbrace{(a_1 \to a_2 \to a_3)}_{\beta} \to a_4 = \alpha \Rightarrow \beta \to a_4 = \alpha.$$

Для этой операции следования таблица возможных вариантов значений переменных β и a_4 составляется точно так же, но при этом надо расписывать эти варианты для всех значений α, которые были определены в таблице 1.1 (откуда берутся соответствующие значения переменной a_5):

Таблица 1.1

α	a_5
0	0
0	1
1	1

Таблица 2.1

	β	a_4	α	a_5
❶	1	0	0	0
❷	1	0	0	1
❸	0	0	1	1
	0	1		
	1	1		

1.3. Логическое уравнение «расплетается» дальше и записывается его часть, которая обозначена как β:

$$a_1 \to a_2 \to a_3 = \beta \Rightarrow \underbrace{(a_1 \to a_2)}_{\gamma} \to a_3 = \beta \Rightarrow \gamma \to a_3 = \beta.$$

Для этой операции следования расписывается таблица возможных вариантов значений переменных γ и a_3 аналогичным образом, учитывая все значения β, которые были определены в таблице 2.1, и беря оттуда соответствующие значения переменных a_4 и a_5:

Таблица 2.1

β	a_4	α	a_5
1	0	0	0
1	0	0	1
0	0	1	1
0	1		
1	1		

Таблица 3.1

	γ	a_3	β	a_4	a_5
❶	0	0	1	0	0
	0	1			
	1	1			
❷	0	0	1	0	1
	0	1			
	1	1			
❸	1	0	0	0	1
❹	1	0	0	1	1
❺	0	0	1	1	1
	0	1			
	1	1			

1.4. Последней операцией является запись той части логического уравнения, которая обозначена как γ:

$a_1 \to a_2 = \gamma$.

Для этой операции следования расписывается таблица возможных вариантов значений переменных a_1 и a_2, учитывая все значения γ, которые были определены в таблице 3.1, и беря оттуда соответствующие значения переменных a_3, a_4 и a_5:

Таблица 3.1

γ	a_3	β	a_4	a_5
0	0	1	0	0
0	1			
1	1			
0	0	1	0	1
0	1			
1	1			
1	0	0	0	1
1	0	0	1	1
0	0	1	1	1
0	1			
1	1			

Таблица 4.1

	a_1	a_2	γ	a_3	a_4	a_5
❶	1	0	0	0	0	0
❷	1	0	0	1	0	0
❸	0	0	1	1	0	0
	0	1				
	1	1				
❹	1	0	0	0	0	1
❺	1	0	0	1	0	1
❻	0	0	1	1	0	1
	0	1				
	1	1				
❼	0	0	1	0	0	1
	0	1				
	1	1				
❽	0	0	1	0	1	1
	0	1				
	1	1				
❾	1	0	0	0	1	1
❿	1	0	0	1	1	1
⓫	0	0	1	1	1	1
	0	1				
	1	1				
21 вариант			←	**ИТОГО**		

Таким образом, первое уравнение даёт **21 вариант** возможных значений переменных $a_1 - a_5$.

2. Второе уравнение: $b_1 \to b_2 \to b_3 \to b_4 \to b_5 = 0$. Для него можно было бы получить все возможные варианты значений переменных $b_1 - b_5$ аналогично тому, как было сделано для первого уравнения. Однако учитывая, что требуется только количество таких вариантов, можно упростить решение: если уравнение $b_1 \to b_2 \to b_3 \to b_4 \to b_5 = 1$ (по аналогии с первым) имеет 21 вариант решения, а всего возможных решений уравнения — $2^5 = 32$ варианта (так как в нём пять переменных), то на уравнение $b_1 \to b_2 \to b_3 \to b_4 \to b_5 = 0$ приходятся оставшиеся $32 - 21 = \mathbf{11}$ **вариантов** значений переменных $b_1 - b_5$.

Полезно запомнить следующее **правило**: если известно количество решений уравнения $F(x_1, x_2, \ldots x_n) = 1$, то количество возможных решений «противоположного» уравнения $F(x_1, x_2, \ldots x_n) = 0$ равно разности количества всех возможных комбинаций значений переменных $x_1, x_2, \ldots x_n$ (которое равно 2^n) и количества решений уравнения $F(x_1, x_2, \ldots x_n) = 1$ (и, соответственно, наоборот):

Кол-во решений $F(x_1, x_2, \ldots x_n) = 1$	**Кол-во всех возможных комбинаций $x_1, x_2, \ldots x_n$**	**Кол-во решений $F(x_1, x_2, \ldots x_n) = 0$**
X_1		X_0
=	2^n	=
$2^n - X_0$		$2^n - X_1$

Это правило легко доказать, рассмотрев полную таблицу истинности логической функции $F(x_1, x_2, \ldots x_n)$: если исключить из неё строки, соответствующие значению $F = 1$, то останутся строки, соответствующие значению $F = 0$, и наоборот.

3. Поскольку первое и второе уравнения никак не связаны друг с другом (не имеют общих переменных), то очевидно, что каждому возможному решению первого уравнения можно сопоставить все возможные решения второго уравнения (или наоборот). Поэтому общее количество решений данной системы уравнений можно вычислить как произведение количеств решений каждого уравнения в отдельности:

$a_1 \to a_2 \to a_3 \to a_4 \to a_5 = 1$ $\qquad$ $b_1 \to b_2 \to b_3 \to b_4 \to b_5 = 0$

$$21 \times 11 = \textbf{231 вариант.}$$

Ответ: 231 вариант.

Задача 19. Сколько различных решений имеет система уравнений

$x_1 \to x_2 \to x_3 \to x_4 \to x_5 = 1,$

$y_1 \to y_2 \to y_3 \to y_4 \to y_5 = 0,$

$x_1 \vee y_1 = 0,$

где $x_1, x_2, \ldots, x_5, y_1, y_2, \ldots, y_5$ — логические переменные? В ответе не нужно перечислять все различные наборы значений переменных, при которых выполнено данное равенство. В качестве ответа нужно указать количество таких наборов.

Решение

Очевидна аналогия этой задачи как с предыдущей (№ 18), так и с ранее рассмотренными задачами №№ 8–14. Первые два уравнения выступают как «генераторы» вариантов решений, а последнее является «ключом» и осуществляет «фильтрацию» среди ранее полученного полного набора возможных вариантов иксов и игреков.

1. Определяется полный перечень решений первого уравнения (аналогично тому, как это было сделано в предыдущей задаче):

$(x_1, x_2, x_3, x_4, x_5)$ = (10000, 10100, 00100, 01100, 11100, 10001, 10101, 00101, 01101, 11101, 00001, 01001, 11001, 00011, 01011, 11011, 10011, 10111, 00111, 01111, 11111)

2. Определяется полный перечень решений второго уравнения. Это можно сделать аналогично первому уравнению, но можно использовать и более простой способ.

Генерируется полный перечень возможных комбинаций пяти логических переменных. Это — ряд двоичных чисел от 00000 до 11111 включительно:

(00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, 01000, 01001, 01010, 01011, 01100, 01101, 01110, 01111, 10000, 10001, 10010, 10011, 10100, 10101, 10110, 10111, 11000, 11001, 11010, 11011, 11100, 11101, 11110, 11111).

Количество таких вариантов равно 2^n, где n — количество двоичных разрядов. В данном случае для 5 переменных (5 двоичных разрядов) количество вариантов равно $2^5 = 32$.

3. В полученном полном наборе возможных комбинаций значений переменных вычеркиваются варианты, которые ранее были определены для первого уравнения.

(00000, ~~00001~~, 00010, ~~00011~~, ~~00100~~, ~~00101~~, 00110, ~~00111~~, 01000, ~~01001~~, 01010, ~~01011~~, ~~01100~~, ~~01101~~, 01110, ~~01111~~, ~~10000~~, ~~10001~~, 10010, ~~10011~~, ~~10100~~, ~~10101~~, 10110, ~~10111~~, 11000, ~~11001~~, 11010, ~~11011~~, ~~11100~~, ~~11101~~, 11110, ~~11111~~).

Оставшиеся варианты являются решениями второго уравнения:
(00000, 00010, 00110, 01000, 01010, 01110, 10010, 10110, 11000, 11010, 11110).

Такое решение следует из:

а) идентичности первого и второго уравнения;

б) того, что второе уравнение должно быть равно 0, а не 1, как первое: очевидно, что для получения комбинаций значений переменных, приводящих к нулевому результату, надо брать все возможные их комбинации, кроме тех, которые приводят к результату 1.

4. Анализируется третье («ключевое») уравнение: $x_1 \vee y_1 = 0$. Здесь операция ИЛИ связывает первые переменные в наборах значений иксов и игреков, причём искомое значение 0 достигается только если обе связываемые переменные равны нулю: $x_1 = 0$ и $y_1 = 0$.

5. Составляется «матрица решений» (как и раньше, анализируемые цифры в ней выделяются подчеркиванием):

$(x_1x_2x_3x_4x_5)$	$(y_1y_2y_3y_4y_5)$											*Кол-во вариантов*
	(00000)	(00010)	(00110)	(01000)	(01010)	(01110)	(10010)	(10110)	(11000)	(11010)	(11110)	
(00001)	+	+	+	+	+	+	—	—	—	—	—	**6**
(00011)	+	+	+	+	+	+	—	—	—	—	—	**6**
(00100)	+	+	+	+	+	+	—	—	—	—	—	**6**
(00101)	+	+	+	+	+	+	—	—	—	—	—	**6**
(00111)	+	+	+	+	+	+	—	—	—	—	—	**6**
(01001)	+	+	+	+	+	+	—	—	—	—	—	**6**
(01011)	+	+	+	+	+	+	—	—	—	—	—	**6**
(01100)	+	+	+	+	+	+	—	—	—	—	—	**6**
(01101)	+	+	+	+	+	+	—	—	—	—	—	**6**
(01111)	+	+	+	+	+	+	—	—	—	—	—	**6**
(10000)	—	—	—	—	—	—	—	—	—	—	—	—
(10001)	—	—	—	—	—	—	—	—	—	—	—	—
(10011)	—	—	—	—	—	—	—	—	—	—	—	—
(10100)	—	—	—	—	—	—	—	—	—	—	—	—
(10101)	—	—	—	—	—	—	—	—	—	—	—	—
(10111)	—	—	—	—	—	—	—	—	—	—	—	—
(11001)	—	—	—	—	—	—	—	—	—	—	—	—
(11011)	—	—	—	—	—	—	—	—	—	—	—	—
(11100)	—	—	—	—	—	—	—	—	—	—	—	—
(11101)	—	—	—	—	—	—	—	—	—	—	—	—
(11111)	—	—	—	—	—	—	—	—	—	—	—	—
Всего возможных вариантов (пар) наборов значений $(x_1x_2x_3x_4x_5)$ ***и*** $(y_1y_2y_3y_4y_5)$***:***												**60**

Задачи для самостоятельного решения

1. Сколько различных решений имеет уравнение
   $((K \vee \neg L \vee M) \to \neg(N \to \neg O)) \wedge ((N \wedge O) \to (K \vee \neg L \vee M)) \wedge (N \vee O \vee \neg L) = 1$, где K, L, M, N, O — логические переменные?
2. Сколько существует различных наборов значений логических переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, которые удовлетворяют всем перечисленным ниже условиям?
   $(z_1 \to z_2) \wedge (z_2 \to z_3) \wedge (z_3 \to z_4) \wedge (z_4 \to z_5) = 1$
   $(f_1 \to f_2) \wedge (f_2 \to f_3) \wedge (f_3 \to f_4) = 1$

   В качестве ответа Вам нужно указать количество таких наборов.
3. Сколько существует различных наборов значений логических переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5, которые удовлетворяют всем перечисленным ниже условиям?
   $(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1$,
   $(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1$,
   $y_4 \to x_4 = 1$.

   В качестве ответа Вам нужно указать количество таких наборов.
4. Сколько существует различных наборов значений логических переменных x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5,, которые удовлетворяют всем перечисленным ниже условиям?
   $(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) \wedge (x_4 \to x_5) = 1$,
   $(y_1 \to y_2) \wedge (y_2 \to y_3) \wedge (y_3 \to y_4) \wedge (y_4 \to y_5) = 1$,
   $x_3 \to y_3 = 1$.

   В качестве ответа Вам нужно указать количество таких наборов.
5. Сколько различных решений имеет система уравнений
   $\neg(z_1 \equiv z_2) \wedge \neg(z_2 \equiv z_3) = 1$,
   $\neg(z_2 \equiv z_3) \wedge \neg(z_3 \equiv z_4) = 1$,
   ...
   $\neg(z_8 \equiv z_9) \wedge \neg(z_9 \equiv z_{10}) = 1$,
   где z_1, z_2,..., z_{10} — логические переменные?

   В качестве ответа вам нужно указать количество таких наборов.

Ответы для самопроверки

№ задания	Ответ
1	9
2	30
3	28
4	27
5	2

Элементы теории алгоритмов

A5 Анализ работы автомата, формирующего число по заданным правилам

Конспект

При решении задач, в которых предполагаются вычисления в недесятичной системе счисления, полезна следующая таблица.

Основание системы счисления	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
***Max* цифра**	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
***Max* сумма цифр**	10_2	11_3	12_4	13_5	14_6	15_7	16_8	17_9	18_{10}	19_{11}	$1A_{12}$	$1B_{13}$	$1C_{14}$	$1D_{15}$	$1E_{16}$

Разбор типовых задач

Задача 1. Устройство считывает три двухзначных числа и строит по ним новое число по следующему алгоритму:

1) вычисляется сумма старших разрядов заданных чисел;

2) вычисляется сумма младших разрядов заданных чисел;

3) эти суммы записываются друг за другом без разделителей по возрастанию.

Пример. Заданы двухзначные числа: 11, 19, 87. Поразрядные суммы: 10, 17. Результат: 1017.

Какое из чисел может быть результатом работы такого устройства.

1) 2528 2) 129 3) 311 4) 1613

Решение

В данной задаче речь идёт о сложении цифр неизвестных чисел. Пусть эти числа обозначаются как X, Y и Z и записываются их в виде:

$X = x_1x_2$; $Y = y_1y_2$; $Z = z_1z_2$

Суммы цифр, которые вычисляет автомат, есть:

$S_1 = x_1 + y_1$ и $S_2 = x_2 + y_2$.

Чтобы оценить возможные значения этих сумм, следует сложить цифры. Старшая цифра в двузначном числе может иметь значение от 1 до 9 (если она равна нулю, то число уже не будет двузначным!), а младшая — значение от 0 до 9. Тогда сумма $S_1 = x_1 + y_1$ может иметь значение от 3 до 27 (исходных чисел — три), а сумма $S_2 = x_2 + y_2$ — значение от 0 до 27.

Теперь нужно просмотреть предлагаемые варианты ответов.

1) *Число 2528.* Оно может быть составлено из сумм 25 и 28. Но тогда вторая сумма превышает максимально возможное значение суммы трёх цифр (27), поэтому данный вариант ответа не может быть правильным.

2) *Число 129*. Его можно представить или как сочетание сумм 1 и 29, или как сочетание сумм 12 и 9. Но первый вариант не подходит (сумма старших цифр трёх двузначных чисел может иметь наименьшее значение 3). Варианты 12 и 9 тоже не годится в качестве ответа, так как в условии задачи указано, что суммы должны записываться по возрастанию. Значит, этот вариант ответа тоже неверен.

3) *Число 311*. Рассуждая аналогично, его можно представить тоже как 3 и 11 или как 31 и 1. Очевидно, второй вариант записи не подходит, — а вот первый (3 и 11) вполне удовлетворяет условию задачи.

4) *Число 1613*. Его можно разделить на числа (суммы) 16 и 13, но они записаны не по возрастанию, значит данный вариант ответа тоже ложный.

Следовательно, правильным является ответ 311, как запись сумм 3 и 11.

Ответ: число 311 (вариант ответа №3).

Задача 2. Устройство считывает три двухзначных числа и строит по ним новое число по следующему алгоритму:

1) вычисляется сумма старших разрядов заданных чисел;

2) вычисляется сумма младших разрядов заданных чисел;

3) эти суммы записываются друг за другом без разделителей по убыванию.

Пример. Заданы двухзначные числа: 11, 19, 87. Поразрядные суммы: 10, 17. Результат: 1710. Какое из чисел **НЕ** может быть результатом работы такого устройства.

1) 228 2) 282 3) 120 4) 222

Решение

В данной задаче, в отличие от предыдущей, требуется найти единственный *неправильный* вариант ответа, тогда как остальные три — правильные. В остальном же решение этой задачи аналогично предыдущей.

1) *Число 228*. Можно представить его как 2 и 28 или как 22 и 8. Первый вариант не годится, но второй удовлетворяет условиям задачи.

2) *Число 282*. Его можно представить как 2 и 82 или как 28 и 2. Ни один из этих двух вариантов не годится: в обоих случаях одна из предполагаемых сумм выходит за пределы возможного диапазона значений суммы трёх цифр (от 3 до 27 — для старших и от 0 до 27 — для младших цифр).

3) *Число 120*. Представимо как 1 и 20 или как 12 и 0. Второй вариант удовлетворяет условиям задачи.

4) Число 222. Представимо как 2 и 22 или как 22 и 2. Второй вариант также удовлетворяет условиям задачи.

Следовательно, единственный вариант ответа, который НЕ может быть результатом работы описанного в задаче автомата, — это число 282.

Ответ: число 282 (вариант ответа №2).

Задача 3. Устройство считывает четырёхзначное восьмеричное число и строит по нему новое число по следующему алгоритму:

1) вычисляется сумма первой и второй цифр;

2) вычисляется сумма третьей и четвёртой цифр;

3) эти суммы записываются друг за другом без разделителей по убыванию.

Пример. Задано число 7145. Суммы: 7 + 1 = 10; 4 + 5 = 11. Результат: 1110.

Какое из чисел может быть результатом работы такого устройства.

1) 119 2) 1213 3) 1411 4) 1715

Решение

Эта задача отличается от предыдущих тем, что к ней вычисления производятся в недесятичной системе счисления. В остальном принцип решения остается тот же.

Обозначаются неизвестные цифры числа как x, y, z и t (подразумевается, что само число тогда имеет вид: $xyzt$).

Суммы цифр, которые вычисляет автомат, — это:

$S_1 = x + y$ и $S_2 = z + t$.

Оценивая возможные значения этих сумм, складываются восьмеричные цифры, которые могут меняться от 0 до 7. Тогда, согласно правилам восьмеричной арифметики, сумма двух таких цифр может меняться от 0 до $7_8 + 7_8 = 16_8$.

Предлагаемые варианты ответов:

1) *Число 119.* Можно представить его как 11 и 9 или как 1 и 19. Однако ни один из этих вариантов записи не годится: в первом случае восьмеричная запись суммы не может содержать цифры больше 7, а во втором число 19 не может быть суммой двух восьмеричных цифр.

2) *Число 1213.* Его можно представить как 12 и 13. Оба эти числа могут быть суммами восьмеричных цифр (так как удовлетворяют допустимому диапазону значений таких сумм). Однако они записаны по возрастанию (а не по убыванию, как требуется в условии задачи). Поэтому данное число не может быть решением.

3) *Число 1411.* Его можно представить как 14 и 11. Обе эти составляющие могут быть значениями сумм восьмеричных цифр, записаны они по убыванию. Значит, это число может быть решением данной задачи.

4) *Число 1715.* Может быть представлено как 17 и 15. Поскольку его первая составляющая (17) превышает максимально возможное значение суммы, данное число не может быть решением задачи.

Следовательно, единственный вариант ответа, который может быть результатом работы описанного в задаче автомата, — это число 1411.

Ответ: число 1411 (вариант ответа №3).

Задача 4. Алёша и Лена придумали такую игру с числами. Лена записывает четырёхзначное шестнадцатеричное число, в котором все цифры не превышают 6, а Алёша придумывает новое шестнадцатеричное число по следующему алгоритму:

1) вычисляется шестнадцатеричная сумма двух первых разрядов числа, записанного Леной, и сумма двух последних разрядов этого числа;

2) две вычисленные шестнадцатеричные суммы записываются друг за другом без разделителей по убыванию.

Пример. Лена записала число 3456. Поразрядные суммы: 7, B. Число, придуманное Алёшей: B7.

Какое из чисел может получиться у Алёши (исходное число, записанное Леной, не известно).

1) 93 2) D5 3) 119 4) 6B

Решение

Задача решается аналогично предыдущей, однако при определении возможного диапазона значений сумм следует учитывать, что в задаче используется шестнадцатеричная система счисления, а на значения цифр исходного числа наложено ограничение: они не могут быть больше 6.

Обозначаются неизвестные цифры числа как x, y, z и t (подразумевается, что само число тогда имеет вид: $xyzt$).

Суммы цифр, которые вычисляет автомат, — это:

$S_1 = x + y$ и $S_2 = z + t$.

Оценивая возможные значения этих сумм, складываются шестнадцатеричные цифры, которые по условию задачи могут меняться от 0 до 6. Тогда, согласно правилам шестнадцатеричной арифметики, сумма двух таких цифр может меняться от 0 до $6_{16} + 6_{16} = C_{16}$ (12_{10}).

Теперь просматриваются предлагаемые варианты ответов.

1) *Число 93.* Можно представить его как 9 и 3. Обе эти составляющие входят в допустимый диапазон значений суммы, записаны они по убыванию, значит, данное число может быть решением задачи.

2) *Число D5.* Его можно представить как D и 5. Поскольку первая составляющая (D) превышает максимальное значение шестнадцатеричной суммы двух шестерок, данное число не может быть решением.

3) *Число 119*. Его можно представить как 11 и 9 или как 1 и 19. В обоих вариантах представления числа в нём имеются составляющие, не являющиеся шестнадцатеричными цифрами (11 и 19) и превышающие максимально возможное значение суммы двух шестнадцатеричных шестёрок. Значит, это число тоже не может быть решением данной задачи.

4) *Число 6B*. Может быть представлено как 6 и B. Обе эти составляющие могут быть шестнадцатеричными суммами двух шестерок, но записаны они не по убыванию. Поэтому данное число не может быть решением задачи.

Следовательно, единственный вариант ответа, который может быть результатом работы описанного в задаче автомата, — это число 93.

Ответ: число 93 (вариант ответа №1).

Задача 5. Устройство считывает два двухзначных шестнадцатеричных числа, в которых все цифры не превышают 5, и вырабатывает новое шестнадцатеричное число по алгоритму:

1) вычисляется шестнадцатеричная сумма старших разрядов считанных чисел;

2) вычисляется шестнадцатеричная сумма младших разрядов считанных чисел;

3) вычисленные шестнадцатеричные суммы записываются друг за другом без разделителя по возрастанию.

Пример. Заданы числа 66 и 43. Поразрядные суммы: A, 9. Результат: 9A.

Какое из чисел может быть результатом работы такого устройства.

1) 8A 2) 410 3) 9C 4) 76

Решение

Задача решается аналогично предыдущей, но максимальное значение каждой из складываемых цифр здесь равно 5 (а не 6).

Если складываются шестнадцатеричные цифры, которые по условию задачи могут меняться от 0 до 5, то сумма двух таких цифр может меняться от 0 до $5_{16} + 5_{16} = A_{16}$ (10_{10}).

Теперь следует просматреть предлагаемые варианты ответов.

1) *Число 8A*. Можно представить его как 8 и A. Обе эти составляющие входят в допустимый диапазон значений суммы, записаны они по возрастанию, значит, данное число может быть решением задачи.

2) *Число 410*. Его можно представить как 4 и 10 или как 41 и 5. В обоих вариантах представления числа в нём имеются составляющие, не являющиеся шестнадцатеричными цифрами (10 и 41) и превышающие максимально возможное значение суммы двух шестнадцатеричных пятерок. Значит, это число не может быть решением данной задачи.

3) *Число 9C*. Его можно представить как 9 и C. Поскольку вторая составляющая (C) превышает максимальное значение шестнадцатеричной суммы двух пятерок, данное число не может быть решением.

4) *Число 76*. Может быть представлено как 7 и 6. Обе эти составляющие могут быть шестнадцатеричными суммами двух пятерок, но записаны они не по возрастанию. Поэтому данное число не может быть решением задачи.

Следовательно, единственный вариант ответа, который может быть результатом работы описанного в задаче автомата, — это число 8A.

Ответ: число 8A (вариант ответа №1).

Задача 6. Устройство считывает два двухзначных восьмеричных числа и вырабатывает новое восьмеричное число по алгоритму:

1) вычисляется восьмеричная сумма старших разрядов считанных чисел;

2) вычисляется восьмеричная сумма младших разрядов считанных чисел;

3) вычисленные восьмеричные суммы записываются друг за другом без разделителя по убыванию.

Пример. Заданы числа 66 и 24. Поразрядные суммы: 10, 12. Результат: 1210.

Какое из чисел может быть результатом работы такого устройства.

1) 27 2) 112 3) 129 4) 2111

Решение

В данной задаче используется восьмеричная арифметика. Решение аналогично решению задачи №3.

Обозначаются неизвестные числа как $X = x_1x_2$ и $Y = y_1y_2$.

Суммы цифр, которые вычисляет автомат, — это:

$S_1 = x_1 + y_1$ и $S_2 = x_2 + y_2$.

Оценивая возможные значения этих сумм, складываются восьмеричные цифры, которые могут меняться от 0 до 7. Тогда, согласно правилам восьмеричной арифметики, сумма двух таких цифр может меняться от 0 до $7_8 + 7_8 = 16_8$.

Теперь просматриваются предлагаемые варианты ответов.

1) *Число 27.* Можно представить как 2 и 7 или как 0 и 27. Второй случай явно недопустим (число 27 слишком велико). В первом случае обе составляющие входят в допустимый диапазон значений суммы восьмеричных цифр, но записаны не по убыванию. Поэтому данное число не может быть решением задачи.

2) *Число 112.* Его можно представить как 11 и 2 или как 1 и 12. В обоих случаях составляющие числа допустимы по величине, при этом в первом случае они записаны по убыванию. Значит, данное число может быть решением задачи.

3) *Число 129.* Его можно представить как 12 и 9 или как 1 и 29. В первом случае в записи числа присутствует цифра 9, недопустимая в восьмеричной системе счисления. Во втором случае вторая составляющая превышает максимально возможное значение суммы восьмеричных цифр и также содержит недопустимую цифру (9). Следовательно, это число не является решением задачи.

4) *Число 2111.* Может быть представлено как 21 и 11. Первая составляющая превышает максимально возможное значение суммы восьмеричных цифр, поэтому данное число тоже не является решением задачи.

Следовательно, единственный вариант ответа, который может быть результатом работы описанного в задаче автомата, — это число 112.

Ответ: число 112 (вариант ответа №2).

Задача 7. Устройство считывает трёхзначное десятичное число и вырабатывает новое число по алгоритму:

1) вычисляется произведение первой и второй цифр считанного числа;

2) вычисляется произведение второй и третьей цифр считанного числа;

3) вычисленные произведения записываются друг за другом без разделителя по возрастанию.

Пример. Задано трёхзначное число 157. Произведения: $1 \cdot 5 = 5$; $5 \cdot 7 = 35$. Результат: 535.

Какое из чисел может быть результатом работы такого устройства.

1) 1214 2) 1612 3) 2433 4) 244

Решение

Здесь вместо сложения используется операция умножения цифр, но смысл решения остается прежним.

Обозначается неизвестное число как $X = x_1x_2x_3$. Тогда произведения цифр, которые вычисляет автомат, — это:

$P_1 = x_1 \cdot x_2$ и $P_2 = x_2 \cdot x_3$

Оценивая возможные значения этих произведений, умножаются десятичные цифры, которые могут меняться от 0 до 9. Произведение двух таких цифр может меняться от 0 до 81. Однако из-за того, что цифра x_2 входит в оба этих произведения, кроме вхождения в допустимый диапазон при анализе записи числа необходимо также проверять, чтобы обе составляющие числа имели общий делитель, равный от 1 до 9, и при делении на него также давали результат от 1 до 9.

Теперь просматриваются предлагаемые варианты ответов.

1) *Число 1214.* Можно представить его как 12 и 14. Обе составляющие входят в допустимый диапазон значений произведений десятичных цифр, имеют общий делитель 2 ($12 = 2 \cdot 6$, $14 = 2 \cdot 7$), записаны по возрастанию, поэтому данное число может быть решением задачи.

2) *Число 1612.* Его можно представить как 16 и 12. Обе составляющие входят в допустимый диапазон значений произведений десятичных цифр, но записаны не по возрастанию, поэтому данное число не является решением задачи.

3) *Число 2433.* Его можно представить как 24 и 33. Обе составляющие входят в допустимый диапазон значений произведений десятичных цифр и имеют единственный общий делитель 3. Но $24 = 3 \cdot 7$, а $33 = 3 \cdot 11$, т. е. число 33 не может быть произведением цифр при общем делителе 3. Поэтому данное число не является решением задачи.

4) *Число 244.* Может быть представлено как 2 и 44 или как 24 и 4. В первом случае общим делителем может быть только 2, но тогда второе число $44 = 2 \cdot 22$ и не может быть произведением цифр. Во втором случае имеется два общих делителя: 2 и 4. При общем делителе, равном 2, получается: $24 = 2 \cdot 12$. Такой вариант, очевидно, недопустим. При общем делителе, равном 4, получается: $24 = 4 \cdot 6$. Этот вариант «разложения» числа 244 допустим, но составляющие 24 и 4 записаны не по возрастанию. Поэтому данное число тоже не является решением задачи.

Следовательно, единственный вариант ответа, который может быть результатом работы описанного в задаче автомата, — это число 1214.

Ответ: число 1214 (вариант ответа №1).

Задача 8. Автомат получает на вход два трёхзначных числа. По этим числам строится новое число по следующим правилам.

1. Вычисляются три числа — сумма старших разрядов заданных трёхзначных чисел, сумма средних разрядов этих чисел, сумма младших разрядов.

2. Полученные три числа записываются друг за другом в порядке убывания (без разделителей).

Пример. Исходные трёхзначные числа: 835, 196. Поразрядные суммы: 9, 12, 11. Результат: 12119.

Определите, какое из следующих чисел может быть результатом работы автомата.

1) 15012 2) 191313 3) 121111 4) 10911

Решение

Усложнение данной задачи по сравнению с предыдущими — в том, что автомат вычисляет не два, а три числа.

Обозначаются неизвестные числа как $X = x_1x_2x_3$ и $Y = y_1y_2y_3$.

Суммы цифр, которые вычисляет автомат, — это:

$S_1 = x_1 + y_1$, $S_2 = x_2 + y_2$ и $S_3 = x_3 + y_3$.

Используется десятичная система счисления, поэтому цифры могут меняться от 0 до 9, а их сумма может меняться от 0 до 18.

Теперь просматриваются предлагаемые варианты ответов.

1) *Число 15012.* Можно представить следующим образом:

- 1, 50, 12 — число 50 превышает максимально возможное значение суммы цифр;
- 15, 0, 12 — значения составляющих допустимы, но записаны не по убыванию.

Поэтому данное число не может быть решением задачи.

2) *Число 191313.* Его можно представить как 19, 13 и 13. Первая составляющая превышает максимально допустимую величину суммы двух десятичных цифр. Поэтому данное число тоже не может быть решением задачи.

3) *Число 121111.* Его можно представить как 12, 11 и 11. Все три составляющие соответствуют диапазону возможных значений суммы двух десятичных цифр и записаны по убыванию (невозрастанию). Следовательно, это число может являться решением задачи.

4) *Число 10911.* Можно представить его следующим образом:

• 10, 9, 11 — значения составляющих допустимы, но записаны не по убыванию;
• 10, 91, 1 — число 91 превышает максимально возможное значение суммы цифр.

Поэтому данное число не может быть решением задачи.

Следовательно, единственный вариант ответа, который может быть результатом работы описанного в задаче автомата, — это число 121111.

Ответ: число 121111 (вариант ответа №3).

Задача 9. Предлагается некоторая операция над двумя произвольными трёхзначными десятичными числами:

1) записывается результат сложения значений старших разрядов заданных чисел;

2) к нему дописывается результат сложения значений средних разрядов этих чисел по такому правилу: если он меньше первой суммы, то второе полученное число приписывается к первому слева, иначе — справа.

3) итоговое число получают приписыванием справа к полученному после второго шага числу суммы значений младших разрядов исходных чисел.

Определите, какое из предложенных чисел может быть результатом такой операции.

1) 161312 2) 171918 3) 131808 4) 141711

Решение

Эта задача почти полностью аналогична предыдущей, изменён только порядок записи получаемых сумм: согласно ему вторая составляющая всегда должна быть больше первой (соотношение третьей составляющей с двумя предыдущими безразлично).

Используется десятичная система счисления, поэтому цифры могут меняться от 0 до 9, а их сумма может меняться от 0 до 18.

Теперь просматриваются предлагаемые варианты ответов.

1) *Число 161312.* Его можно представить как 16, 13 и 12. Все три составляющие соответствуют диапазону возможных значений суммы двух десятичных цифр, но записаны они в порядке, не удовлетворяющем условию задачи (вторая составляющая меньше первой). Значит, данное число не может быть решением задачи.

2) *Число 171918.* Его можно представить как 17, 19 и 18. Вторая составляющая превышает максимально допустимую величину суммы двух десятичных цифр, поэтому данное число тоже не может быть решением задачи.

3) *Число 131808.* Его можно представить как 13, 18 и 08. Здесь запись третьей составляющей некорректна (если бы сумма младших цифр исходных чисел была равна 8, то полученное число имело бы вид 13188). Поэтому данное число также не может являться решением задачи.

4) *Число 141711.* Его можно представить как 14, 17 и 11. Все три составляющие соответствуют диапазону возможных значений суммы двух десятичных цифр, записаны они в порядке, удовлетворяющем условию задачи (вторая составляющая больше первой). Значит, данное число может быть решением задачи.

Следовательно, единственный вариант ответа, который может быть результатом работы описанного в задаче автомата, — это число 141711.

Ответ: число 141711 (вариант ответа №4).

Задачи для самостоятельного решения

1. Устройство получает на вход два двузначных шестнадцатеричных числа. В этих числах все цифры не превосходят цифру 7 (если в числе есть цифра больше 7, автомат отказывается работать). По этим числам строится новое шестнадцатеричное число по следующим правилам.

   1. Вычисляются два шестнадцатеричных числа — сумма старших разрядов заданных чисел и сумма младших разрядов этих чисел.

2. Полученные два шестнадцатеричных числа записываются друг за другом в порядке возрастания (без разделителей).

Определите, какое из предложенных чисел может быть результатом работы автомата.

1) AF 2) 410 3) 4) 76

2. Устройство получает на вход четырёхзначное восьмеричное число. По этому числу строится новое число по следующим правилам.

1. Складываются первая и вторая, а также третья и четвёртая цифры.

2. Полученные два числа записываются друг за другом в порядке возрастания (без разделителей).

Определите, какое из следующих чисел может быть результатом работы автомата.

1) 912 2) 1213 3) 1517 4) 1715

3. Устройство получает на вход трёхзначное пятеричное число. По этому числу строится новое число по следующим правилам.

1. Складываются первая и вторая, а также вторая и третья цифры.

2. Полученные два числа записываются друг за другом в порядке убывания (без разделителей).

Определите, какое из следующих чисел может быть результатом работы такого устройства.

1) 35 2) 604 3) 81 4) 91

4. Автомат получает на вход четыре трёхзначных троичных числа. По этим числам строится новое число по следующим правилам.

1. Вычисляются три числа — сумма старших разрядов заданных чисел, сумма их средних разрядов и сумма младших разрядов.

2. Полученные три числа записываются друг за другом без разделителей так: первой записывается большая из двух сумм: старших и младших разрядов, затем записывается сумма средних разрядов, а потом справа дописывается оставшаяся сумма.

Какое из следующих чисел **не** может быть результатом работы такого автомата.

1) 738 2) 782 3) 321 4) 100

5. Автомат получает на вход два трёхзначных числа. По этим числам строится новое число по следующим правилам.

1. Вычисляются три числа — сумма старших разрядов заданных трёхзначных чисел, сумма средних разрядов этих чисел, сумма младших разрядов.

2. Полученные три числа записываются друг за другом в порядке убывания (без разделителей).

Определите, какое из следующих чисел может быть результатом работы автомата.

1)151413 2)141513 3)14019 4)191611

Ответы для самопроверки

№ задания	Ответ
1	3
2	2
3	3
4	1
5	1

Элементы теории алгоритмов

A13 Робот в лабиринте

Конспект

Исполнитель — любое устройство или живое существо, которое способно точно и однозначно выполнять заданные команды (действия, заданные алгоритмом).

Система команд исполнителя — совокупность команд, которые он способен выполнять.

Алгоритм — запись в той или иной форме (словесной, графической, на языке программирования) последовательности команд для исполнителя. Команды алгоритма должны соответствовать системе команд исполнителя.

Исполнитель РОБОТ — перемещается по клеткам лабиринта.

Типичная система команд:

- перемещение: вверх, вниз, влево, вправо;
- проверка наличия препятствия (стенки): сверху свободно, снизу свободно, слева свободно, справа свободно — играют роль условия;
- команда ветвления:

```
ЕСЛИ <условие>
       ТО команда1
       ИНАЧЕ команда2
КОНЕЦ ЕСЛИ
```

— выполняется *команда1* (если *условие* истинно) или *команда2* (если *условие* ложно), где условие — это команда проверки наличия препятствия;

- команда цикла:

```
ПОКА <условие> команда
```

или

```
ПОКА <условие>
     последовательность команд
КОНЕЦ ПОКА
```

— выполняется, пока *условие* истинно (условие — это команда проверки наличия препятствия).

Разбор типовых задач

Задача 1*. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	вниз	влево	вправо

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

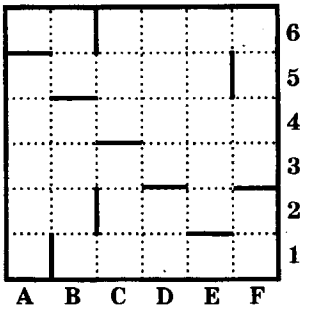
Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

Цикл ПОКА <условие> команда выполняется, пока условие истинно, иначе происходит переход на следующую строку.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ остановится в той же клетке, с которой он начал движение?

```
НАЧАЛО
ПОКА <справа свободно> вправо
ПОКА <сверху свободно> вверх
ПОКА <слева свободно> влево
ПОКА <снизу свободно> вниз
КОНЕЦ
```



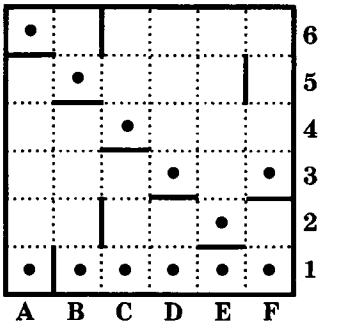
1) 1 2) 0 3) 3 4) 4

Решение

Проще всего решить эту задачу, что называется, «в лоб», — поочередно перебирая каждую клетку лабиринта и «проводя» робота из неё по маршруту согласно заданной программе. Но такой способ требует слишком много времени, поскольку приходится проверять 36 вариантов (клеточек). Поэтому основная задача — научиться заранее отсекать заведомо неподходящие варианты, чтобы найти решение достаточно быстро (так как время на ЕГЭ ограничено).

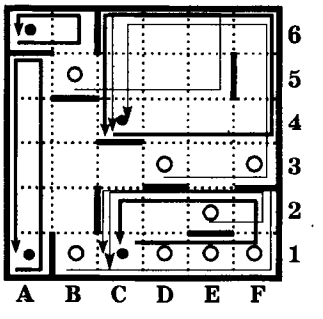
Прежде всего, нужно обратить внимание на то, что Робот, с какой бы клеточки он ни начинал движение, однозначно останавливается (согласно программе) перед преграждающей ему путь стенкой. Поэтому проще всего сначала определить клеточки, в которых Робот должен согласно его программе завершать движение. Таких клеточек будет не так много, а дальше можно проверять каждую из них на соответствие условию: мог ли Робот прийти в эту клетку, начав движение с неё же?

В данном случае Робот завершает программу, если обнаруживает стенку снизу. Отмечаются на схеме лабиринта все такие точки, — их оказывается всего 12 (а не 36):



Теперь они поочередно перебираются и проверяются, сможет ли Робот, начав путь из какой-либо точки, вернуться в неё. При этом следует учесть, что если движение в каком-то направлении невозможно, так как оно изначально перекрыто стенкой, то соответствующая строка программы пропускается, и происходит переход к следующей.

На рисунке ниже показаны маршруты движения Робота из каждой отмеченной точки (толстые стрелки — Робот возвращается в ту же точку; тонкие стрелки — Робот останавливается в другой клетке).



Итого — 4 клеточки, удовлетворяющие условию.

Ответ: 4 (вариант №4).

Задача 2. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	вниз	влево	вправо

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре команды проверяют истинность условия отсутствия стены у каждой той клетки, где находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

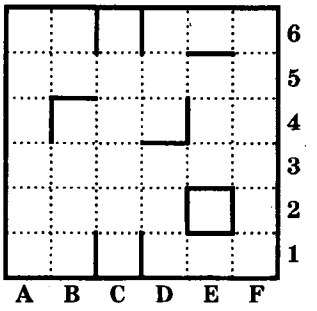
Цикл

ПОКА <условие> команда

выполняется, пока условие истинно, иначе происходит переход на следующую строку.

Сколько клеток приведенного лабиринта соответствует требованию, что, выполнив предложенную ниже программу, РОБОТ уцелеет и остановится в той же клетке, с которой он начал движение?

```
НАЧАЛО
ПОКА <сверху свободно> вправо
ПОКА <справа свободно> вниз
ПОКА <снизу свободно> влево
ПОКА <слева свободно> вверх
```



1) 1 2) 2 3) 3 4) 4

Решение

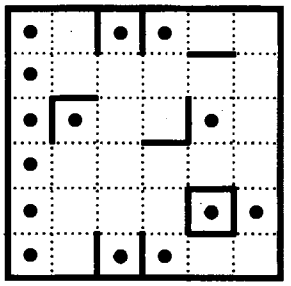
В этой задаче есть дополнительное усложнение. Ранее каждая команда ПОКА содержала как условие, так и команду движения, «одинаковой направленности», например: ПОКА <снизу свободно> вниз. Это было очевидным: РОБОТ двигается вниз, пока не встретит на своём пути стенку и не остановится. Теперь же направления движения и условия остановки РОБОТа в каждой команде ПОКА различны. Поэтому при решении такой задачи следует быть более внимательными: РОБОТ либо может остановиться, когда стенка (как условие остановки движения) появится в очередной клетке, в которую он перейдёт, с соответствующего бока, либо может разрушиться, если первой встретится стенка, перекрывающая ему путь (тогда как с соответствующего бока стенки, которая бы его остановила, нет). В условии таких задач обычно особо оговаривается факт разрушения РОБОТа, натолкнувшегося на полном ходу на стенку (в данном случае оно опущено).

В остальном же принципы решения такой задачи традиционны.

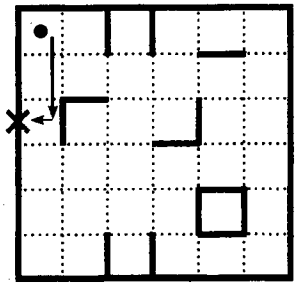
Сначала надо обратить внимание на последнюю команду программы РОБОТа, которая определяет клетку, в которой РОБОТ может остановиться (если, конечно, не разобьётся раньше) по окончании программы. В данном случае это команда

ПОКА <слева свободно> вверх

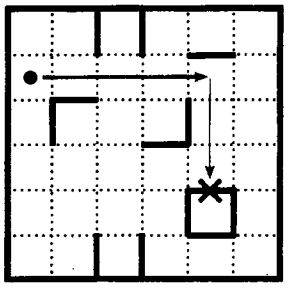
т. е. РОБОТ может остановиться только в клетках, в которых имеется левая стенка. Все такие клетки помечаются кружочками:



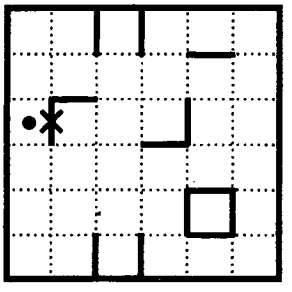
Далее нужно проверить работу программы РОБОТа для каждой отмеченной точки, прорисовывая получаемые траектории и особо отмечая клетки, для которых выполняется условие задачи: совпадение начальной и конечной точки маршрута. Для наглядности это лучше разработать на отдельных рисунках для каждой намеченной ранее ячейки:



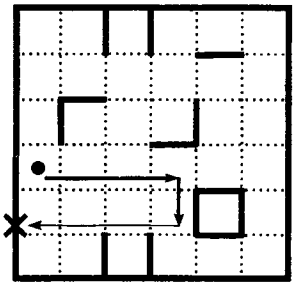
— сверху есть стенка — вправо движения нет;
— движение вниз, пока справа не появится стенка;
— снизу стенки нет, поэтому РОБОТ пойдёт влево и разобьётся.



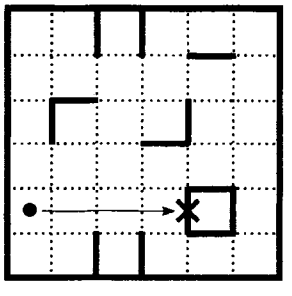
— движение вправо, пока не появится стенка сверху;
— движение вниз, пока справа не появится стенка: однако ещё до этого РОБОТ наткнётся на стенку и разобьётся.



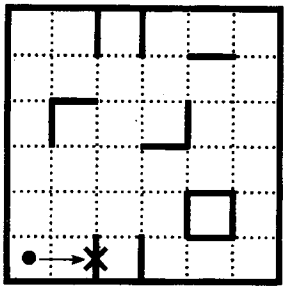
— очевидно, РОБОТ разобьётся сразу же, поскольку попытается двигаться вправо (сверху стенки нет).



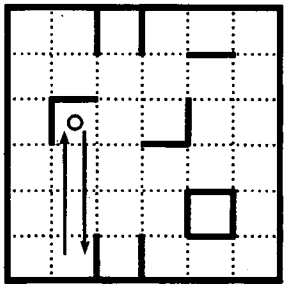
— движение вправо, пока не появится стенка сверху;
— движение вниз, пока не появится стенка справа;
— движение влево, пока не появится стенка снизу, а поскольку её нет, РОБОТ разобьётся.



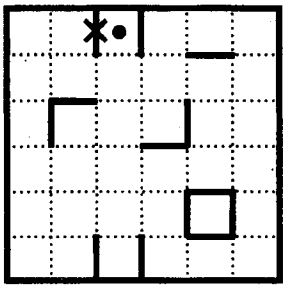
— движение вправо, пока не появится стенка сверху; очевидно, при этом РОБОТ разобьётся.



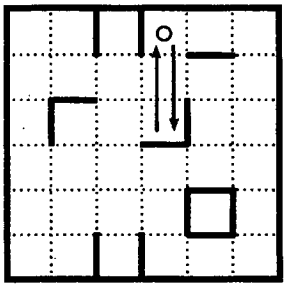
— движение вправо, пока не появится стенка сверху; аналогично, при этом РОБОТ разобьётся.



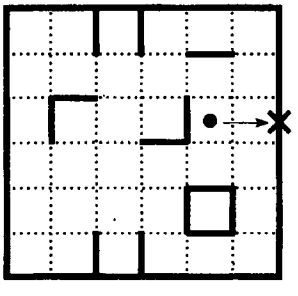
— сверху есть стенка — вправо движения нет;
— движение вниз, пока справа не появится стенка;
— снизу есть стенка — влево движения нет;
— движение вверх, пока слева не появится стенка.
ДАННАЯ КЛЕТКА УДОВЛЕТВОРЯЕТ УСЛОВИЮ ЗАДАЧИ



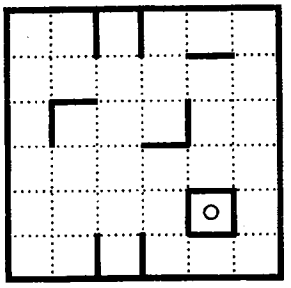
— сверху есть стенка — вправо движения нет;
— справа есть стенка — вниз движения нет;
— снизу стенки нет, поэтому РОБОТ начнёт двигаться влево и разобьётся.



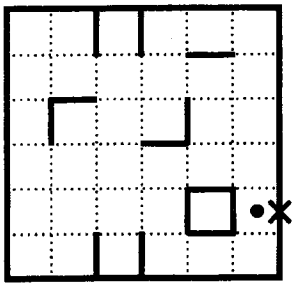
— сверху есть стенка — вправо движения нет;
— движение вниз, пока справа не появится стенка;
— снизу есть стенка — влево движения нет;
— движение вверх, пока слева не появится стенка.
ДАННАЯ КЛЕТКА ТАКЖЕ УДОВЛЕТВОРЯЕТ УСЛОВИЮ ЗАДАЧИ



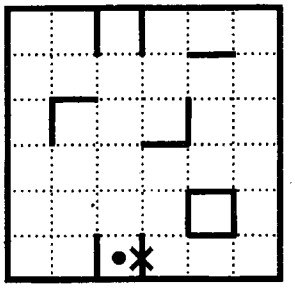
— движение вправо, пока не появится стенка сверху; очевидно, при этом РОБОТ разобьётся.



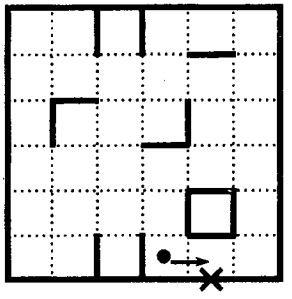
— поскольку данная ячейка окружена стенками со всех сторон, никакого движения РОБОТА не будет. Значит, можно формально считать, что РОБОТ по окончании работы программы будет в той же самой клетке, что и в начале.
ДАННАЯ КЛЕТКА УДОВЛЕТВОРЯЕТ УСЛОВИЮ ЗАДАЧИ



— стенки сверху нет, поэтому РОБОТ начнёт движение вправо и разобьётся.



— аналогично: стенки сверху нет, поэтому РОБОТ начнёт движение вправо и разобьётся.



— движение вправо, пока сверху не появится стенка;
— стенки справа нет, поэтому РОБОТ начнёт движение вниз и разобьётся.

Итого получается, что из всех клеток лабиринта условию задачи удовлетворяют три.
Ответ: 3 клетки (вариант ответа №3).

В подобных задачах в командах ПОКА проверяется наличие соответствующей стенки (левой, правой, верхней или нижней) по отношению к **текущей** ячейке, в которой находится РОБОТ, а не по отношению к самому РОБОТу с учётом направления его движения.

Задача 3*. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	вниз	влево	вправо

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

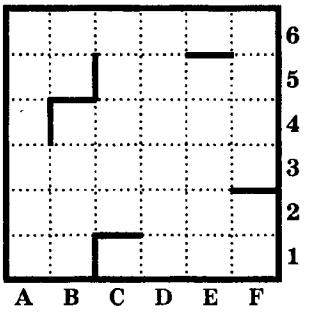
Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

Цикл ПОКА <условие> команда выполняется, пока условие истинно, иначе происходит переход на следующую строку.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ остановится в той же клетке, с которой он начал движение?

```
НАЧАЛО
ПОКА <сверху свободно> вправо
ПОКА <справа свободно> вниз
ПОКА <снизу свободно> влево
ПОКА <слева свободно> вверх
КОНЕЦ
```

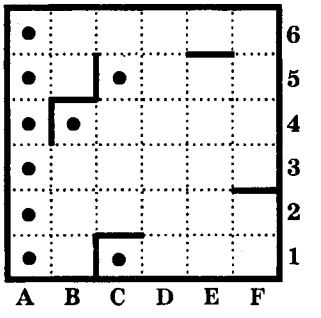


1) 1 2) 2 3) 3 4) 4

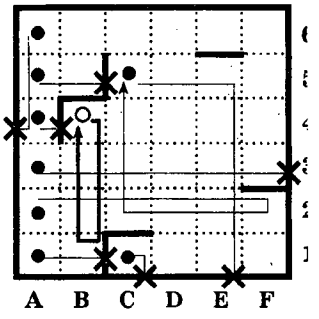
Решение

Данная задача аналогична предыдущей и её решение аналогично.

Размечаются клетки окончания пути по признаку — стенка стоит слева от них:



Проверяются найденные клетки (толстые стрелки — успешный маршрут, тонкие — неуспешный маршрут, крестиком показаны места разрушения РОБОТа при движении на стенку):



Ответ: 1 клеточка (вариант №1).

Задача 4. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	вниз	влево	вправо

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

Цикл

```
ПОКА <условие> последовательность команд
КОНЕЦ ПОКА
```

выполняется, пока условие истинно.

В конструкции

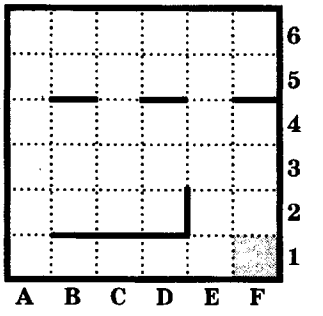
```
ЕСЛИ <условие>
       ТО команда1
       ИНАЧЕ команда2
КОНЕЦ ЕСЛИ
```

выполняется команда1 (если условие истинно) или команда2 (если условие ложно).

Если РОБОТ начнёт движение в сторону находящейся рядом с ним стены, то он разрушится и программа прервётся.

Сколько клеток лабиринта соответствуют требованию, что, начав движение в ней и выполнив предложенную программу, РОБОТ уцелеет и остановится в закрашенной клетке (клетка F6)?

```
НАЧАЛО
ПОКА <справа свободно ИЛИ снизу свободно>
     ПОКА <справа свободно> вправо
     КОНЕЦ ПОКА
     ПОКА <снизу свободно> вниз
     КОНЕЦ ПОКА
КОНЕЦ ПОКА
КОНЕЦ
```



1) 8 2) 12 3) 16 4) 20

Решение

Особенность этой задачи по сравнению с традиционной в том, что здесь программа для РОБОТа состоит не из линейной последовательности из нескольких циклов ПОКА, каждый из которых определяет количество шагов движения в одном из четырёх направлений, а из вложенных циклов ПОКА. Соответственно, иным будет и путь решения задачи.

Проанализируем заданную в условии программу для РОБОТа.

Во-первых, вместо компактной однооператорной записи циклов ПОКА (например: ПОКА <справа свободно> вниз) используется полная запись с использованием особой завершающей команды КОНЕЦ ПОКА. Однако это — лишь формальное изменение, важное только для об-

рамляющего цикла ПОКА, тогда как внутренние (вложенные) циклы легко представить в прежней, компактной форме:

```
НАЧАЛО
ПОКА <справа свободно ИЛИ снизу свободно>
     ПОКА <справа свободно> вправо
     ПОКА <снизу свободно> вниз
КОНЕЦ ПОКА
КОНЕЦ
```

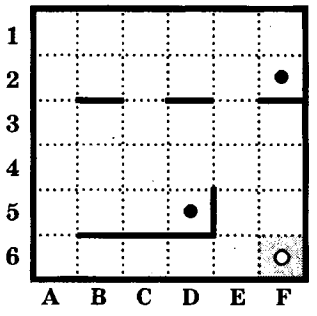
В этом виде программа становится несколько более простой и понятной; в ней легче выявить аналогию с традиционным вариантом задач про РОБОТа.

Во-вторых, внутренние (вложенные) циклы ПОКА, как и раньше, определяют перемещение РОБОТа в соответствующем направлении, пока с указанной стороны от него не будет обнаружена стенка: первый вложенный цикл ПОКА определяет движение вправо до стенки справа, а второй цикл ПОКА — движение вниз до стенки снизу. В сумме же это означает, что РОБОТ должен двигаться «Г-образно»: сначала вправо до препятствия, а потом вниз до препятствия.

В-третьих, условие внешнего цикла ПОКА <справа свободно ИЛИ снизу свободно> показывает, что вся «Г-образная» траектория движения РОБОТа может повторяться многократно, если после остановки РОБОТа из-за препятствия в виде стенки внизу выяснится, что справа стенки нет: тогда РОБОТ может снова двигаться вправо до препятствия вправо, а потом вниз до препятствия снизу. Согласно принципам работы цикла ПОКА, условие внешнего цикла проверяется тогда и только тогда, когда РОБОТ уже завершил проход очередного «Г-образного» фрагмента пути, а не, например, по завершении выполнения первого из двух вложенных циклов ПОКА или в процессе их выполнения. То есть отработка каждого из внутренних циклов ПОКА является процессом, полностью не зависимым от условия внешнего цикла.

Наконец, запись условия внешнего цикла: <справа свободно ИЛИ снизу свободно> означает, что завершение выполнения этого внешнего цикла ПОКА и, соответственно, завершение работы программы РОБОТа происходит, когда препятствие (стенка) имеется И справа, И снизу от текущего расположения РОБОТа. Следовательно, «финальной» ячейкой (имеется в виду «благополучный» финал при корректном завершении работы программы, а не авария, когда РОБОТ разбивается о «незамеченную» им стенку; впрочем, структура вложенных циклов ПОКА такой аварийный случай исключает) может быть только ячейка, ограниченная стенками снизу и справа.

В представленном лабиринте:



этим условиям соответствуют только три ячейки: F6 (требуемая), F2 и D5 («ложные»). При этом «ложные» ячейки можно рассматривать как «ловушки» для РОБОТа, попадание в которые делает требуемый результат работы его программы невозможным.

Теперь, вернувшись к вложенным циклам ПОКА нужно рассмотреть, как РОБОТ будет согласно этим командам двигаться по лабиринту.

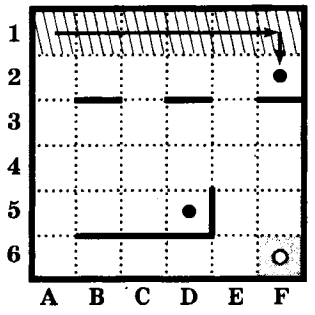
Очевидно, что первый вложенный цикл ПОКА для любой исходной ячейки определяет одно и то же движение РОБОТа вправо до ближайшей стенки справа. Следовательно, можно разделить весь лабиринт на такие фрагменты (последовательности ячеек в одной строке), которые завершаются стенкой справа (внешней границей лабиринта или промежуточной стенкой).

И если хотя бы одна ячейка каждого такого фрагмента удовлетворяет либо, наоборот, не удовлетворяет условию задачи (является либо не является решением задачи), то тогда и все ячейки соответствующего фрагмента являются либо не являются решением. В данном лабиринте эти фрагменты (придерживаясь для их обозначения привычной по электронным таблицам записи диапазонов) будут следующими:

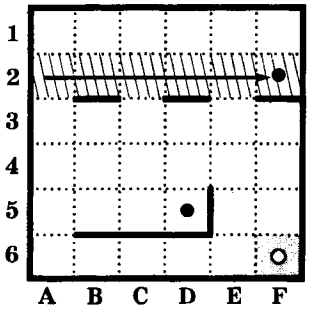
A1:F1, A2:F2, A3:F3, A4:F4, A5:D5, E5:F5, A6:F6.

Учитывая сказанное выше, не обязательно проверять все ячейки лабиринта — достаточно в каждом диапазоне проверить только по одной любой ячейке — например, крайней слева (т.е. проверить, например, ячейки A1, A2, A3, A4, A5, E5 и A6). Однако в случае успешного перемещения РОБОТа из проверяемой ячейки в «целевую» ячейку F6 нужно подсчитать как удовлетворяющие условию задачи все ячейки соответствующего диапазона.

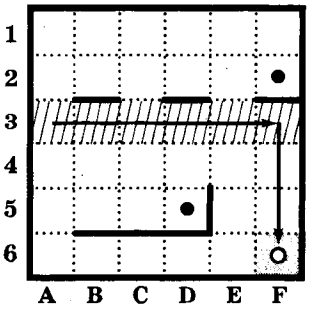
Ячейка A1. Движение РОБОТа вправо будет происходить до правой границы лабиринта, после чего он будет двигаться вниз до ячейки F2 («ловушки»). Поэтому ни одна ячейка диапазона A1:F1 не подходит.



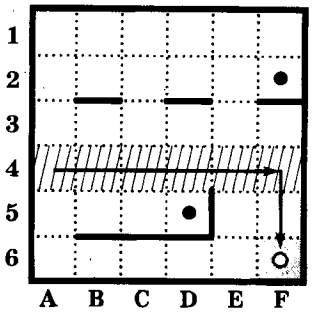
Ячейка A2. РОБОТ из неё тоже будет двигаться вправо до правой границы лабиринта (до ячейки-«ловушки» F2), после чего в ней и останется, — второй вложенный цикл ПОКА просто не будет выполняться, так как снизу от ячейки F2 есть стенка. Значит, ни одна ячейка диапазона A2:F2 нам тоже не подходит.



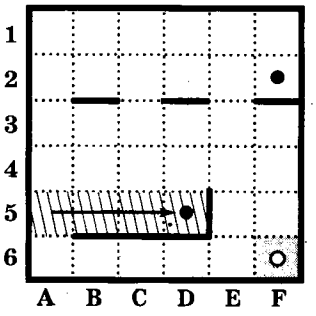
Ячейка A3. Движение РОБОТа вправо опять-таки будет совершаться до правой границы лабиринта, а затем РОБОТ пойдёт вниз до требуемой ячейки F6. Следовательно, все ячейки диапазона A3:F3 (их — 6) являются решениями данной задачи.



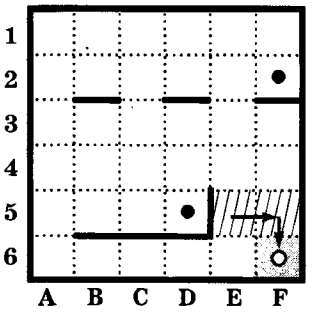
Ячейка A4. Повторив те же самые рассуждения для неё, выясняется, что все ячейки диапазона A4:F4 также являются решениями задачи (это ещё 6 ячеек).



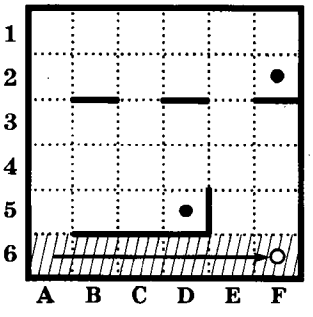
Ячейка A5. Очевидно, что РОБОТ при движении вправо попадёт в ячейку-«ловушку» D5 и останется в ней (так как вниз идти ему некуда). Поэтому все ячейки диапазона A5:D5 не пригодны.



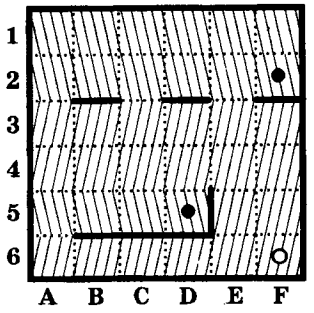
Ячейка E5. Здесь, напротив, РОБОТ, выполняя свою программу, благополучно дойдёт до нужной ячейки F6. Тем самым определились ещё 2 ячейки диапазона E5:F5, которые являются решением задачи.



Ячейка A6. Для неё, а значит, и для всех 6 ячеек диапазона A6:F6 выполнение РОБОТом заданной программы благополучно приведёт его в нужную ячейку F6.



Итого решениями задачи являются: 6 ячеек диапазона A3:F3, 6 ячеек диапазона A4:F4, 2 ячейки диапазона E5:F5 и 6 ячеек диапазона A6:F6 (включая «целевую» F6: если РОБОТ изначально в ней находился, то в ней он и останется, т. е. условие задачи тоже будет выполнено), то есть всего — 6 + 6 + 2 + 6 = 20 ячеек.



Ответ: 20 клеток (вариант ответа №4).

Задача 5. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	вниз	влево	вправо

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

Цикл

```
ПОКА <условие> последовательность команд
КОНЕЦ ПОКА
```

выполняется, пока условие истинно.

В конструкции

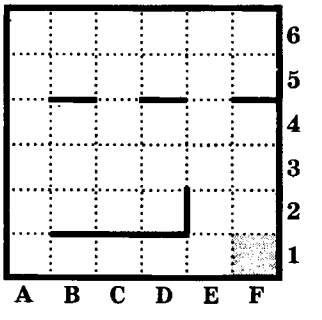
```
ЕСЛИ <условие>
      ТО команда1
      ИНАЧЕ команда2
КОНЕЦ ЕСЛИ
```

выполняется команда1 (если условие истинно) или команда2 (если условие ложно).

Если РОБОТ начнёт движение в сторону находящейся рядом с ним стены, то он разрушится и программа прервётся.

Сколько клеток лабиринта соответствуют требованию, что, начав движение в ней и выполнив предложенную программу, РОБОТ уцелеет и остановится в закрашенной клетке (клетка F6)?

```
НАЧАЛО
ПОКА <справа свободно ИЛИ снизу свободно>
     ПОКА <снизу свободно> вниз
     КОНЕЦ ПОКА
     ПОКА <справа свободно> вправо
     КОНЕЦ ПОКА
КОНЕЦ ПОКА
КОНЕЦ
```



1) 14 3) 19
2) 17 4) 21

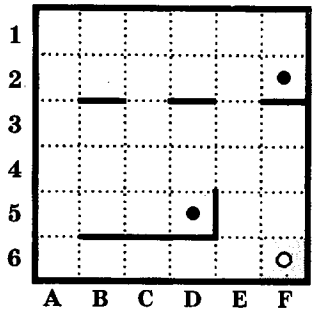
Решение

1. Вложенные циклы — их можно записать в компактной форме:

```
ПОКА <снизу свободно> вниз
ПОКА <справа свободно> вправо
```

определяют «Г-образное» движение РОБОТа сначала вниз до обнаружения стенки внизу, а затем вправо до обнаружения стенки справа.

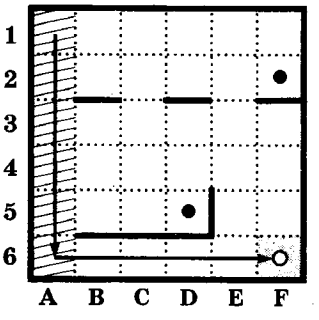
2. Внешний цикл ПОКА с условием `<справа свободно ИЛИ снизу свободно>` указывает, что такие «Г-образные» фрагменты пути РОБОТа могут повторяться, если по завершении выполнения предыдущего такого фрагмента выяснится, что можно продолжить движение вниз. Завершится же этот внешний цикл ПОКА (и, соответственно, вся программа) тогда, когда стенка будет И снизу, И справа от текущего местоположения РОБОТа. То есть, финальными могут быть ячейки, ограниченные стенками снизу и справа.



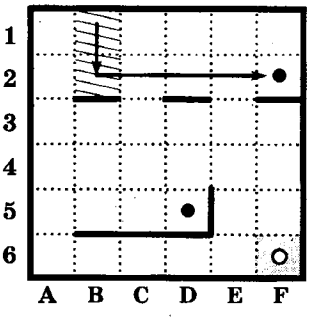
Из трёх таких ячеек две (F2 и D5) — ложные («ловушки»), а одна — требуемая согласно условию задачи (F6).

3. Из вложенных циклов ПОКА первым отрабатывается цикл, который определяет движение РОБОТа по вертикали вниз до обнаружения первой же стенки снизу. Поэтому весь лабиринт можно разделить на фрагменты (диапазоны) столбцов ячеек, и в каждом таком диапазоне все его ячейки «равноправны» (т.е. все они или являются, или не являются решениями задачи, если хотя бы одна любая ячейка диапазона является или, наоборот, не является решением). Поэтому из выделенных диапазонов: A1:A6, B1:B2, B3:B5, B6 (диапазон может состоять и только из одной ячейки!), C1:C5, C6, D1:D2, D3:D5, D6, E1:E6, F1:F2, F3:F6, — достаточно проверять ячейки: A1, B1, B3, B6, C1, C6, D1, D3, D6, E1, F1 и F3.

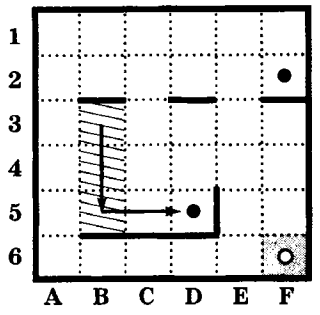
4. **Ячейка A1.** РОБОТ движется вниз до нижней границы лабиринта, затем при выполнении второго вложенного цикла ПОКА следует до требуемой ячейки F6. Значит, все 6 ячеек диапазона A1:A6 являются решениями задачи.



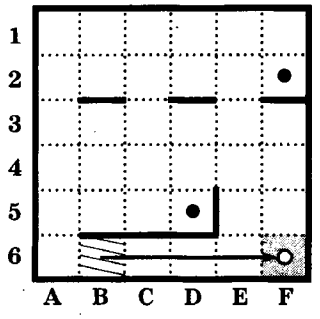
Ячейка B1. РОБОТ дойдёт до промежуточной стенки, а затем, двигаясь вправо, попадёт в «ловушку» — ячейку F2. Значит, все ячейки диапазона B1:B2 не являются решением задачи.



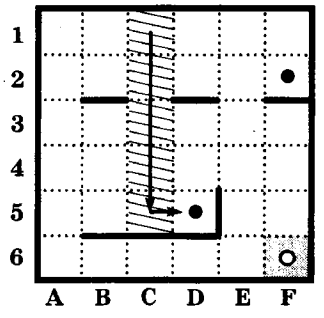
Ячейка B3. РОБОТ, выполняя программу, попадёт в «ловушку» D5, т. е. весь диапазон B3:B5 не является решением.



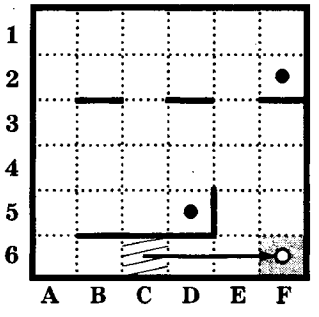
Ячейка B6 (она одна в соответствующем диапазоне). РОБОТ вниз идти не может (внизу — граница лабиринта), поэтому первый из вложенных циклов не выполняется. В ходе же выполнения второго вложенного цикла РОБОТ благополучно попадает в требуемую ячейку F6. Значит, ячейка B6 — решение задачи.



Ячейка C1. Повторяя рассуждения для неё, выясняется, что РОБОТ в результате выполнения программы попадёт в ячейку-«ловушку» D5. Значит, все ячейки рассматриваемого диапазона C1:C5 не являются решением.

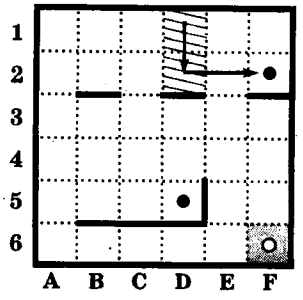


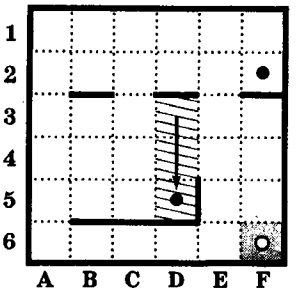
Ячейка C6. Как и в случае с ячейкой B6, она является решением.



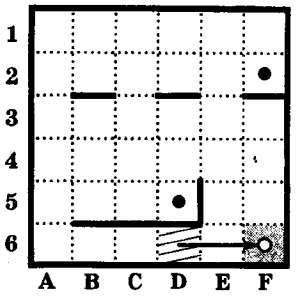
Рассуждая далее в том же духе, нетрудно сделать выводы:

- ячейка D1 (и вообще все ячейки диапазона D1:D2) приведёт РОБОТа в «ловушку» F2, а ячейка D3 (как и все ячейки диапазона D3:D5) — в «ловушку» D5;

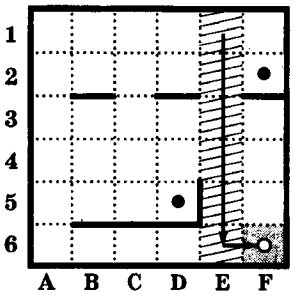




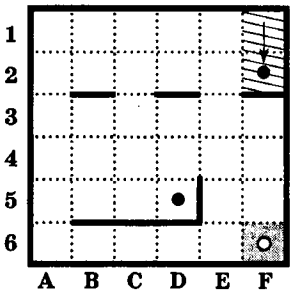
• ячейка D6 является решением;

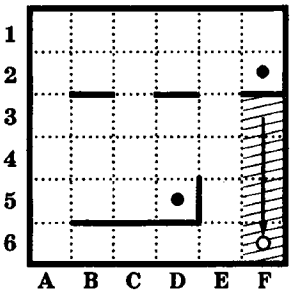


• ячейка E1 (равно как и все ячейки диапазона E1:E6) тоже являются решениями (их — 6);

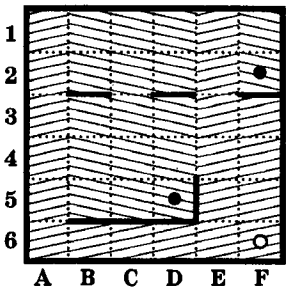


• ячейка F1 (и весь диапазон F1:F2) решением не является, так как приводит РОБОТа в «ловушку» F2, а вот ячейка F3 и вообще все 4 ячейки диапазона F3:F6 (включая «целевую») являются решениями.





Итого решениями задачи являются: 6 ячеек диапазона A1:A6, ячейки B6, C6, D6, 6 ячеек диапазона E1:E6 и ещё 4 ячейки диапазона F3:F6. Всего — 6 + 1 + 1 + 1 + 6 + 4 = 19 ячеек.



Ответ: 19 ячеек (вариант ответа №3).

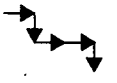
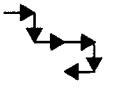
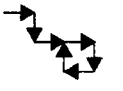
Задача 6*. Исполнитель РОБОТ действует на клетчатой доске, между соседними клетками которой могут стоять стены. РОБОТ передвигается по клеткам доски и может выполнять команды 1 (вверх), 2 (вниз), 3 (вправо), 4 (влево), переходя на соседнюю клетку в направлении, указанном в скобках. Если в этом направлении между клетками стоит стена, то РОБОТ разрушается. Робот успешно выполнил программу 3233241.

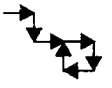
Какую последовательность из трёх команд должен выполнить РОБОТ, чтобы вернуться в ту клетку, где он был перед началом выполнения программы, и не разрушиться вне зависимости от того, какие стены стоят на поле?

Решение

Это одна из более ранних задач про РОБОТа, несколько иная по характеру. Вид лабиринта здесь неизвестен, поэтому для решения требуется опираться только на информацию об успешности прохождения заданного маршрута. Очевидно, что нужно построить такую эквивалентную программу, чтобы РОБОТ шёл по тем же самым клеткам, что и по исходной программе.

Вручную «интерпретируется» исходная программа для РОБОТа и одновременно прочерчивается его маршрут:

Шаг программы	1	2	3	4	5	6	7
Команда	3	2	3	3	2	4	1
Расшифровка команды	вправо	вниз	вправо	вправо	вниз	влево	вверх
Текущий маршрут Робота							

Полученный маршрут: 

Этот маршрут содержит петлю, которую можно исключить и тем самым уменьшить длину программы. Оптимизированный маршрут имеет вид:

Соответствующая программа имеет вид: 323 (совпадает с началом исходной программы).
Обратный маршрут имеет вид:

Соответствующая программа: 414 (влево, вверх, влево)
Ответ: 414.

Задача 7*. Исполнитель Робот ходит по клеткам бесконечной вертикальной клетчатой доски, переходя по одной из команд вверх, вниз, вправо, влево в соседнюю клетку в указанном направлении. Робот выполнил следующую программу:

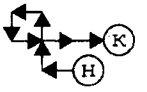
```
влево
вверх
вверх
влево
вниз
вправо
вправо
вправо
```

Укажите наименьшее возможное число команд в программе, приводящей РОБОТа из той же начальной клетки в ту же конечную.

Решение

Эта задача во многом аналогична предыдущей — и даже проще неё, поскольку в условии не сказано, что нужно при движении РОБОТа опасаться стенок. Решение же в целом аналогично предыдущему.

1. Рисуется маршрут РОБОТа, получаемый по исходной программе (на рисунке приведён окончательный вид маршрута; «н» — начало, «к» — конец):



В этом маршруте имеется петля, которую можно устранить, тем самым сократив длину программы. Кроме того, можно из начальной точки сразу идти вверх. Оптимизированный маршрут имеет вид:

Соответствующая программа: вверх, вправо, что составляет две команды.

Ответ: 2.

Задачи для самостоятельного решения

1. Исполнитель Робот действует на клетчатой доске, между соседними клетками которой могут стоять стены. Робот передвигается по клеткам доски и может выполнять команды 1 (вверх), 2 (вниз), 3 (вправо), 4 (влево), переходя на соседнюю клетку в направлении, указанном в скобках. Если в этом направлении между клетками стоит стена, то Робот разрушается. Робот успешно выполнил программу 4413323.
   Какой минимальной по длине последовательностью команд можно заменить эту программу, чтобы Робот не разрушился вне зависимости от того, какие стены стоят на поле?
2. Исполнитель Робот действует на клетчатой доске, между соседними клетками которой могут стоять стены. Робот передвигается по клеткам доски и может выполнять команды 1 (вверх), 2 (вниз), 3 (вправо), 4 (влево), переходя на соседнюю клетку в направлении, указанном в скобках. Если в этом направлении между клетками стоит стена, то Робот разрушается. Робот успешно выполнил программу 331133222441.
   Какую минимальную по длине последовательность команд должен выполнить Робот, чтобы вернуться в ту клетку, где он был перед началом выполнения программы, и не разрушиться вне зависимости от того, какие стены стоят на поле?
3. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	вниз	влево	вправо

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

Цикл

```
ПОКА <условие> последовательность команд
КОНЕЦ ПОКА
```

выполняется, пока *условие* **истинно.**

В конструкции

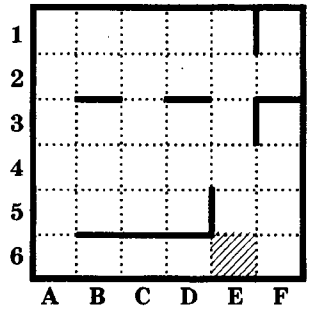
```
ЕСЛИ <условие>
       ТО команда1
       ИНАЧЕ команда2
КОНЕЦ ЕСЛИ
```

выполняется команда1 **(если** условие **истинно) или** команда2 **(если** условие **ложно).**

Если РОБОТ начнёт движение в сторону находящейся рядом с ним стены, то он разрушится и программа прервётся.

Сколько клеток лабиринта соответствуют требованию, что, начав движение в ней и выполнив предложенную программу, РОБОТ уцелеет и остановится в закрашенной клетке (клетка Е6)?

```
НАЧАЛО
ПОКА <справа свободно> И <снизу свободно>
  ПОКА <справа свободно> вправо
  КОНЕЦ ПОКА
    ПОКА <снизу свободно> вниз
          КОНЕЦ ПОКА
    КОНЕЦ ПОКА
КОНЕЦ
```



1) 8 **2) 9** **3) 10** **4) 20**

4. **Система команд исполнителя РОБОТ:**

вверх	вниз	влево	вправо

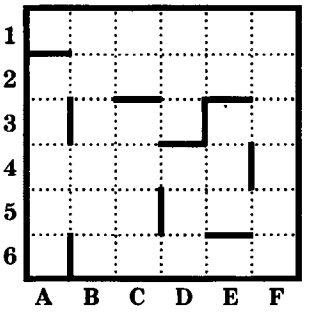
При выполнении команды РОБОТ перемещается на одну клетку: вверх ↑, вниз ↓, влево ←, вправо →.

Следующие команды проверяют истинность условия отсутствия стены у каждой стороны клетки, в которой находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

Сколько клеток соответствуют требованию, что, начав движение в ней и выполнив предложенную программу, РОБОТ уцелеет и остановится в той же клетке, откуда начал движение:

```
ПОКА <слева свободно> вниз
ПОКА <снизу свободно> вправо
ПОКА <справа свободно> вверх
ПОКА <сверху свободно> влево
КОНЕЦ
```



1) 1 **2) 2** **3) 3** **4) 4**

5. **Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:**

вверх	вниз	влево	вправо

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

Цикл

```
ПОКА <условие> последовательность команд
КОНЕЦ ПОКА
```

выполняется, пока условие истинно.

В конструкции

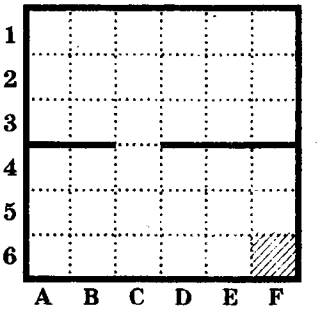
```
ЕСЛИ <условие>
        ТО команда1
        ИНАЧЕ команда2
КОНЕЦ ЕСЛИ
```

выполняется команда1 (если условие истинно) или команда2 (если условие ложно).

Если РОБОТ начнёт движение в сторону находящейся рядом с ним стены, то он разрушится и программа прервётся.

Сколько клеток лабиринта соответствуют требованию, что, начав движение в ней и выполнив предложенную программу, РОБОТ уцелеет и остановится в закрашенной клетке (клетка F6)?

```
НАЧАЛО
ПОКА <снизу свободно ИЛИ справа свободно>
      ЕСЛИ <снизу свободно>
        ТО вниз
      КОНЕЦ ЕСЛИ
      ЕСЛИ <справа свободно>
        ТО вправо
      КОНЕЦ ЕСЛИ
    КОНЕЦ ПОКА
```



1) 18 2) 24 3) 21 4) 27

Ответы для самопроверки

№ задания	Ответ
1	3
2	44
3	2
4	2
5	3

Элементы теории алгоритмов

B2, B13 Исполнители

Конспект

Исполнитель — любое устройство или живое существо, которое способно точно и однозначно выполнять заданные команды (действия, заданные алгоритмом).

Система команд исполнителя — совокупность команд, которые он способен выполнять.

Алгоритм — запись в той или иной форме (словесной, графической, на языке программирования) последовательности команд для исполнителя. Команды алгоритма должны соответствовать системе команд исполнителя.

Разбор типовых задач

Задача 1. Исполнитель Увеличитель имеет две команды:

1) **прибавить 3;**

2) **умножить на 2.**

Запишите порядковые номера команд в программе для преобразования числа 1 в число 44, содержащей не более 5 строк.

Пример: 22121 соответствует программе:

умножить на 2
умножить на 2
прибавить 3
умножить на 2
прибавить 3,

которая преобразует число 1 в 17.

Если таких программ возможно более одной, то запишите любую из них.

Решение

Решать такие задачи проще всего «с конца»: сначала записать число, которое надо получить (в данном случае 44), а затем пытаться от него прийти к исходному числу (1), используя «обратные» команды Исполнителя (в данном случае — **вычесть 3** и **разделить на 2**). При этом, поскольку программа должна содержать мало команд (всего 5), надо прийти к числу 1 как можно быстрее. Значит, надо по возможности использовать команду деления на 2, а к вычитанию прибегать, если разделить число нацело на 2 не удаётся. Итак:

Команда	Текущее число
	44
разделить на 2	22
разделить на 2	11
вычесть 3	8
разделить на 2	4

Потратив 4 команды, от числа 44 пришли к числу 4. Теперь нужно всего за одну команду получить из него единицу. Поэтому деление на 2 уже не подходит: потребуется две команды деления на 2. А вот команда вычитания тройки годится. В итоге полная «обратная» программа (получения исходного числа из конечного) будет иметь вид:

Команда	Текущее число
	44
разделить на 2	22
разделить на 2	11
вычесть 3	8
разделить на 2	4
вычесть 3	1

Теперь остаётся записать эту программу в обратном порядке (от числа 1 к числу 44), используя изначальные команды Исполнителя — **прибавить 3** и **умножить на 2**:

Команда	Текущее число
	1
прибавить 3	4
умножить на 2	8
прибавить 3	11
умножить на 2	22
умножить на 2	44

Если записать (как требуется в задаче) последовательность порядковых номеров этих команд, то получается программа: 12122.

Ответ: 12122.

Задача 2. Исполнитель Квадрат имеет две команды:

1) **прибавить 1;**

2) **возвести в квадрат.**

Запишите порядковые номера команд в программе для преобразования числа 3 в число 27, содержащей не более 5 строк.

Пример: 21211 соответствует программе:

возвести в квадрат
прибавить 1
возвести в квадрат
прибавить 1
прибавить 1,

которая преобразует число 3 в число 102.

Если таких программ возможно более одной, то запишите любую из них.

Решение

Задача в целом аналогична предыдущей, только вместо умножения здесь используется возведение в квадрат.

Задача решается «с конца»: из числа 27 следует получить число 3, используя «обратные» команды: **квадратный корень** и **вычесть 1**. Поскольку надо прийти к числу 3 как можно быстрее, надо по возможности использовать команду вычисления квадратного корня, а к вычитанию прибегать, если корень нацело не извлекается.

Команда	Текущее число
	27
вычесть 1	26
вычесть 1	25
квадратный корень	5
вычесть 1	4
вычесть 1	3

Остаётся записать эту программу в обратном порядке (от числа 3 к числу 27), используя изначальные команды Исполнителя — **прибавить 1** и **возвести в квадрат**:

Команда	Текущее число
	3
прибавь 1	4
прибавь 1	5
возведи в квадрат	25
прибавь 1	26
прибавь 1	27

Если записать (как требуется в задаче) последовательность порядковых номеров этих команд, то получается программа: 11211.

Ответ: 11211.

Задача 3. Исполнитель Пятачок имеет две команды:

1) **прибавить 1;**

2) **умножить на 5.**

Запишите порядковые номера команд в программе для преобразования числа 1 в число 76, содержащей не более 5 строк.

Пример: 21211 соответствует программе:

умножить на 5
прибавить 1
умножить на 5
прибавить 1
прибавить 1,

которая преобразует число 1 в число 32.

Если таких программ возможно более одной, то запишите любую из них.

Решение

Задача аналогична предыдущим. Решается «с конца»: из числа 76 нужно получить число 1, используя «обратные» команды: **вычесть 1** и **разделить на 5**. Поскольку надо прийти к числу 1 как можно быстрее, по возможности используется команда деления, а к вычитанию прибегается, если число на 5 нацело не делится.

Команда	Текущее число
	76
вычесть 1	75
разделить на 5	15
разделить на 5	3
вычесть 1	2
вычесть 1	1

Остаётся записать эту программу в обратном порядке (от числа 1 к числу 76), используя изначальные команды Исполнителя — **прибавить 1** и **умножить на 5**:

Команда	Текущее число
	1
прибавить 1	2
прибавить 1	3
умножить на 5	15
умножить на 5	75
прибавить 1	76

Если записать последовательность порядковых номеров этих команд, то получается программа: 11221.

Ответ: 11221.

Задача 4. Исполнитель Вычислитель имеет две команды:

1) **вычесть 1;**

2) **разделить на 3.**

Запишите порядковые номера команд в программе для преобразования числа 66 в число 2, содержащей не более 5 строк.

Пример: 1211 соответствует программе:

вычесть 1

разделить на 3

вычесть 1

вычесть 1,

которая преобразует число 22 в число 5.

Если таких программ возможно более одной, то запишите любую из них.

Решение

В этом случае, когда команды исполнителя уменьшают обрабатываемое число, решать задачу удобнее напрямую — пытаться сразу из числа 22 получить число 5. Поскольку надо прийти к числу 5 как можно быстрее, надо по возможности использовать команду деления, а к вычитанию прибегать, если деление на 3 нацело невозможно.

Команда	Текущее число
	22
вычесть 1	21
разделить на 3	7
вычесть 1	6
вычесть 1	5

Таким образом, программа для данного исполнителя содержит четыре команды, однако это тоже удовлетворяет условию задачи (не более 5 команд).

Если записать последовательность порядковых номеров этих команд, то получается программа: 1211.

Ответ: 1211.

В любом случае такие задачи лучше решать «в направлении уменьшения чисел» (в том числе с «обратными» командами исполнителя и последующим переписыванием полученной программы «наоборот»).

Задача 5*. Имеется исполнитель Кузнечик, который живёт на числовой оси. Система команд Кузнечика: «Вперед N» (Кузнечик прыгает вперёд на N единиц); «Назад M» (Кузнечик прыгает назад на M единиц). Переменные N и M могут принимать любые целые положительные значения. Известно, что Кузнечик выполнил программу из 50 команд, в которой команд «Назад 2» на 12 больше, чем команд «Вперед 3». Других команд в программе не было. На какую одну команду можно заменить эту программу, чтобы Кузнечик оказался в той же точке, что и после выполнения программы?

Решение

Исходная программа для Кузнечика включала в себя 50 команд «Назад 2» и «Вперед 3», из которых команд «Назад 2» на 12 больше, чем команд «Вперед 3».

Пусть количество команд «Вперед 3» обозначается как x, тогда команд «Назад 2» было $(x + 12)$.

Составляется уравнение: $x + (x + 12) = 50$. Отсюда $2x = 38$, тогда $x = 19$.

Следовательно, исходная программа состояла из 19 команд «Вперед 3» и 31 команды «Назад 2».

Каково будет общее перемещение Кузнечика по этой программе относительно исходной точки (её координата в условии задачи не оговорена, для простоты можно считать её равной нулю)? Очевидно, что конечная координата Кузнечика будет равна:

$19 \cdot 3 - 31 \cdot 2 = -5$.

Тогда команда, эквивалентная исходной программе, — «Назад 5».

Ответ: «Назад 5».

Задача 6*. Исполнитель Черепашка перемещается на экране компьютера, оставляя след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существуют две команды:

Вперед n (*n* — целое число) вызывающая передвижение черепашки на *n* шагов в направлении движения.

Направо m (*m* — целое число) вызывающая изменение направления движения на *m* градусов по часовой стрелке.

Запись Повтори 5 [Команда1 Команда2] означает, что последовательность команд в скобках повторится 5 раз.

Черепашке был дан для исполнения следующий алгоритм:

```
Повтори 5 [Вперед 10 Направо 72]
```

Какая фигура появится на экране?

1) Незамкнутая ломаная линия.
2) Правильный треугольник.
3) Квадрат.
4) Правильный пятиугольник.

Решение

Решение этой задачи выполняется путём ручной трассировки заданной для неё программы с одновременной прорисовкой получаемой траектории её движения:

Шаг цикла «Повтори»	1	2	3	4	5
Вид получаемой траектории					

Очевидно, что:

- угол поворота 72 градуса, повторенный 5 раз, даёт полный оборот вокруг своей оси $(72 \cdot 5 = 360)$;
- поскольку на каждом шаге цикла Черепашка проходит одно и то же расстояние, то она за 5 шагов цикла вернётся в исходную точку, изобразив при этом геометрическую фигуру с пятью равными сторонами — правильный пятиугольник.

Ответ: правильный пятиугольник (вариант ответа №4).

Задача 7*. Исполнитель Черепашка перемещается на экране компьютера, оставляя след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существуют две команды:

`Вперед n`, вызывающая передвижение Черепашки на *n* шагов в направлении движения.

`Направо m`, вызывающая изменение направления движения на *m* градусов по часовой стрелке.

(Вместо *n* и *m* должны стоять целые числа).

Запись:

```
Повтори 5 [Команда1 Команда2]
```

означает, что последовательность команд в квадратных скобках повторится 5 раз.

Какое число необходимо записать вместо *n* в следующем алгоритме:

```
Повтори 7 [Вперед 40 Направо n],
```

чтобы на экране появился правильный шестиугольник?

1) 30 2) 45 3) 50 4) 60

Решение

Эта задача обратна предыдущей.

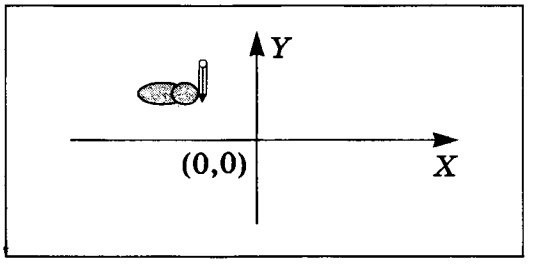
Что нужно, чтобы получить правильный шестиугольник? Необходимо отрисовать 6 одинаковых отрезков (т. е. достаточно шести шагов цикла; седьмой шаг цикла — это рисование повторно по уже нарисованному следу, на это можно не обращать внимание). Чему равен угол при вершине правильного шестиугольника? 360 / 6 = 60 градусов.

Следовательно, в команде `Направо` нужно записать значение 60.

Ответ: 60 (вариант ответа №4).

Задача 8. Имеется исполнитель типа «Черепашки ЛОГО», который может выполнять команды: «поднять перо», «опустить перо» и «перейти в точку с заданными координатами (*x*, *y*)». После выполнения команды текущие координаты точки сохраняются до следующей команды.

Имеется «виртуальный лист бумаги» — координатная плоскость с началом координат в середине листа, осями *X* и *Y* **и одинаковым масштабом этих осей.**



Для этого исполнителя была составлена программа на алгоритмическом языке, но в этой программе оказались пропущены координаты точек в командах рисования. Какие из предложенных координат нужно подставить в пропущенные места программы, чтобы исполнитель вычертил линию, похожую на окружность?

```
алг чертеж
нач
  вещ U, R, D
  цел I, N
U = 0
R = 1
N = 50
D = 2 · PI / N
поднять перо
```

```
перейти в точку (          )
опустить перо
нц для I от 1 до N
  U = U + D
  перейти в точку (          )
кц
кон
```

1) `R, R`
2) `R * R, U * U`
3) `R * cos(U), R * sin (U)`
4) `R * sin(U), R * cos(U)`

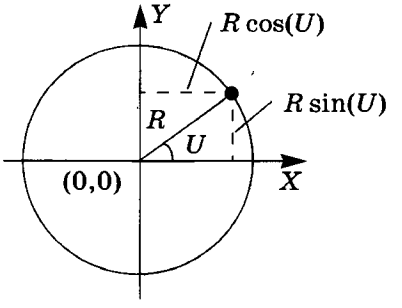
Решение

Анализируется приведённая программа. Поскольку в ней имеется цикл: `нц для I от 1 до N`, нетрудно догадаться, что поскольку цикловая переменная `I` нигде в теле цикла не используется, данный цикл служит лишь для отсчитывания `N` шагов отрисовки.

Далее, поскольку в цикле имеется команда `U = U + D`, становится понятно, что величина переменной `U` на каждом шаге цикла увеличивается и определяет очередную точку чертежа. А вспомнив, что ранее значение `D` задано как `D = 2 * PI / N`, а `U` — как `U = 0`, можно догадаться, что `U` — это угол (от 0 до 360 градусов), а `D` — приращение этого угла (вычисляемое делением всей окружности на заданное количество шагов цикла отрисовки).

Тогда команды `поднять перо`, `перейти в точку ( )` и `опустить перо`, стоящие до цикла, соответствуют выводу исполнителя из начала координат в начальную точку рисования, соответствующую углу 0 градусов, «вхолостую» (без черчения). А когда уже в цикле значение угла увеличивается на приращение `D` и вызывается команда `перейти в точку ( )`, то исполнитель (поскольку перо было опущено) вычерчивает отрезок от прежней точки до новой.

Остаётся выбрать, как должны рассчитываться координаты этих точек, лежащих на окружности. Вспоминаем геометрию:



Следовательно, координата x должна вычисляться как $R \cdot \cos(U)$, а координата y — как $R \cdot \sin(U)$.

Ответ: вариант 3.

Задача 9. Исполнитель Вычислитель имеет две команды:

1) прибавить 1;

2) умножить на 2.

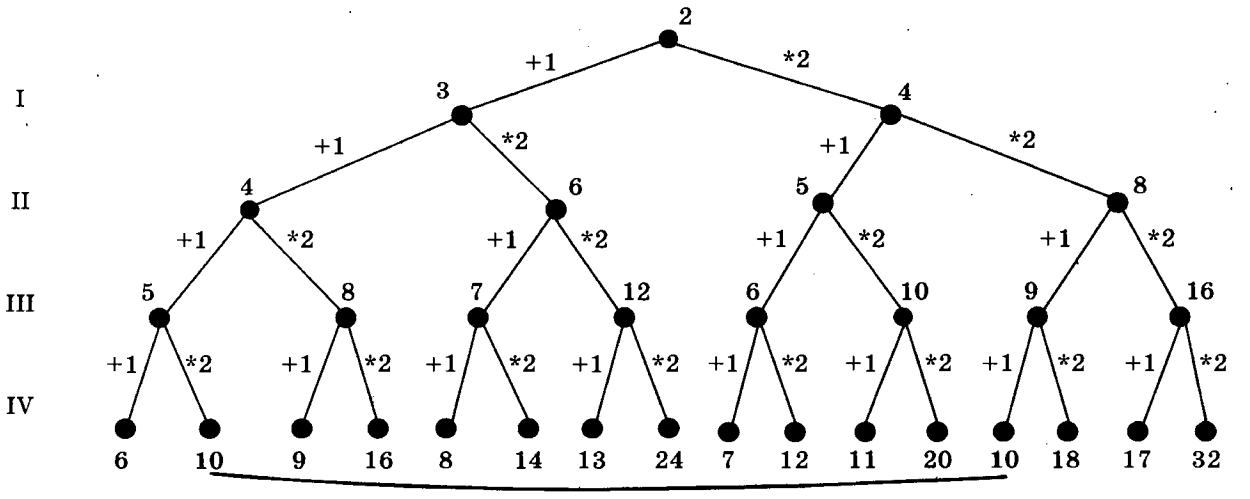
Сколько различных чисел можно получить из числа 2 с помощью программы, которая содержит ровно 4 команды?

Решение

В этой задаче нужно определить количество возможных вариантов. Поэтому лучше всего воспользоваться построением графа — дерева, вершинами которого являются числа (причём корневая вершина — это исходное число). Ветви этого дерева соответствуют воз-

можным каждый раз операциям (их две, поэтому от каждой вершины будет отходить ровно две ветви дерева). А количество таких ветвлений (глубина дерева) будет равно 4 (так как в задаче требуется узнать, сколько чисел можно получить с помощью программы ровно из четырёх команд).

Команды



Теперь остается только подсчитать количество различных (без учёта повторов!) чисел, соответствующих полученным концевым вершинам (для наглядности на рисунке выше два повторяющихся числа соединены линией). Таких различных чисел — 15.

Ответ: 15 различных чисел.

Этот способ решения является универсальным. Построенное дерево вариантов помогает узнать также и общее количество чисел (в том числе различных), которые позволяет получить программа для исполнителя, содержащая не более указанного количества команд: в этом случае нужно включать в подсчёт все вершины построенного дерева (а не только концевые), за исключением корневой.

Задача 10. Исполнитель Вычислитель имеет две команды:

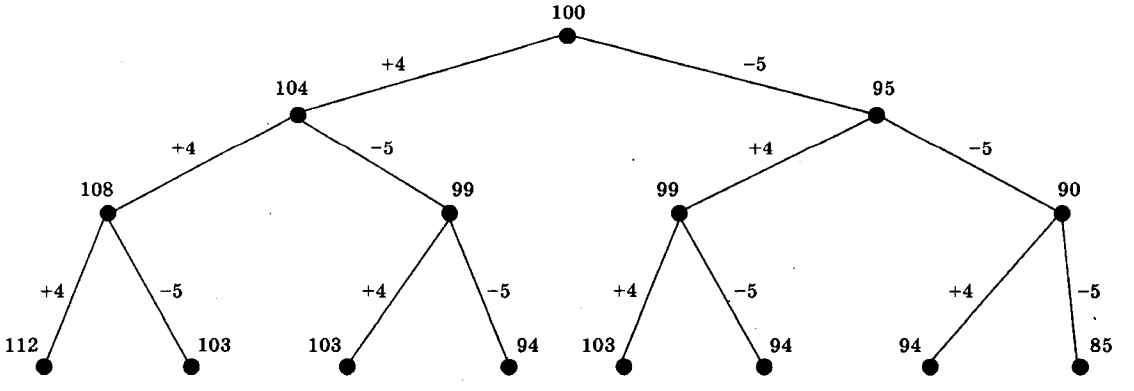
1) **прибавить 4;**

2) **вычесть 5.**

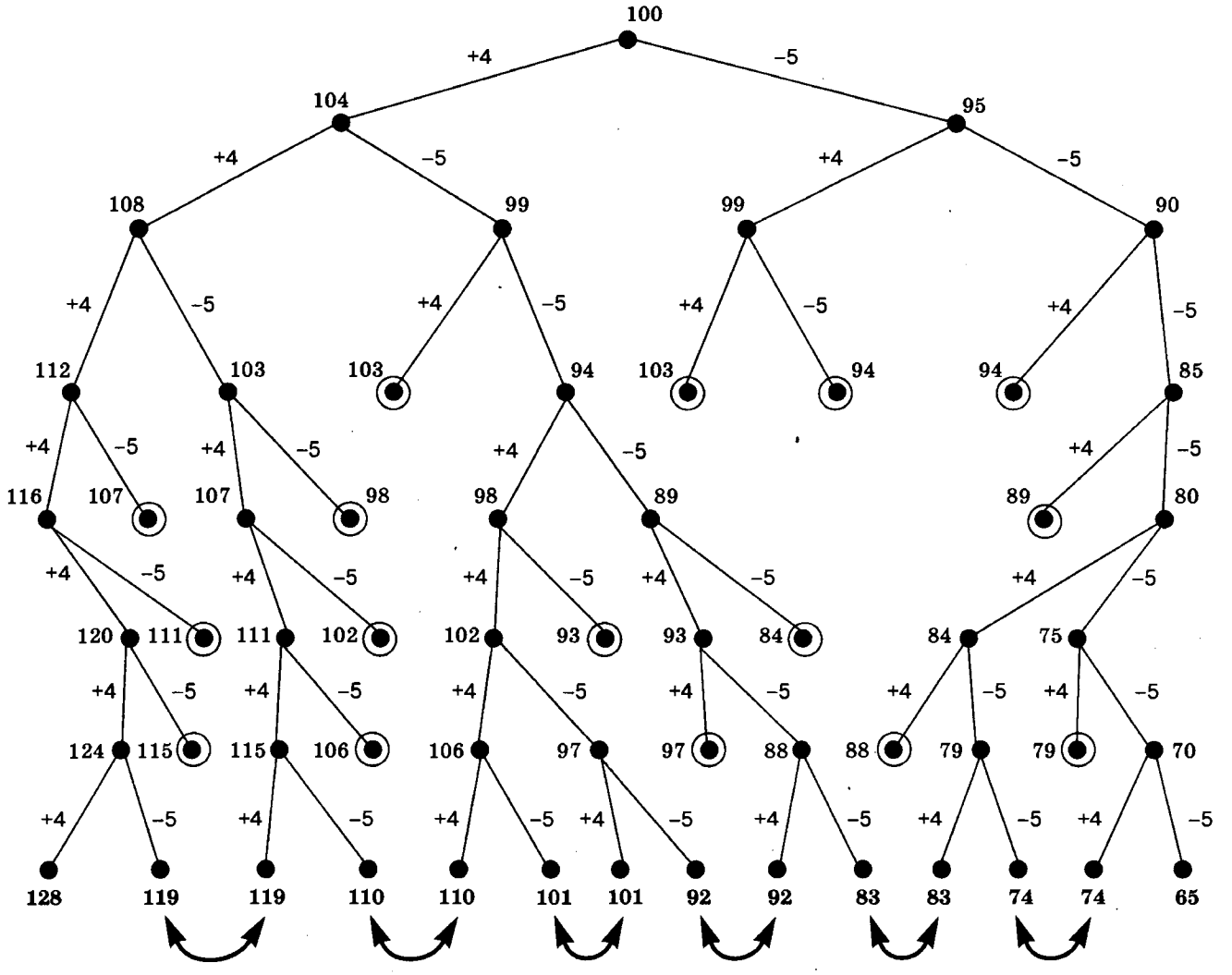
Сколько различных чисел можно получить из числа 100 с помощью программы, которая содержит ровно 7 команд?

Решение

Аналогично предыдущей задаче, составляется дерево вариантов. Корневая вершина в нём соответствует исходному числу 100, от каждой вершины отходит две ветви (соответствующих двум возможным операциям), глубина дерева равна 7 (количество команд в программах).



Из-за большого количества команд в программе исполнителя (7) дерево получается очень громоздким. Однако нетрудно заметить, что в его середине имеются повторяющиеся числа. Их можно исключать уже на этапе построения дерева, «обрубая» в нём лишние ветви и продолжая только одну ветвь из нескольких, соответствующих каждому повторяющемуся числу. Такие «обрубленные» повторяющиеся ветви можно обозначать двойным кружком.



С учётом исключения повторов количество различных чисел, получаемых после выполнения программы из семи команд, равно 8.

Ответ: 8 различных чисел.

Задача 11. Исполнитель Вычислитель имеет две команды:

1) умножить на 4;

2) поделить на 2.

Сколько различных чисел можно получить из числа 1024 с помощью программы, которая содержит ровно 10 команд?

Решение

Эта задача (равно как и все другие задачи данного типа) решается аналогично предыдущим. Однако из-за громоздкости получаемого дерева (в данном случае его глубина будет рав-

на 10) можно использовать упрощённое графическое представление множества вариантов в виде таблицы. Её смысл — тот же, что и у аналогичного дерева, но обозначения команд каждый раз не приводятся: подразумевается, что левое из двух получаемых значений всегда соответствует первой команде (в данном случае — умножению), а правое — второй команде (здесь — делению). Появляющиеся повторы чисел также исключаются по мере построения таблицы, при этом соседние одинаковые числа объединяются в одно число.

						1024						
					4096		512					
				16384		2048		256				
			65536		8192		1024		128			
		262144		32768		4096		512		64		

Чтобы сделать таблицу ещё более компактной, (в данном случае) можно заменить сами числа соответствующими степенями двоек, а операции умножения на 4 и деления на 2 — соответственно, на операции прибавления 2 и вычитания 1.

										10									
									12		9								
								14		11		8							
							16		13		10		7						
						18		15		12		9		6					
					20		17		14		11		8		5				
				22		19		16		13		10		7		4			
			24		21		18		15		12		9		6		3		
		26		23		20		17		14		11		8		5		2	
	28		25		22		19		16		13		10		7		4		1	
30		27		24		21		18		15		12		9		6		3		0

Вычислить сами числа, получаемые после выполнения программы из 10 команд, можно, возводя 2 в соответствующие степени. Однако это условием задачи не требуется! Следует определить только количество таких чисел, которое, очевидно, равно количеству полученных в таблице показателей степени двойки: 11 чисел.

Ответ: 11 чисел.

Приведённые ниже задачи, хотя и имеют другую формулировку, но также могут быть отнесены к рассматриваемой теме, поскольку описанный в них процесс выполняется персонажем задачи формально (т.е. персонаж задачи является исполнителем).

Задача 12*. Витя пригласил своего друга Сергея в гости, но не сказал ему код от цифрового замка своего подъезда, а послал следующее SMS-сообщение: «в последовательности чисел 3, 1, 8, 2, 6 все числа больше 5 разделить на 2, а затем удалить из полученной последовательности все чётные числа». Выполнив указанные в сообщении действия. Сергей получил следующий код для цифрового замка:

1) 3,1,1 2) 1, 1, 3 3) 3, 1, 3 4) 3, 3, 1

Решение

В данной задаче достаточно выполнить указания исполнителю.

Исходная последовательность чисел:

3	1	8	2	6

Все числа больше 5 делятся на 2:

3	1	4	2	3

Из полученной последовательности удаляются все чётные числа:

3	1	4	2	3

Полученная последовательность:

3	1	3

Ответ: 3, 1, 3 (вариант ответа №3).

Задача 13*. Лена забыла пароль для входа в Windows XP, но помнила алгоритм его получения из символов «A153B42FB4» в строке подсказки. Если последовательность символов «B4» заменить на «B52» и из получившейся строки удалить все трёхзначные числа, то полученная последовательность и будет паролем:

1) ABFB52
2) AB42FB52
3) ABFB4
4) AB52FB

Решение

Исходная последовательность символов: A153B42FB4

Последовательность символов «B4» заменяется на «B52»:

A153B42FB4 → A153B522F B52

Удаляются из полученной последовательности все трёхзначные числа: A~~153B522~~FB52

Полученная последовательность: ABFB52.

Ответ: ABFB52 (вариант ответа №1).

Задачи для самостоятельного решения

1. У исполнителя Плюсовой есть две команды:
   1. Прибавь 5.
   2. Вычти 2.

   Первая из них увеличивает число на экране на 5, вторая — уменьшает его на 2.

   Программа для Плюсовой — это последовательность команд. Сколько различных чисел можно получить из числа 95 с помощью различных программ, каждая из которых содержит ровно 6 команд?
2. У исполнителя Кузнечик две команды:
   1. Вычти 3.
   2. Прибавь 5.

   Первая из них уменьшает число на экране на 3, вторая — увеличивает его на 5 (отрицательные числа допускаются).

   Программа для Кузнечика — это последовательность команд. Сколько различных чисел можно получить из числа 1 с помощью программы, которая содержит ровно 6 команд?

3. У исполнителя Квадратик две команды, которым присвоены номера:
   1. Прибавь 1.
   2. Возведи в квадрат.
   Первая из них увеличивает число на экране на 1, вторая — возводит его в квадрат. Запишите порядок команд в программе получения из числа 6 числа 66 для исполнителя Квадратик, содержащей не более 5 команд, указывая лишь номера команд.
4. У исполнителя Увеличитель две команды, которым присвоены номера:
   1. Умножь на 3.
   2. Прибавь 2.
   Первая из них утраивает число на экране, вторая — увеличивает его на 2.
   Запишите порядок команд в программе преобразования числа 0 в число 72, содержащей не более 5 команд, указывая лишь номера команд.
5. У исполнителя имеется две команды:
   1. Умножь на 3.
   2. Вычти 7.
   Первая команда умножает число на 3, вторая — вычитает из него 7.
   Запишите порядок команд, необходимых для получения из 5 числа 17, количество команд не должно превышать 4.
6. Для составления 4-значных чисел используются цифры 1, 2, 3, 4, 5. При этом соблюдаются следующие правила:
   • на первом месте стоит одна из цифр 1, 2 или 3.
   • после каждой чётной цифры идёт нечётная, а после каждой нечётной — чётная.
   • третьей цифрой не может быть цифра 5. Какое из перечисленных чисел получено по этим правилам?
   1) 4325 3)1241
   2)1432 4)3452
7. Для составления цепочек используются разноцветные бусины: тёмные — синяя (С), зелёная (З) и светлые — жёлтая (Ж), белая (Б), голубая (Г). На первом месте в цепочке стоит бусина синего или жёлтого цвета. В середине цепочки — любая из светлых бусин, если первая бусина тёмная, и любая из тёмных бусин, если первая бусина светлая. На последнем месте — одна из бусин белого, голубого или зелёного цвета, не стоящая в цепочке в середине. Какая из перечисленных цепочек создана по этим правилам?
   1) ЖСГ 3) СГЖ
   2) БГЗ 4) ГЖБС
8. Исполнитель КАЛЬКУЛЯТОР имеет только две команды, которым присвоены номера:
   1. Вычти 3.
   2. Умножь на 2.
   Выполняя команду номер 1, КАЛЬКУЛЯТОР вычитает из числа на экране 3, а выполняя команду номер *2*, умножает число на экране на 2. Напишите программу, содержащую не более 5 команд; которая преобразует число 4 в число 17. В качестве ответа запишите подряд номера команд, составляющих такую программу.
9. Система команд исполнителя Сумматор включает две команды:
   1. Плюс 6.
   2. Плюс 3.
   Первая из них увеличивает текущее число на 6, а вторая увеличивает текущее число на 3.
   Сколько различных чисел можно получить из числа 0 с помощью последовательности не более чем из 8 команд?

10. У исполнителя Кузнечик две команды:

1. Прибавь 4.
2. Вычти 2.

Первая из них увеличивает число на экране на 4, вторая — уменьшает его на 2 (отрицательные числа допускаются).

Программа для Кузнечика — это последовательность команд. Сколько различных чисел можно получить из числа 1 с помощью программы, которая содержит не более 8 команд?

Ответы для самопроверки

№ задания	Ответ
1	7
2	7
3	11211
4	21211
5	1212
6	1432
7	ЖСК
8	21221
9	17
10	24

Элементы теории алгоритмов

С3 Исполнители

Конспект

Исполнитель — любое устройство или живое существо, которое способно точно и однозначно выполнять заданные команды (действия, заданные алгоритмом).

Система команд исполнителя — совокупность команд, которые он способен выполнять.

Алгоритм — запись в той или иной форме (словесной, графической, на языке программирования) последовательности команд для исполнителя. Команды алгоритма должны соответствовать системе команд исполнителя.

Разбор типовых задач

Задача 1*. У исполнителя Утроитель две команды, которым присвоены номера:

1. **Прибавь 1,**
2. **Умножь на 3.**

Первая из них увеличивает число на экране на 1, вторая — утраивает его.

Программа для Утроителя — это последовательность команд.

Сколько есть программ, которые число 1 преобразуют в число 29?

Ответ обоснуйте.

Решение[1]

Как и в предыдущих задачах, можно построить полное дерево вариантов для получения числа 1 из числа 29 (уже не «обрубая» повторяющиеся ветви!) и подсчитать количество возможных способов получения этого числа (т.е. количество ветвей, равное количеству конечных вершин с учётом всех повторяющихся значений).

Однако, хотя данный способ наиболее понятен и нагляден, экзаменаторы для получения максимального балла требуют запись аналитического решения задачи.

Пусть R(n) — количество программ, которые преобразуют число 1 в число n.

Для анализа решения лучше (как и в предыдущих задачах) рассматривать обратный процесс — получение числа 1 из числа 29 при помощи обратных команд «вычесть 1» и «делить на 3».

Тогда R(n) = R(n/3) + R(n − 1) {в общем случае очередное число может быть получено или делением на 3, или вычитанием единицы}.

Вернувшись к исходной задаче, расписывается каждый из шагов последовательности действий для получения числа 29 из числа 1.

R(1) = 1 {в любом случае исходное число — «в единственном экземпляре»}.

[1] Решение предложено Денисом Королевым (школа №1360).

R(2) = R(2/3) + R(2 − 1) = ~~R(2/3)~~ + R(2 − 1) = R(2 − 1) = R(1) = 1 {слагаемые, дающие нецелый результат, исключаются; для получаемого значения R(n) берётся ранее вычисленное значение}.

R(3) = R(3/3) + R(3 − 1) = R(1) + R(2) = 1 + 1 = 2.

R(4) = R(4/3) + R(4 − 1) = ~~R(4/3)~~ + R(4 − 1) = R(3) = 2.

R(5) = R(5/3) + R(5 − 1) = ~~R(5/3)~~ + R(5 − 1) = R(4) = 2.

R(6) = R(6/3) + R(6 − 1) = R(2) + R(5) = 1 + 2 = 3.

R(7) = R(7/3) + R(7 − 1) = ~~R(7/3)~~ + R(7 − 1) = R(6) = 3.

R(8) = R(8/3) + R(8 − 1) = ~~R(8/3)~~ + R(8 − 1) = R(7) = 3.

R(9) = R(9/3) + R(9 − 1) = R(3) + R(8) = 2 + 3 = 5.

R(10) = R(10/3) + R(10 − 1) = ~~R(10/3)~~ + R(10 − 1) = R(9) = 5.

R(11) = R(11/3) + R(11 − 1) = ~~R(11/3)~~ + R(11 − 1) = R(10) = 5.

R(12) = R(12/3) + R(12 − 1) = R(4) + R(11) = 2 + 5 = 7.

R(13) = R(13/3) + R(13 − 1) = ~~R(13/3)~~ + R(13 − 1) = R(12) = 7.

R(14) = R(14/3) + R(14 − 1) = ~~R(14/3)~~ + R(14 − 1) = R(13) = 7.

R(15) = R(15/3) + R(15 − 1) = R(5) + R(14) = 2 + 7 = 9.

R(16) = R(16/3) + R(16 − 1) = ~~R(16/3)~~ + R(16 − 1) = R(15) = 9.

R(17) = R(17/3) + R(17 − 1) = ~~R(17/3)~~ + R(17 − 1) = R(16) = 9.

R(18) = R(18/3) + R(18 − 1) = R(6) + R(17) = 3 + 9 = 12.

R(19) = R(19/3) + R(19 − 1) = ~~R(19/3)~~ + R(19 − 1) = R(18) = 12.

R(20) = R(20/3) + R(20 − 1) = ~~R(20/3)~~ + R(20 − 1) = R(19) = 12.

R(21) = R(21/3) + R(21 − 1) = R(7) + R(20) = 3 + 12 = 15.

R(22) = R(22/3) + R(22 − 1) = ~~R(22/3)~~ + R(22 − 1) = R(21) = 15.

R(23) = R(23/3) + R(23 − 1) = ~~R(23/3)~~ + R(23 − 1) = R(22) = 15.

R(24) = R(24/3) + R(24 − 1) = R(8) + R(23) = 3 + 15 = 18.

R(25) = R(25/3) + R(25 − 1) = ~~R(25/3)~~ + R(25 − 1) = R(24) = 18.

R(26) = R(26/3) + R(26 − 1) = ~~R(26/3)~~ + R(26 − 1) = R(25) = 18.

R(27) = R(27/3) + R(27 − 1) = R(9) + R(26) = 5 + 18 = 23.

R(28) = R(28/3) + R(28 − 1) = ~~R(28/3)~~ + R(28 − 1) = R(27) = 23.

R(29) = R(29/3) + R(29 − 1) = ~~R(29/3)~~ + R(29 − 1) = R(28) = 23.

Ответ: 23 программы.

Задача 2. У исполнителя Троитель — шестеритель две команды, которым присвоены номера:

1. **Прибавь 6,**
2. **Умножь на 3.**

Первая из них увеличивает число на экране на 6, вторая — утраивает его.

Программа для исполнителя — это последовательность команд.

Сколько есть программ, которые число 9 преобразуют в число 87?

Ответ обоснуйте.

Решение

Строится аналогично предыдущей задаче, но поскольку предполагается прибавление не 1, а 6, нужно записывать значения R(n) для n с шагом 6.

R(9) = 1 {в любом случае исходное число — «в единственном экземпляре»}.

R(9 + 6) = R(15) = R(15/3) + R(15 − 6) = ~~R(5)~~ + R(9) = R(9) = 1 {очередная строка записывается для значения n, увеличенного на 6; слагаемые, дающие нецелый результат, исключаются; для получаемого значения R(n) берётся ранее вычисленное значение; если значение R(n) не существует, то оно исключается}.

Если использовать шаг 1, то получается запись:
R(10) = R(10/3) + R(10 – 6) = R(4).
Но поскольку вычисления начинаются с числа 9, значение R(4) не существует, и данная запись бессмысленна. Аналогично, несуществующие значения R(n) получаются и в других случаях при попытке записи строк для n с шагом, отличающимся от 6.

R(21) = R(21/3) + R(21 – 6) = ~~R(7)~~ + R(15) = 1.

R(27) = R(27/3) + R(27 – 6) = R(9) + R(21) = 1 + 1 = 2.

R(33) = R(33/3) + R(33 – 6) = ~~R(11)~~ + R(27) = 2 {несуществующее значение R(11) также исключаются; далее аналогично}.

R(39) = R(39/3) + R(39 – 6) = ~~R(13)~~ + R(33) = 2.

R(45) = R(45/3) + R(45 – 6) = R(15) + R(39) = 1 + 2 = 3.

R(51) = R(51/3) + R(51 – 6) = ~~R(17)~~ + R(45) = 3.

R(57) = R(57/3) + R(57 – 6) = ~~R(19)~~ + R(51) = 3.

R(63) = R(63/3) + R(63 – 6) = R(21) + R(57) = 1 + 3 = 4.

R(69) = R(69/3) + R(69 – 6) = ~~R(23)~~ + R(63) = 4.

R(75) = R(75/3) + R(75 – 6) = ~~R(25)~~ + R(69) = 4.

R(81) = R(81/3) + R(81 – 6) = R(27) + R(75) = 2 + 4 = 6.

R(87) = R(87/3) + R(81 – 6) = ~~R(29)~~ + R(81) = 6.

Ответ: 6 программ.

Задача 3. У исполнителя Квадратик две команды, которым присвоены номера:
1. **Прибавь 2,**
2. **Возведи в квадрат.**
Первая из них увеличивает число на экране на 2, вторая — возводит в квадрат.
Программа для исполнителя — это последовательность команд.
Сколько есть программ, которые число 4 преобразуют в число 66?
Ответ обоснуйте.

Решение

Аналогично предыдущей задаче записываются строки, содержащие «обратные» операции вычитания пятёрки и вычисления корня квадратного. Кроме того, поскольку в исходной задаче прибавляется число 2, нужно записывать такие строки, начиная с исходного числа 4 и с шагом 2.

R(4) = 1 {в любом случае исходное число — «в единственном экземпляре»}.

R(4 + 2) = R(6) = R($\sqrt{6}$) + R(6 – 2) = ~~R($\sqrt{6}$)~~ + R(4) = R(4) = 1 {очередную строку записываем для значения n, увеличенного на 2; слагаемые, где квадратный корень извлекается не нацело, исключаются; для получаемого значения R(n) берется ранее вычисленное значение; если значение R(n) не существует, то оно исключается}.

R(8) = R($\sqrt{8}$) + R(8 – 2) = ~~R($\sqrt{8}$)~~ + R(6) = R(6) = 1.

R(10) = R($\sqrt{10}$) + R(10 – 2) = ~~R($\sqrt{10}$)~~ + R(8) = R(8) = 1.

R(12) = R($\sqrt{12}$) + R(12 – 2) = ~~R($\sqrt{12}$)~~~~R(@)~~ + R(10) = R(10) = 1.

R(14) = R($\sqrt{14}$) + R(14 – 2) = ~~R($\sqrt{14}$)~~ + R(12) = R(12) = 1.

R(16) = R($\sqrt{16}$) + R(16 – 2) = R(4) + R(14) = 1 + 1 = 2.

R(18) = R($\sqrt{18}$) + R(18 – 2) = ~~R($\sqrt{18}$)~~ + R(16) = R(16) = 2.

R(20) = R($\sqrt{20}$) + R(20 – 2) = ~~R($\sqrt{20}$)~~ + R(18) = R(18) = 2.

R(22) = R($\sqrt{22}$) + R(22 – 2) = ~~R($\sqrt{22}$)~~ + R(20) = R(20) = 2.

R(24) = R($\sqrt{24}$) + R(24 – 2) = ~~R($\sqrt{24}$)~~ + R(22) = R(22) = 2.

R(26) = R($\sqrt{26}$) + R(26 – 2) = ~~R($\sqrt{26}$)~~ + R(24) = R(24) = 2.

R(28) = R($\sqrt{28}$) + R(28 – 2) = ~~R($\sqrt{28}$)~~ + R(26) = R(26) = 2.

$R(30) = R(\sqrt{30}) + R(30 - 2) = \cancel{R(\sqrt{30})} + R(28) = R(28) = 2.$
$R(32) = R(\sqrt{32}) + R(32 - 2) = \cancel{R(\sqrt{32})} + R(30) = R(30) = 2.$
$R(34) = R(\sqrt{34}) + R(34 - 2) = \cancel{R(\sqrt{34})} + R(32) = R(32) = 2.$
$R(36) = R(\sqrt{36}) + R(36 - 2) = R(6) + R(34) = 1 + 2 = 3.$
$R(38) = R(\sqrt{38}) + R(38 - 2) = \cancel{R(\sqrt{38})} + R(36) = R(36) = 3.$
$R(40) = R(\sqrt{40}) + R(40 - 2) = \cancel{R(\sqrt{40})} + R(38) = R(38) = 3.$
$R(42) = R(\sqrt{42}) + R(42 - 2) = \cancel{R(\sqrt{42})} + R(40) = R(40) = 3.$
$R(44) = R(\sqrt{44}) + R(44 - 2) = \cancel{R(\sqrt{44})} + R(42) = R(42) = 3.$
$R(46) = R(\sqrt{46}) + R(46 - 2) = \cancel{R(\sqrt{46})} + R(44) = R(44) = 3.$
$R(48) = R(\sqrt{48}) + R(48 - 2) = \cancel{R(\sqrt{48})} + R(46) = R(46) = 3.$
$R(50) = R(\sqrt{50}) + R(50 - 2) = \cancel{R(\sqrt{50})} + R(48) = R(48) = 3.$
$R(52) = R(\sqrt{52}) + R(52 - 2) = \cancel{R(\sqrt{52})} + R(50) = R(50) = 3.$
$R(54) = R(\sqrt{54}) + R(54 - 2) = \cancel{R(\sqrt{54})} + R(52) = R(52) = 3.$
$R(56) = R(\sqrt{56}) + R(56 - 2) = \cancel{R(\sqrt{56})} + R(54) = R(54) = 3.$
$R(58) = R(\sqrt{58}) + R(58 - 2) = \cancel{R(\sqrt{58})} + R(56) = R(56) = 3.$
$R(60) = R(\sqrt{60}) + R(60 - 2) = \cancel{R(\sqrt{60})} + R(58) = R(58) = 3.$
$R(62) = R(\sqrt{62}) + R(62 - 2) = \cancel{R(\sqrt{62})} + R(60) = R(60) = 3.$
$R(64) = R(\sqrt{64}) + R(64 - 2) = R(8) + R(62) = 1 + 3 = 4.$
$R(66) = R(\sqrt{66}) + R(66 - 2) = \cancel{R(\sqrt{66})} + R(64) = R(64) = 4.$

Ответ: 4 программы.

Задачи для самостоятельного решения

1. У исполнителя Калькулятор две команды, которым присвоены номера:
   1. Умножь на 2
   2. Прибавь 5
   Первая команда умножает число на 2, вторая увеличивает его на 5.
   Программа для Калькулятора — это последовательность команд.
   Сколько есть программ, которые число 3 преобразуют в число 32?

2. Система команд исполнителя Вычислитель состоит из двух команд:
   1. Плюс 3.
   2. Умножить на 3.
   Первая команда увеличивает текущее число на 3, вторая — увеличивает текущее число в три раза.
   Программа для Вычислителя — это последовательность таких команд.
   Сколько существует программ, преобразующих число 3 в число 93?

3. Система команд исполнителя Вычислитель состоит из двух команд:
   1. Плюс 1.
   2. Умножить на 2.
   Первая команда увеличивает текущее число на 1, вторая — увеличивает текущее число в два раза.
   Программа для Вычислителя — это последовательность таких команд.
   Сколько существует программ, преобразующих число 1 в число 20?
   Ответ обоснуйте.

4. У исполнителя Квадратик две команды, которым присвоены номера:
   1. **Прибавь 4.**
   2. **Возведи в квадрат.**
   Первая из них увеличивает число на экране на 4, вторая — возводит в квадрат.
   Программа для исполнителя — это последовательность команд.
   Сколько есть программ, которые число 8 преобразуют в число 100?
   Ответ обоснуйте.
5. У исполнителя Увеличитель две команды, которым присвоены номера:
   1. **Прибавь 2.**
   2. **Умножь на 3.**
   Первая из них увеличивает число на экране на 2, вторая — умножает его на 3.
   Программа Увеличителя — это последовательность команд. Сколько есть прог
   которые число 1 преобразуют в число 59?

Ответы для самопроверки

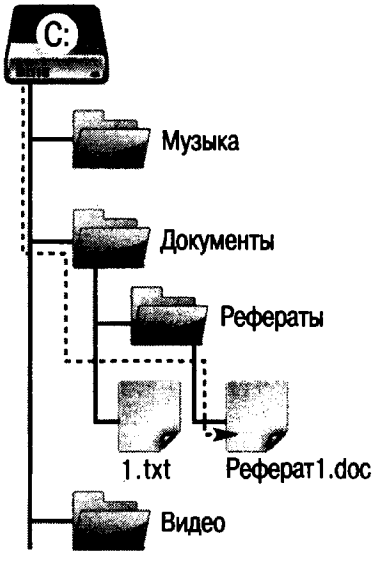
№ задания	Ответ
1	4
2	28
3	60
4	2
5	38

Архитектура компьютеров и компьютерных сетей

A4 Файловая система ПК

Конспект

Типовая древовидная структура файловой системы, принятой в ОС MS-DOS и Windows (используемой в задачах ЕГЭ):



Корневой каталог — метка диска

Каталог (папка) верхнего уровня

Каталог (папка) верхнего уровня

Вложенный каталог (папка)

Вложенные файлы

Каталог (папка) верхнего уровня

Путь к файлу — запись, начинающаяся меткой диска и содержащая имена всех папок, которые нужно одну за другой раскрыть, чтобы кратчайшим способом прийти к файлу.

Полное имя файла — запись пути к файлу, завершаемая именем и расширением файла.

В записи пути и полного имени файла метка диска, имена каталогов и имя файла разделяются символом обратной косой черты — «\».

Например, для файловой структуры, изображённой на рисунке выше:

- путь к файлу **Реферат1.doc** — **C:\Документы\Рефераты** (показан пунктирной стрелкой);
- полное имя файла **Реферат1.doc** — **C:\Документы\Рефераты\Реферат1.doc**.

Маска (шаблон) имени файла — запись, обозначающая группу файлов, имена которых отвечают заданным в этой маске требованиям. Маска обычно используется в качестве фильтра, чтобы выделить (или отобрать для выборочного показа в списке содержимого папки) файлы с нужными именами (и/или расширениями имени) и отсеять ненужные.

Символы-шаблоны — специальные символы-«джокеры», обозначающие один или несколько любых символов:

- символ «*» (звёздочка) — заменяет собой любое количество любых символов (в том числе нулевое количество — этих символов может не быть вовсе);

• символ «?» (знак вопроса) — заменяет один (и только один) обязательно стоящий в данном месте любой символ.

Маска может содержать как обычные символы (буквы, цифры и прочие знаки, допустимые в именах файлов), так и символы-шаблоны.

Примеры:

. — все файлы (т.е. файлы с любым именем и любым расширением);

*.doc — все файлы с любыми именами и расширением doc;

text??.txt — все файлы, имена которых начинаются с букв text и завершаются обязательно имеющимися двумя любыми символами, а расширение которых — txt (например, это могут быть файлы text01.txt, text22.txt, text_x.txt, textik.txt, но не text3.txt или textdoc.txt).

text**01**.txt
text**22**.txt
text**_x**.txt
text**ik**.txt
text??.txt
text**3**.txt
text**doc**.txt

Важное различие!

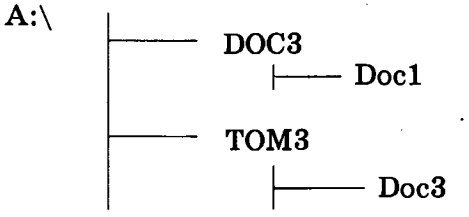
Символ «*» обозначает любое количество любых символов, в том числе нулевое (т.е. когда символов нет вообще).

Символ «?» обозначает один и только один любой символ; несколько символов «?» подряд обозначают ровно такое же количество любых символов (например, ??? — ровно три любых символа, не больше и не меньше).

Чтобы задать количество любых символов, не меньшее заданного, нужно использовать оба указанных символа-шаблона, когда символы «?» задают минимально допустимое число символов, а последующий символ «*» указывает, что символов может быть и больше. Например, маска ???* означает запись, содержащую не менее трёх любых символов (три обязательных — ??? и любое количество, в том числе нулевое, необязательных — *).

Разбор типовых задач

Задача 1*. Дано дерево каталогов.



Определите полное имя файла Doc3.

1) A:\DOC3
2) A:\DOC3\Doc3
3) A:\DOC3\Doc1
4) A:\TOM3\Doc3

Решение

Очевидно, что запись полного имени файла включает перечисляемые через знак «\» обозначения метки диска (A:) и каталогов, через которые нужно пройти по пути к файлу.

Ответ: A:\TOM3\Doc3 (вариант ответа №4).

Задача 2. В некотором каталоге хранился файл **Задача5**. После того, как в этом каталоге создали подкаталог и переместили в созданный подкаталог файл **Задача5**, полное имя файла стало **E:\Класс9\Физика\Задачник\Задача5**. Каково было полное имя этого файла до перемещения?

1) E:\Физика\Задачник\Задача5
2) E:\Физика\Задача5
3) E:\Класс9\Задачник\Задача5
4) E:\Класс9\Физика\Задача5

Решение

Запись полного имени файла после его перемещения в созданный подкаталог: **E:\Класс9\Физика\Задачник\Задача5.**

Тогда очевидно, что созданный подкаталог (тот, в котором теперь расположен файл), — **Задачник** (предпоследняя запись в строке полного имени).

Этот подкаталог располагается в каталоге **Физика**, в котором он и был создан. Но по условию задачи, файл первоначально хранился в том самом каталоге, в котором затем был создан новый подкаталог (куда после этого переместили файл). Следовательно, файл изначально находился в каталоге **Физика**.

Таким образом, нужно лишь исключить из записи полученного полного имени имя вновь созданного подкаталога **Задачник**. Тогда исходное полное имя файла: **E:\Класс9\Физика\Задача5.**

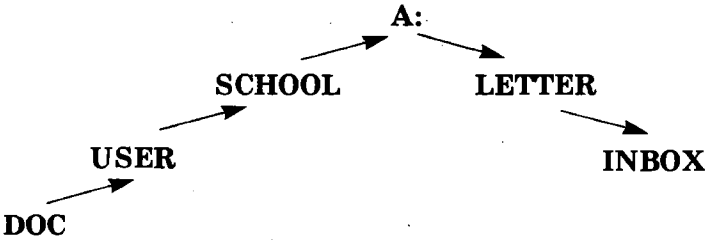
Ответ: E:\Класс9\Физика\Задача5 (вариант ответа №4).

Задача 3*. Перемещаясь из одного каталога в другой, пользователь последовательно посетил каталоги **DOC, USER, SCHOOL, A:\, LETTER, INBOX.** При каждом перемещении пользователь либо спускался в каталог на уровень ниже, либо поднимался на уровень выше. Каково полное имя каталога, из которого начал перемещение пользователь?

1) A:\DOC
2) A:\LETTER\INBOX
3) A:\SCHOOL\USER\DOC
4) A:\DOC\USER\SCHOOL

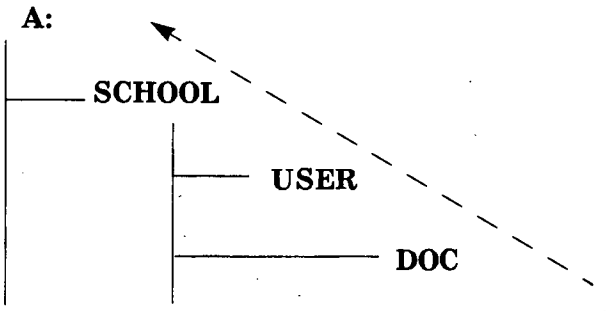
Решение

В середине пройденной цепочки имён каталогов располагается метка диска **A:**. Это — наивысший уровень иерархии в файловой системе. Следовательно, по пути к этой метке диска пользователь поднимался вверх уровень за уровнем (причём, поскольку в приведённой записи слева от **A:** имена каталогов не повторяются, этот подъём не прерывался спуском), а после прохождения этой метки диска — спускался вниз уровень за уровнем. Изобразим этот путь графически:



Левая ветвь такого рисунка строится от метки **A:** в записи последовательности пройденных каталогов сверху вниз в обратном порядке, а правая — от метки **A:** сверху вниз в прямом порядке.

Поскольку нужно полное имя исходного каталога, из полученного графического изображения пути берётся только левая «ветвь». Для неё можно изобразить следующую файловую структуру:



Теперь уже нетрудно записать полное имя исходного каталога: A:\SCHOOL\USER\DOC
Ответ: A:\SCHOOL\USER\DOC (вариант ответа №3).

Задача 4*. Для групповых операций с файлами используются **маски имён файлов**. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которых также могут встречаться следующие символы:

Символ «?» (вопросительный знак) означает ровно один произвольный символ.

Символ «*» (звёздочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность.

Определите, какое из указанных имён файлов удовлетворяет маске: **?a???***

1) dad1 2) dad22 3) 3daddy 4) add444

Решение

Запись **?a???*** означает, что ищутся файлы, в имени которых:

- символ «a» обязательно второй по счёту (перед ним обязательно стоит один любой символ — в маске он закодирован знаком «?»);
- после символа «a» обязательно идут три любых символа (в маске закодированы тремя знаками «?»), после которых может (но уже не обязательно) следовать ещё любое число символов.

Первому из указанных параметров соответствуют только файлы с именами **dad1** и **dad22** (в имени **3daddy** перед символом «a» стоят не один, а два символа, а в имени **add444** перед «a» нет обязательного какого-то символа).

Второму параметру из отобранных выше двух имён файлов соответствует только имя **dad22** (в имени **dad1** после символа «a» стоят не три, а только два обязательных символа).

Следовательно, указанной маске соответствует только имя файла **dad22**.

Ответ: dad22 (вариант ответа №2).

Задача 5*. Для групповых операций с файлами используются маски имён файлов. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которых также могут встречаться следующие символы:

Символ «?» (вопросительный знак) означает ровно один произвольный символ.

Символ «*» (звёздочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность.

Определите, по какой из масок будет выбрана указанная группа файлов:

1234.xls
23.xml
234.xls
23.xml

1) *23*.?x* 2) ?23?.x?? 3) ?23?.x* 4) *23*.???

Решение

Эта задача несколько усложнена по сравнению с предыдущей, но принцип её решения остаётся аналогичным. Нужно поочередно проверять каждую из предложенных масок (в вариантах ответа) на соответствие указанным именам файлов.

1. **Маска *23*.?x*.** Предполагает, что имя файла обязательно содержит цифры **23**, до и после которых может быть любое количество других символов (но их может и не быть!). В расширении же имени файла обязательно имеется символ «x», перед которым обязательно есть какой-то символ, а после него может (но необязательно) быть любое число символов.

Этой маске не соответствует ни один из заданных файлов, так как в расширениях их имён символ «x» стоит первым, а не вторым. Следовательно, данная маска не является решением задачи.

2. **Маска ?23?.x??.** Предполагает, что в имени файла перед и после цифр **23** обязательно есть по одному какому-то символу (знаки «?» в маске), а в расширении имени символ «x» обязательно стоит самым первым и после него обязательно есть ещё два каких-то символа.

Этой маске не соответствуют имена файлов **23.xml** и **234.xls**, так как в них не обеспечено наличие по одному символу до и после цифр **23**. Следовательно, данная маска также не является решением задачи.

3. **Маска ?23?.x*.** Предполагает, что в имени файла перед и после цифр **23** обязательно есть по одному какому-то символу (знаки «?» в маске), а в расширении имени символ «x» обязательно стоит самым первым и после него могут (но не обязательно) стоять какие-то другие символы.

Этой маске (как и предыдущей) не соответствуют имена файлов **23.xml** и **234.xls**, так как в них не обеспечено наличие по одному символу до и после цифр **23**. Следовательно, данная маска тоже не является решением задачи.

4. **Маска *23*.???.** Предполагает, что имя файла обязательно содержит цифры **23**, до и после которых может быть любое количество других символов (но их может и не быть!). В расширении имени обязательно должно быть три любых символа (не больше и не меньше).

Этой маске полностью соответствуют все заданные файлы. Следовательно, данная маска является решением задачи.

Ответ: маска *23*.??? (вариант ответа №4).

Задача 6. Для групповых операций с файлами используются маски имён файлов. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которых также могут встречаться следующие символы.

Символ «?» (вопросительный знак) означает ровно один произвольный символ.

Символ «*» (звёздочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность.

В каталоге находится 6 файлов:

motors.dat	torsten.docx
motors.doc	victoria.docx
storch.doc	x-torero.doc

Определите, по какой из масок из каталога будет отобрана указанная группа файлов:

motors.doc
storch.doc
victoria.docx
x-torero.doc

1) *tor?*.d* 2) ?tor*.doc 3) *?tor?*.do* 4) *tor?.doc*

Решение

Данная задача предполагает дополнительное усложнение: теперь нужно, анализируя каждую из предложенных масок, проверять не только то, что она позволяет выбрать все ука-

занные требуемые файлы, но и не приводит к ошибочному выбору каких-либо других файлов из заданного полного их списка.

1. **Маска *tor?*.d***. Предполагает, что имя файла обязательно содержит буквы **tor**, до которых не обязательно может быть любое количество символов, а после которых обязательно есть хотя бы один символ (знак «?» в маске). В расширении же имени файла обязательно первым стоит буква **d**, за которой может быть любое количество необязательных символов.

Этой маске соответствуют все имена файлов, которые должны быть отобраны по этой маске. Однако ей соответствуют и имена файлов **motors.dat** и **torsten.docx**, которые не должны быть отобраны. Следовательно, данная маска не является решением задачи.

2. **Маска ?tor*.doc**. Предполагает, что имя файла обязательно содержит буквы **tor**, перед которыми обязательно есть ровно один символ (знак «?» в маске) и после которых может быть любое количество необязательных символов. Расширение же имени файла обязательно должно быть **doc** и никакое другое.

Этой маске не соответствует имя файла **victoria.docx**, который должен быть отобран по этой маске. Следовательно, данная маска также не является решением задачи.

3. **Маска *?tor?*.do***. Предполагает, что имя файла обязательно содержит буквы **tor**, перед которыми обязательно есть хотя бы один символ (знак «?» в маске) и после которых тоже обязательно есть хотя бы один символ. Расширение же имени файла обязательно должно начинаться с букв **do**, после которых может быть любое число необязательных символов.

Этой маске соответствуют все имена файлов, которые должны быть отобраны по этой маске. Кроме того, ей не соответствуют имена файлов **motors.dat** (расширение начинается не с **do**, а с **da**) и **torsten.docx** (здесь в имени файла перед **tor** нет обязательного другого символа), которые не должны быть отобраны. Следовательно, данная маска является решением задачи.

4. **Маска *tor?.doc***. Предполагает, что имя файла обязательно содержит буквы **tor**, перед которыми может быть несколько необязательных символов и после которых обязательно есть ровно один символ (не больше и не меньше). Расширение же имени файла обязательно должно начинаться с букв **doc**, после которых может быть любое число необязательных символов.

Этой маске из файлов, которые должны быть отобраны по ней, соответствует только имя **motors.doc**, а имена **storch.doc**, **victoria.docx** и **x-torero.doc** не соответствуют этой маске (в них после **tor** имеется более одного символа). Следовательно, данная маска не является решением задачи.

Значит, решением может быть только маска ***?tor?*.do***.

Ответ: маска *?tor?*.do* (вариант ответа №3).

✎ Задачи для самостоятельного решения

1. В некотором каталоге хранился файл **Дневник.txt**. После того, как в этом каталоге создали подкаталог и переместили в созданный подкаталог файл **Дневник.txt**, полное имя файла стало **A:\SCHOOL\USER\TXT\MAY\Дневник.txt**. Каково полное имя каталога, в котором хранился файл до перемещения?

   1) MAY
   2) A:\SCHOOL\USER\TXT
   3) TXT
   4) A:\SCHOOL\USER\TXT\MAY

2. Для групповых операций с файлами используются маски имён файлов. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которых также могут встречаться следующие символы:

   Символ «?» (вопросительный знак) означает ровно один произвольный символ.

   Символ «*» (звёздочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность.

   Определите, какое из указанных имён файлов удовлетворяет маске: ?hel*lo.c?*.

   1) hello.c
   2) hello.cpp
   3) hhelolo.cpp
   4) hhelolo.c

3. Для групповых операций с файлами используются маски имён файлов. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которых также могут встречаться следующие символы:
Символ «?» (вопросительный знак) означает ровно один произвольный символ.
Символ «*» (звёздочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последов ательность.
Определите, какое из указанных имен файлов удовлетворяет маске: ?ba*r.?xt
1) bar.txt 2) obar.txt 3) obar.xt 4) barr.txt

4. Для групповых операций с файлами используются маски имён файлов. Маска представляет собой последовательность букв, цифр и прочих допустимых в имёнах файлов символов, в которых также могут встречаться следующие символы:
Символ «?» (вопросительный знак) означает ровно один произвольный символ.
Символ «*» (звёздочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность.
В каталоге находится 6 файлов:

final, mpeg	marine.mpg
fine.mdb	pinoccio.mp3
fine.mp3	tinatin.mpg

Определите, по какой из масок из каталога будет отобрана указанная группа файлов:
final.mpeg
fine.mp3
pinoccio.mp3
tinatin.mpg
1) ?in*.mp* 2) ?in*.m* 3) *in*.mp* 4) *in?.mp*

5. Для групповых операций с файлами используются маски имён файлов. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которых также могут встречаться следующие символы:
Символ «?» (вопросительный знак) означает ровно один произвольный символ.
Символ «*» (звёздочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность.
В каталоге находятся пять файлов:
kantek.doc
partner.dos
barter.mos
gadget.doc
yparser.dos
Определите, по какой из масок из них будет отобрана указанная группа файлов:
kantek.doc
yparser.dos
gadget.doc
1) ?a??e?.do? 2) ?a*e?.do? 3) *a??e?.do? 4) *a*e?.do?

Ответы для самопроверки

№ задания	Ответ
1	2
2	3
3	2
4	1
5	3

Архитектура компьютеров и компьютерных сетей

B11 Основные принципы функционирования сети Интернет. Протокол TCP/IP

Конспект

IP-адрес — уникальный (не повторяющийся) цифровой индивидуальный номер компьютера в сети. IP-адрес может быть закреплён за данным компьютером постоянно (**выделенный IP-адрес**) или выдаваться компьютеру временно из некоторого имеющегося набора только на время данного сеанса работы в сети.

В стандарте IP v.4 (рассматриваемом в задачах ЕГЭ) IP-адрес представляет собой запись из четырёх разделённых точками чисел, каждое из которых соответствует одному байту и имеет значение от 0 до 255. Пример:

168.154.0.177

Некоторые IP-адреса являются служебными и используются для специальных целей:
- адрес сети, в котором один или несколько крайних справа байтов равны нулю (примеры: 168.154.15.0, 168.154.0.0);
- широковещательный адрес, в котором один или несколько крайних справа байтов равны 255 (примеры: 168.154.15.255, 168.154.255.255).

Адрес сети, адрес компьютера в сети, маска IP-адреса

Если компьютер находится в составе локальной сети, которая, в свою очередь, подключена к сети Интернет, то возникает задача — определить IP-адрес всей локальной сети в целом (например, чтобы разослать пакет информации на все компьютеры данной локальной сети) и адрес конкретного компьютера внутри этой локальной сети.

Адрес сети и адрес компьютера в этой сети — это две условно выделяемые части единого IP-адреса данного компьютера, под которым он числится в Интернете:

адрес сети *адрес компьютера в этой сети*

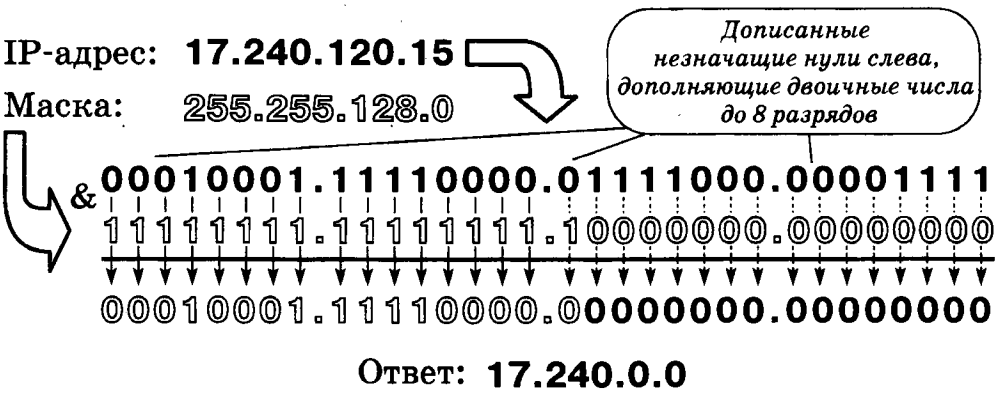
Для разделения единого IP-адреса на адрес локальной сети и адрес компьютера в этой сети используется **маска IP-адреса**. Она имеет такой же вид, как IP-адрес (четыре числа-байта, записанных через точку) и выполняет функцию фильтра.

Принцип использования маски:
- исходный IP-адрес записывается в двоичной форме (каждое число — байт переводится в восьмиразрядное двоичное число отдельно и по-прежнему записывается через точку);
- маска также записывается в двоичной форме;
- двоичная запись маски записывается под двоичной записью IP-адреса;
- для определения адреса сети нужно выполнить логическую операцию И для каждой отдельной пары битов: если в маске в данном разряде стоит 1, то в качестве результата

в данном разряде переписывается значение (0 или 1) из одноименного разряда исходного IP-адреса; если в маске в данном разряде стоит 0, то в качестве результата в данном разряде тоже записывается 0;

- полученная двоичная запись преобразуется в десятичную форму (каждое двоичное число — отдельно).

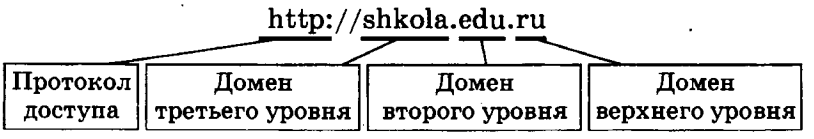
Пример:



Ответ: 17.240.0.0

При переводе десятичных значений чисел — составляющих IP-адреса в двоичный формат нужно помнить: получаемые двоичные числа обязательно должны быть восьмиразрядными! При необходимости надо дополнять полученное двоичное число до 8 разрядов, дописывая к нему слева незначащие нули.

Структура словесного адреса интернет-ресурса (URL)



Разбор типовых задач

Задача 1*. Идентификатор некоторого ресурса сети Интернет имеет следующий вид: http://www.ftp.ru/index.html

Какая часть этого идентификатора указывает на протокол, используемый для передачи ресурса?

1) www 2) ftp 3) http 4) html

Решение

Эта задача нацелена на проверку теоретических знаний.

Протокол, используемый для передачи ресурса, записывается в самом начале адреса, до символов «://».

Ответ: http (вариант ответа №3).

Задача 2*. Доступ к файлу **http.txt**, находящемуся на сервере www.net осуществляется по протоколу **ftp**. В таблице фрагменты адреса файла закодированы буквами от А до Ж. Запишите последовательность этих букв, кодирующую адрес указанного файла.

А	://
Б	http
В	ftp
Г	.net
Д	.txt
Е	/
Ж	www

Решение

Первым записывается протокол: ftp

После этого записываются символы «://»: ftp://

Далее в адресе записывается адрес сервера: ftp://www.net

После этого записывается разделяющий знак «/» и далее — имя файла: ftp://www.net/http.txt.

Если закодировать полученную запись согласно приведённой таблице, то получается следующая строка букв: ВАЖГЕБД.

Ответ: ВАЖГЕБД.

Задача 3*. Петя записал IP-адрес школьного сервера на листке бумаги и положил его в карман куртки. Петина мама случайно постирала куртку вместе с запиской. После стирки Петя обнаружил в кармане четыре обрывка с фрагментами IP-адреса. Эти фрагменты обозначены буквами А, Б, В и Г. Восстановите IP-адрес.

В ответе укажите последовательность букв, обозначающих фрагменты, в порядке, соответствующем IP-адресу.

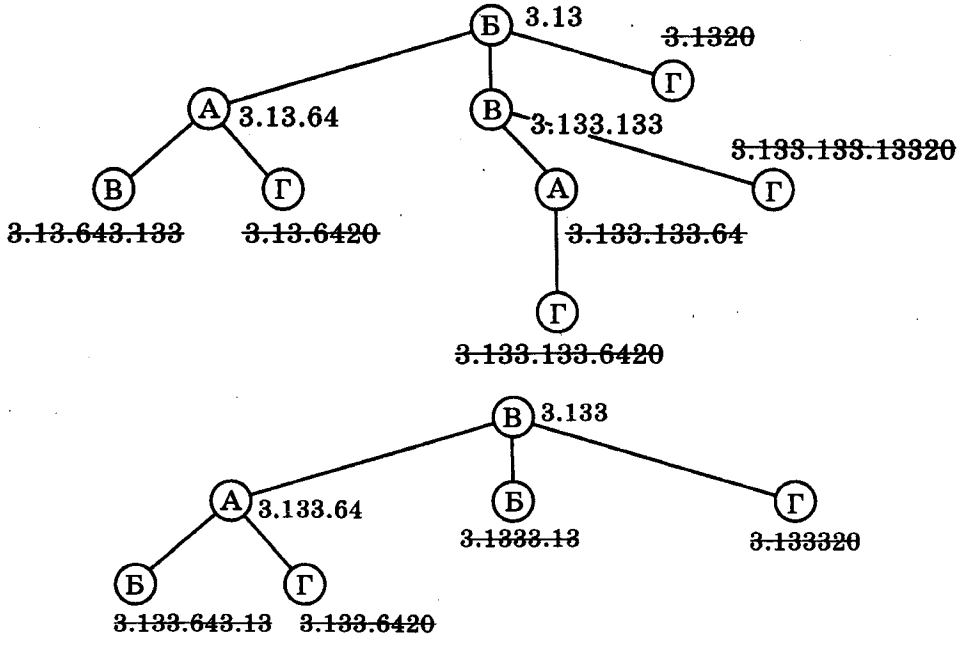
.64	3.13	3.13	20
А	Б	В	Г

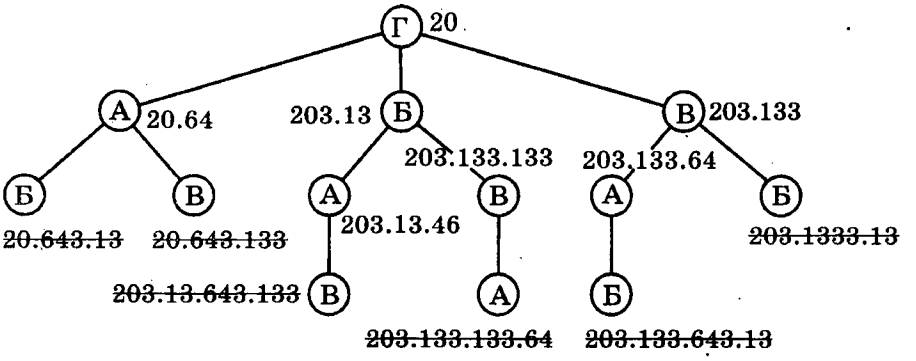
Решение

При решении таких задач нужно помнить следующее:

- ни одно число в записи IP-адреса не должно быть больше 255;
- IP-адрес состоит из четырёх чисел, разделённых точками.

С учётом этих ограничений нужно выполнить перебор всех возможных комбинаций предложенных фрагментов. В общем случае для этого может потребоваться построить 4 дерева вариантов, каждое из которых начинается с одного из этих фрагментов. Однако реально их количество будет меньше, так как с некоторых фрагментов IP-адрес заведомо начинаться не может (в данной задаче это фрагмент А: IP-адрес не может начинаться с точки). Если при построении такого дерева в какой-то ветви получается недопустимая запись IP-адреса, то дальнейшее построение этой ветви прекращается (ниже такие недопустимые IP-адреса показаны зачёркнутыми).





Таким образом, единственный допустимый вариант — это IP-адрес 203.133.133.64 (ГБВА).

Ответ: ГБВА.

Задача 4*. По заданному IP-адресу и маске определить адрес сети.

IP-адрес: 17.240.120.15

Маска: 255.255.128.0

Решение

Маской IP-адреса называют число (так же, как и IP-адрес, состоящее из четырёх записанных через точку байтовых значений от 0 до 255 каждое), которое определяет, какую часть IP-адреса составляет адрес сети. Для этого нужно представить и исходный IP-адрес, и маску как набор двоичных чисел, а затем поразрядно выполнить их конъюнкцию: если в каком-либо разряде маски записана «1», то значение соответствующего разряда исходного IP-адреса переписывается без изменения, а если в маске записан «0», то соответствующий разряд IP-адреса обнуляется (можно сказать, что «0» в маске «гасит» до 0 соответствующий разряд исходного IP-адреса). Полученная в результате запись и будет указывать IP-адрес сети (каждое из её двоичных чисел можно вновь перевести в десятичную систему счисления).

Эта операция для заданных значений IP-адреса и маски:

IP-адрес: 17.240.120.15 → 00010001.11110000.01111000.00001111

Маска: 255.255.128.0 → & 11111111.11111111.10000000.00000000

00010001.11110000.00000000.00000000

↓

17.240.0.0

Ответ: 17.240.0.0

Задача 5. В терминологии сетей TCP/IP маской подсети называется 32-разрядное двоичное число, определяющее, какие именно разряды IP-адреса компьютера являются общими для всей подсети, — в этих разрядах маски стоит 1. Обычно маски записываются в виде четвёрки десятичных чисел — по тем же правилам, что и IP-адреса. Для некоторой подсети используется маска 255.255.254.0. Сколько различных адресов компьютеров допускает эта маска?

Примечание. На практике для адресации компьютеров не используются два адреса: адрес сети и широковещательный адрес.

Решение

Если дан некий конкретный IP-адрес и дана маска подсети, то нужно, преобразовав их десятичную запись (в виде 4 чисел от 0 до 255, записанных через точку) в двоичную (четыре 8-разрядных числа, также записанных через точку), записать IP-адрес, под ним — маску, а затем выполнить для них поразрядно следующие операции:

— для получения адреса подсети — для каждого разряда маски, равного 1, просто переписать значение соответствующего ему разряда исходного IP-адреса, а для каждого разряда

маски, равного 0, записать нулевой разряд в получаемой записи (т.е. разряды исходного IP-адреса «проходят» неизменными «сквозь» единичные биты маски, но гасятся в 0 нулевыми битами маски);

— для получения адреса компьютера в пределах этой подсети надо вначале инвертировать маску (заменить к ней 0 на 1, а 1 на 0), а затем выполнить те же самые действия с исходным IP-адресом и инвертированной маской.

Полученную двоичную запись адреса подсети (в первом случае) или адреса компьютера в пределах подсети (во втором случае) затем надо снова перевести в набор записанных через точку четырёх десятичных чисел.

В рассматриваемой задаче вопрос поставлен несколько иначе. Здесь по заданной маске нужно определить, какое количество различных адресов компьютеров данная маска образует в пределах подсети. Для этого:

1) записывается маска в двоичном виде:

255.255.254.0 ⇒ 11111111.11111111.11111110.00000000

2) инвертируется полученная запись маски, получается:

00000000.00000000.00000001.11111111

3) данную запись маски можно представить в следующем виде:

Здесь серым фоном выделена та часть IP-адресов компьютеров подсети, которая не может изменяться (ведь, по условию, все эти компьютеры должны располагаться в одной и той же подсети!), а белыми оставлены те биты IP-адресов, которые могут меняться;

4) тогда количество теоретически возможных адресов компьютеров можно определить, зная, что в них биты, соответствующие «белым» битам маски, могут формировать все возможные комбинации нулей и единиц — от «0.00000000» до «1.11111111». Но все возможные такие комбинации — это не что иное, как все возможные 9-разрядные двоичные числа от 0 до 111111111, и очевидно, что таких чисел будет $1000000000_2 = 2^9 = 512$;

5) согласно примечанию к условию задачи, из этого теоретического множества в качестве IP-адресов отдельных компьютеров нельзя использовать два адреса:

— адрес сети, в котором все указанные изменяемые биты равны нулю;

— широковещательный адрес, в котором все указанные изменяемые биты равны единице (отправка пакета информации на такой адрес приводит к рассылке пакета всем компьютерам данной подсети — поэтому он и называется «широковещательным»).

Впрочем, для решения данной задачи не так важны эти объяснения, как информация о том, что из 512 найденных нами IP-адресов для отдельных компьютеров нашей подсети можно использовать на 2 адреса меньше, т. е. всего 510 адресов.

Ответ: 510 адресов.

Задача 6*. В терминологии сетей TCP/IP маской подсети называется 32-разрядное двоичное число, определяющее, какие именно разряды IP-адреса компьютера являются общими для всей подсети — в этих разрядах маски стоит 1. Обычно маски записываются в виде четвёрки десятичных чисел — по тем же правилам, что и IP-адреса.

Для некоторой подсети используется маска 255.255.252.0. Сколько различных адресов компьютеров теоретически допускает эта маска?

Примечание. На практике используются не все из этих адресов. Например, как правило, не используются IP-адреса, в десятичном представлении которых последнее (самое правое) число равно 0.

Решение

Аналогично только что рассмотренной задаче, можно по двоичной записи маски:

11111111.11111111.11111100.00000000

определить, что теоретически возможных IP-адресов компьютеров в пределах определяемой этой маской подсети будет $10000000000_2 = 2^{10} = 1024$.

В примечании к задаче сказано, что на практике не используются адреса, в десятичном представлении которых последнее число равно 0. Очевидно, что таких адресов будет четыре (запишем только последние, «изменяемые» биты):

00.00000000
01.00000000
10.00000000
11.00000000

По аналогии с предыдущей задачей «хочется» уменьшить найденное теоретическое количество адресов на 4 и записать в ответе число 1020. **Внимание!** — само это примечание добавлено, только чтобы запутать! Ведь в вопросе к задаче как раз спрашивается именно теоретически возможное количество адресов, а не используемое на практике! Так что правильный ответ будет — именно 1024.

Ответ: 1024 адреса.

Задача 7. В терминологии сетей TCP/IP маской сети называется двоичное число, определяющее, какая часть IP-адреса узла сети относится к адресу сети, а какая — к адресу самого узла в этой сети. Обычно маска записывается по тем же правилам, что и IP-адрес. Адрес сети получается в результате применения поразрядной конъюнкции к заданным IP-адресу узла и маске.

По заданным IP-адресу узла и маске определите адрес сети.

IP-адрес узла: 224.23.252.131

Маска: 255.255.240.0

При записи ответа выберите из приведённых в таблице чисел четыре элемента IP-адреса и запишите в нужном порядке соответствующие им буквы без использования точек.

A	B	C	D	E	F	G	H
255	240	252	224	131	23	8	0

Пример.

Пусть искомый IP-адрес 192.168.128.0 и дана таблица

A	B	C	D	E	F	G	H
128	168	255	8	127	0	17	192

В этом случае правильный ответ будет записан в виде HBAF.

Решение

Такие задачи решаются аналогично предыдущим, но в ответе нужно записать не сам получаемый результат, а закодировать его буквами в соответствии с таблицей.

1. Двоичная запись IP-адреса узла:

224.23.252.131 →

11100000.00010111.11111100.10000011

Следует не забывать дополнять каждое получаемое двоичное число — компонент IP-адреса до 8 разрядов, дописывая незначащие нули слева!

2. Двоичная запись маски:

255.255.240.0 →

11111111.11111111.11110000.00000000

3. Для определения адреса сети подписывается маска под записью IP-адреса и применяется к каждому двоичному разряду логическая операция И (конъюнкция):

```
  11100000.00010111.11111100.10000011
& 11111111.11111111.11110000.00000000
  11100000.00010111.11110000.00000000
```

Смысл операции: в разряде, где в маске стоит 0, записывается 0, а в разряде, где в маске стоит 1, переписывается значение из одноименного разряда исходного IP-адреса.

4. Полученная двоичная запись IP-адреса преобразуется в десятичный формат (отдельно каждое 8-разрядное число между точками):

```
11100000.00010111.11110000.00000000
224.23.240.0
```

5. Кодируется полученная запись в соответствии с заданной таблицей:

B	D	F	H
240	224	23	0

224.23.240.0 → DFBH

Ответ: DFBH.

Задачи для самостоятельного решения

1. Коля записал IP-адрес школьного сервера на листке бумаги и положил его в карман куртки. Петина мама случайно постирала куртку вместе с запиской. После стирки Коля обнаружил в кармане четыре обрывка с фрагментами IP-адреса. Эти фрагменты обозначены буквами А, Б, В и Г. Восстановите IP-адрес.
   В ответе укажите последовательность букв, обозначающих фрагменты, в порядке, соответствующем IP-адресу.

5.20	12	.71	8.212
А	Б	В	Г

2. Доступ к файлу not.htm, находящемуся на сервере kniga.ru, осуществляется по протоколу http. В таблице фрагменты адреса файла закодированы буквами от А до Ж. Запишите последовательность этих букв, кодирующую адрес указанного файла в сети Интернет.

А) /	В) not	Д) http	Ж) .ru
Б) ://	Г) kniga	Е) .htm	

3. В терминологии сетей TCP/IP маской подсети называется 32-разрядное двоичное число, определяющее, какие именно разряды IP-адреса компьютера являются общими для всей подсети — в этих разрядах маски стоит 1. Обычно маски записываются в виде четвёрки десятичных чисел — по тем же правилам, что и IP-адреса. Для некоторой подсети используется маска 255.255.248.0. Сколько различных адресов компьютеров допускает эта маска? *Примечание.* На практике для адресации компьютеров не используются два адреса: адрес сети и широковещательный адрес.

4. По заданным IP-адресу узла и маске определите адрес сети.
IP-адрес узла: 220.125.134.215
Маска: 255.255.240.0
Запишите ответ как последовательность буквенных обозначений элементов IP-адреса согласно таблице (без учёта точек).

A	B	C	D	E	F	G	H
0	128	134	215	220	125	240	255

5. По заданным IP-адресу узла и маске определите адрес сети.
IP-адрес узла: 61.161.220.187
Маска: 255.255.255.224
Запишите ответ как последовательность буквенных обозначений элементов IP-адреса согласно таблице (без учёта точек).

A	B	C	D	E	F	G	H
0	61	160	187	224	220	161	255

Ответы для самопроверки

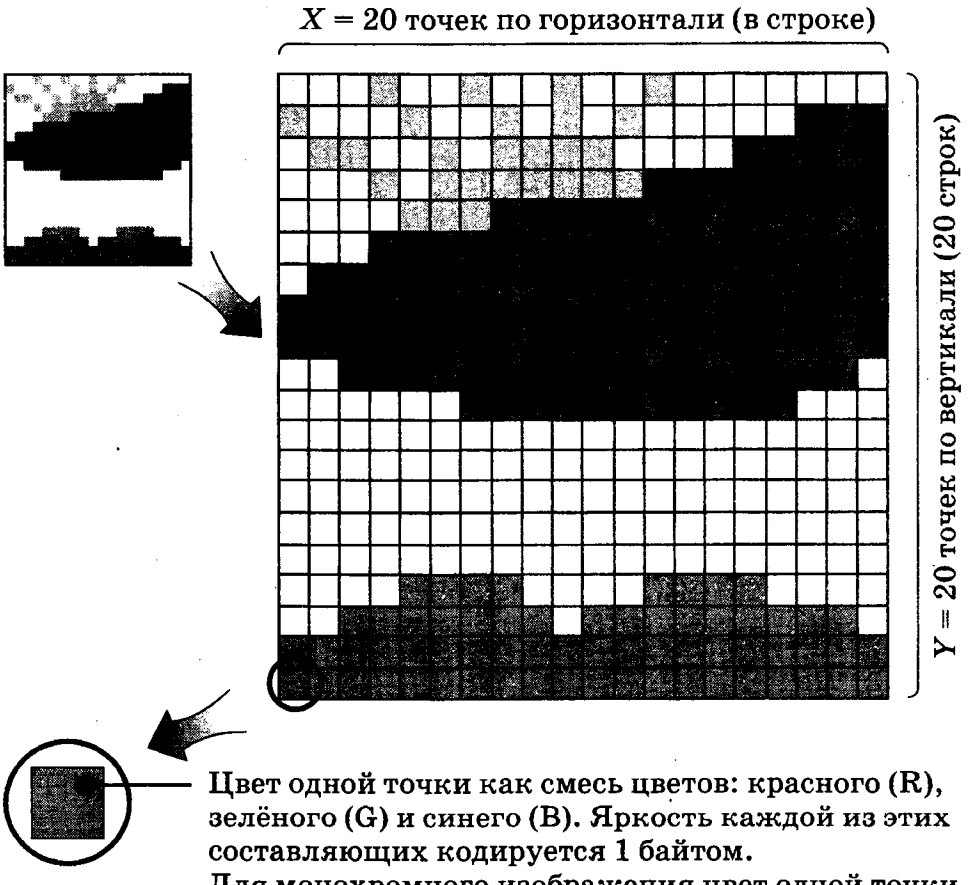
№ задания	Ответ
1	БАГВ
2	ДБГЖАВЕ
3	2046
4	EFBA
5	BGFC

Технология обработки звуковой и графической информации

A8 Определение объёма и скорости передачи цифровой мультимедиа-информации

Конспект

Принципы цифрового кодирования растрового изображения

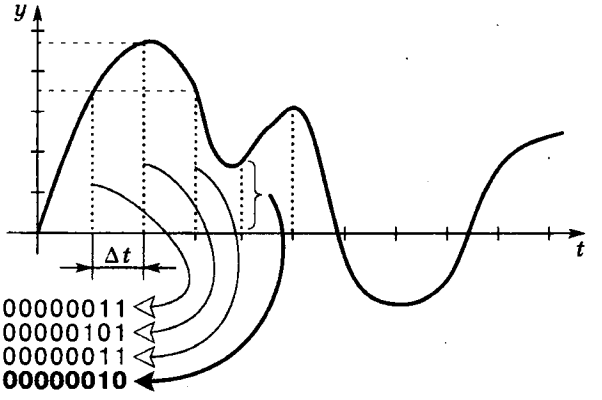


Цвет одной точки как смесь цветов: красного (R), зелёного (G) и синего (B). Яркость каждой из этих составляющих кодируется 1 байтом.
Для монохромного изображения цвет одной точки кодируется 1 битом.

Общий объём информации, байт = (цвет одной точки) × (кол-во точек в строке) × × (кол-во строк) = 3 байта × 20 × 20 = 1200 байт.

Принципы цифрового кодирования аналогового сигнала (на примере записи звука)

Измерение амплитуды (величины напряжения) через равные малые промежутки времени Δt (с определённой частотой). Получаемые округленные числовые значения в двоичной форме последовательно записываются в файл.



Частота дискретизации в Гц означает, что измерение электрического сигнала (громкости звука) осуществляется указанное количество раз в секунду, т.е. в файл каждую секунду записывается данное количество двоичных чисел. Разрешение в битах определяет разрядность каждого записываемого в файл числа. Если записывается стереозвук (т.е. ведётся двухканальная запись), то оцифровке подвергается не один электрический сигнал, а сразу два (с левого и правого микрофона) и, соответственно, удваивается количество сохраняемой цифровой информации; для монофонической записи умножение на 2 не требуется.

Общий объём информации, бит = (частота дискретизации, Гц) × (разрешение) × × (длительность записи, с) × 2 (для стерео).

Например, при частоте дискретизации 48 кГц (= 48000 Гц), разрешении 24 бита, длительности записи 1 мин (= 60 с) и стереозвуке получается:

Общий объём информации, бит = 48 000 × 24 (бит) × 60 (с) × 2 = 138240000 бит = = 17280000 байт = 16875Кбайт.

Частота дискретизации = кол-во замеров в секунду

Разрешение = разрядность одного отсчёта

Длительность записи

Кол-во записываемых каналов

Принципы цифрового кодирования видеозаписи

24 кадра в секунду

Объём одного кадра вычисляется аналогично растровому рисунку

Общий объём информации, байт = (объём одного кадра) × 24 × (длительность, с) = = (цвет одной точки) × (кол-во точек в строке) × (кол-во строк) × 24 × (длительность, с).

Например, для видеофильма длительностью 2 мин (= 120 с) при разрешении 640 × 480 получается:

Общий объём информации = 3 байта × 640 × 480 × 24 × 120 с = 2654208000 байт = = 2592000 кбайт = 2531,25 Мбайт.

Всё сказанное выше относится к так называемым «несжатым» форматам. Существуют также различные форматы хранения графической, аудио- и видеоинформации со сжатием, позволяющие (иной раз существенно) уменьшить объём записываемой информации, однако в ЕГЭ обычно рассматриваются именно «чистые» форматы без сжатия.

Разбор типовых задач

Задача 1*. Для хранения растрового изображения размером 128 × 128 пикселей отвели 4 килобайта памяти. Каково максимально возможное число цветов в палитре изображения?

1) 8 2) 2 3) 16 4) 4

Решение

Общее кол-во информации = 4 Кб = $4 \cdot 2^{10}$ байт.

Количество пикселей в изображении: 128 × 128 = 16384.

Объём информации на 1 пиксель = $4 \cdot 2^{10}$ байт / 16384 = 4096 байт / 16384 = = 32768 бит / 16384 = 2 бита.

Два бита (двухразрядное двоичное число) может кодировать $2^2 = 4$ цвета.

Ответ: 4 цвета.

Задача 2*. Сколько секунд потребуется модему, передающему сообщения со скоростью 28800 бит/с, чтобы передать цветное растровое изображение размером 640 × 480 пикселей, при условии, что цвет каждого пикселя кодируется тремя байтами?

Решение

Объём информации, соответствующей изображению = 3 байта × 640 × 480 = 921600 байт.

Длительность передачи информации = 921600 байт / 28800 бит/с = = 7372800 бит / 28800 бит/с = 256 с.

Ответ: 256 секунд.

Следует не забывать приводить все величины к одной размерности (килобайты к байтам, байты к битам, минуты к секундам и т. д.).

Задача 3*. Укажите минимальный объём памяти (в килобайтах), достаточный для хранения любого растрового изображения размером 64 × 64 пикселя, если известно, что в изображении используется палитра из 256 цветов. Саму палитру хранить не нужно.

1) 128 2) 2 3) 256 4) 4

Решение

Палитра из 256 цветов может быть закодирована восьмиразрядным двоичным числом ($2^8 = 256$), т.е. каждый пиксель соответствует 1 байту.

Объём информации, соответствующей изображению = 1 байт × 64 × 64 = 4096 байтов = = 4 Кбайт.

Ответ: 4 Кбайт.

Задача 4*. Для хранения растрового изображения размером 64 × 64 пикселя отвели 512 байтов памяти. Каково максимально возможное число цветов в палитре изображения?

1) 16 2) 2 3) 256 4) 1024

Решение

Объём информации, соответствующей изображению, равен 512 байтов.

Количество пикселей в изображении: 64 × 64 = 4096.

Объём информации на 1 пиксель = 512 байт / 4096 = 4096 бит / 4096 = 1 бит.

Один бит может кодировать $2^1 = 2$ цвета.

Ответ: 2 цвета.

Задача 5. Производится двухканальная (стерео) звукозапись с частотой дискретизации 48 кГц и 24-битным разрешением. Запись длится 1 минуту, её результаты записываются в файл, сжатие данных не производится. Какое из приведённых ниже чисел наиболее близко к размеру полученного файла, выраженному в мегабайтах?

1) 0,3 2) 4 3) 16 4) 132

Решение

Частота дискретизации = 48 кГц = 48 000 Гц.

Разрешение = 24 бита.

Длительность записи = 1 мин = 60 с.

Стереозвук (двухканальная запись) = 2.

Общий объём записываемой информации = 48 000 Гц × 24 бит × 60 с × 2 бит.

Расчёты удобнее всего производить, по возможности переведя все сомножители в степени двойки (поскольку затем потребуется выполнять перевод в байты, килобайты, мегабайты и пр.). Поэтому формула получит вид:

$48\,000 \times 24$ (бит) $\times 60$ (с) $\times 2 = 375 \cdot 2^7 \times 3 \cdot 2^3 \times 15 \cdot 2^2 \times 2 = 16875 \cdot 2^{(7+3+2+1)} =$

$= 16875 \cdot 2^{13}$ (бит).

По условию задачи нужно сравнить полученный результат с различными вариантами ответа, приведёнными в мегабайтах. Поэтому результат вычислений требуется разделить сначала на 2^3, чтобы перевести биты в байты, а затем — на 2^{20} для перевода байтов в мегабайты:

$16875 \cdot 2^{13}$ (бит) $= 16875 \cdot 2^{10}$ (байт) $= 16875 / 2^{10}$ (Мб) $= 16875 / 1024$ (Мб).

Вычисления достаточно выполнить приближенно, так как не требуется указывать точный ответ, а достаточно указать, к какому из имеющихся значений он ближе всего. Поэтому деление на 1024 можно для упрощения вычислений заменить делением на 1000. Тогда объём получаемого аудиофайла примерно равен 16,875 Мб.

Ответ: вариант ответа № 3.

Задачи для самостоятельного решения

1. Для хранения растрового изображения размером 32 × 32 пикселя отвели 512 байтов памяти. Каково максимально возможное число цветов в палитре изображения?
   1) 256 2) 2 3) 16 4) 4
2. Аналоговый звуковой сигнал был дискретизирован сначала с использованием 65536 уровней интенсивности сигнала (качество звучания аудио-CD), а затем с использованием 16 уровней интенсивности сигнала (качество звучания радиотрансляции). Во сколько раз различаются информационные объемы оцифрованного звука?
   1) в 2 раза 2) в 4 раз 3) в 8 раз 4) в 16 раз

3. Производится одноканальная монозвукозапись с частотой дискретизации 16 кГц и 16-битным разрешением. Запись длится 2 минуты, её результаты записываются в файл, сжатие данных не производится. Какое из приведённых ниже чисел наиболее близко к размеру полученного файла, выраженному в Килобайтах?
1) 0,2 2) 2 3) 3 4) 4

4. Производится одноканальная (моно) звукозапись с частотой дискретизации 64 Гц. При записи использовались 64 уровня дискретизации. Запись длится 4 минуты 16 секунд, её результаты записываются в файл, причём каждый сигнал кодируется минимально возможным и одинаковым количеством битов. Какое из приведённых ниже чисел наиболее близко к размеру полученного файла, выраженному в мегабайтах?
1) 10 2) 12 3) 14 4) 16

5. Производится одноканальная (моно) звукозапись с 24-битным разрешением. Запись длится 1 минуту 4 секунды, её результаты записываются в файл, сжатие данных не производится. Получен файл размером 3275 Кбайт. Определите частоту дискретизации в килогерцах.

6. Проводилась одно канальная (моно) звукозапись с частотой дискретизации 16 кГц и 32-битным разрешением. В результате был получен файл размером 10 Мбайт, сжатие данных не производилось. Какая из приведённых ниже величин наиболее близка к времени, в течение которого проводилась запись?
1) 3 мин 2) 5 мин 3) 10 мин 4) 15 мин

7. 4-цветное растровое изображение размером 64 × 256 пикселей сохранили в виде несжатого файла, закодировав каждый пиксель минимально возможным количеством бит. Какой размер получившегося файла (в килобайтах)?
1) 4 2) 32 3) 4096 4) 8

8. Производится двухканальная (стерео) звукозапись с частотой дискретизации 16 кГц и 4-битным разрешением. Запись длится полминуты, её результаты записываются в файл, сжатие данных не производится. Какая из приведённых ниже величин наиболее близка к размеру полученного файла?
1) 0.5 Мбайт 2) 8 Кбайт 3) 240 Кбайт 4) 4 Мбайт

9. Производится двухканальная (стерео) звукозапись с частотой дискретизации 16 кГц и количеством уровней квантования 65536. Запись длится 4 минуты, её результаты записываются в файл, сжатие данных не производится. Какая из приведённых ниже величин наиболее близка к размеру полученного файла?
1) 2 Мбайта 2) 15 Мбайт 3) 8 Мбайт 4) 120 Мбайт

10. Оцифровывается старый немой черно-белый кинофильм. Разрешение составляет 320 × 240. Частота 24 кадра в секунду. Длительность фильма — 50 минут. Сжатие данных при оцифровке видео не производится. Уместится ли полученный видеофайл на CD ёмкостью 700 Мб? (Ответьте: «Да» или «Нет».)

Ответы для самопроверки

№ задания	Ответ	№ задания	Ответ
1	3	6	4
2	2	7	1
3	4	8	1
4	2	9	2
5	18Кц	10	Да

Обработка числовой информации

A7 Электронные таблицы. Ссылки. Формулы

Конспект

Абсолютные, относительные и смешанные ссылки

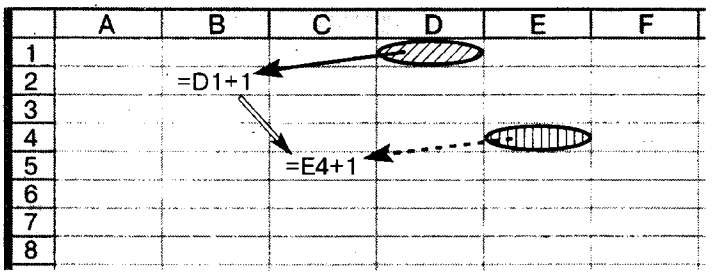
Ссылка на ячейку представляет собой запись имени ячейки. Ссылка на диапазон представляет собой запись имени диапазона.

Примеры:

=A3 + C5 — сложить числа в ячейках **A3** и **C5**;

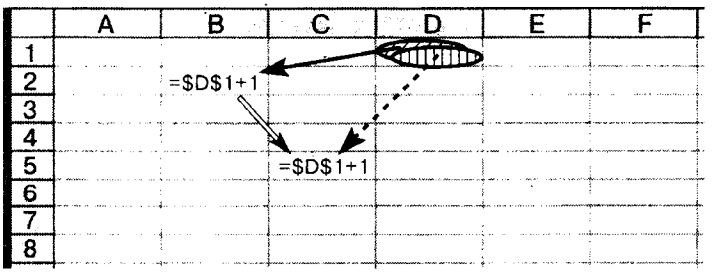
=СУММ(A3:C5) — функция вычисления суммы чисел во всем диапазоне **A3:C5**.

При копировании формулы с такими ссылками в другие ячейки ссылки автоматически изменяются (модифицируются) так, что всегда указывают на ячейки или диапазоны относительно ячейки, содержащей формулу. Поэтому такие ссылки называют **относительными**.



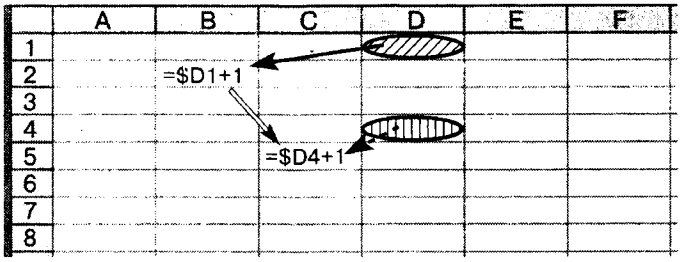
При копировании формулы из ячейки **B2** в ячейку **C5** ссылка меняется так, что всегда указывает на ячейку, расположенную на 2 столбца правее и на 1 строку выше относительно ячейки с формулой.

Запись имени ячейки (или имён ячеек в имени диапазона), в которой имя столбца и номер строки предваряются символом $, являются **абсолютными** ссылками. Абсолютная ссылка не меняется при копировании формулы в другую ячейку.

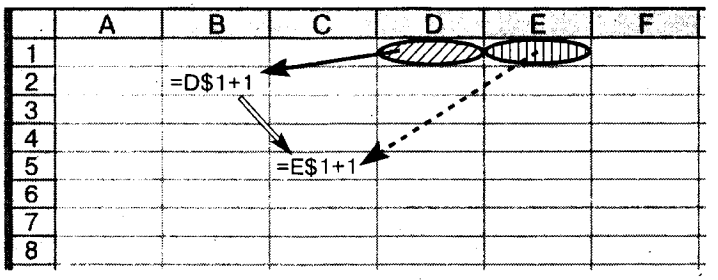


При копировании формулы из ячейки **B2** в ячейку **C5** ссылка не меняется и всегда указывает на одну и ту же ячейку.

Ссылки, в которых символ $ стоит только перед именем столбца или только перед номером строки, называют **смешанными.** Символ $ в смешанной ссылке делает абсолютным только имя столбца или только имя строки, перед которым он стоит.



В данной смешанной ссылке абсолютным является имя столбца. Поэтому при копировании формулы из ячейки **B2** в ячейку **C5** в ссылке имя столбца не меняется, а номер строки меняется относительно ячейки с формулой.



В данной смешанной ссылке абсолютным является номер строки. Поэтому при копировании формулы из ячейки **B2** в ячейку **C5** в ссылке номер строки не меняется, а имя столбца меняется относительно ячейки с формулой.

Функции

В Excel предусмотрены стандартные функции, которые можно использовать в формулах:

- математические — различные вычисления (арифметические, тригонометрические, логарифмические, квадратный корень, степень, округление и т.д.);
- текстовые — работа с текстовыми строками;
- логические — работа с логическими значениями;
- дата и время — работа с датой и временем;
- финансовые — денежные расчёты (проценты по вкладам и пр.);
- статистические — статистическая обработка данных, вероятности и пр.;
- ссылки и массивы — работа с данными в диапазонах (например, поиск значения и возврат адреса ячейки с ним);
- работа с базой данных — работа с записями в электронной таблице как базе данных;
- проверка свойств и значений — определение типа данных в ячейке (число, текст и т.д.).

В формуле записывается имя функции, после которого в круглых скобках через точку с запятой записываются значения — аргументы.

Функции могут быть вложенными: в качестве аргумента одной функции записывается другая функция со своими аргументами.

Разбор типовых задач

Задача 1*. В ячейке **C2** записана формула **$E$3+D2.** Какой вид приобретёт формула после того, как ячейку **C2** скопируют в ячейку **B1**?

Примечание: знак $ используется для обозначения абсолютной адресации.

1) $E$3+C1 2) $D$3+D2 3) $E$3+E3 4) $F$4+D2

Решение

Первое слагаемое в предлагаемой формуле представляет собой абсолютную ссылку и не изменяется при копировании формулы в другую ячейку. Этому требованию соответствуют варианты ответов 1 и 3.

Второе слагаемое является относительной ссылкой, поэтому при копировании формулы в нём могут измениться как имя столбца, так и номер строки.

Копирование формулы из ячейки **C2** в ячейку **B1** соответствует её смещению на один столбец левее и на одну строку выше, поэтому во втором слагаемом имя столбца **D** должно измениться на **C**, а номер строки **2** — на **1**.

Ответ: $E$3+C1 (вариант №1)

Задача 2*. При работе с электронной таблицей в ячейке **A1** записана формула **=D1-$D2**. Какой вид приобретёт формула, после того как ячейку **A1** скопируют в ячейку **B1**?

Примечание: символ $ в формуле обозначает абсолютную адресацию.

1) =E1-$E2 2) =E1-$D2 3) =E2-$D2 4) =D1-$E2

Решение

В приведённой формуле уменьшаемое представляет собой относительную ссылку, в которой при копировании формулы могут меняться как имя столбца, так и номер строки.

Вычитаемое в формуле представляет собой смешанную ссылку, в которой абсолютным является имя столбца, — следовательно, при копировании формулы имя столбца останется неизменным, а номер строки может меняться.

При копировании формулы из ячейки **A1** в ячейку **B1** меняется только имя столбца (формула смещается на один столбец вправо). Следовательно, в изменившейся формуле номера строк останутся неизменными, а изменение имени столбца с **D** на **E** должно произойти только в уменьшаемом. Этим условиям соответствует вариант 2 (в вариантах 1 и 4 ответа изменено имя столбца в абсолютной части смешанной ссылки, а в варианте 3 в уменьшаемом изменён номер строки).

Ответ: =E1-$D2 (вариант №2).

Задача 3*. В ячейке **B1** записана формула **=2*$A1**. Какой вид приобретёт формула после того как ячейку **B1** скопируют в ячейку **C2**?

Примечание: знак $ используется для обозначения абсолютной адресации.

1) =2*$B1 2) =2*$A2 3) =3*$A2 4) =3*$B2

Решение

В данной формуле единственная ссылка на исходную ячейку является смешанной, с абсолютным именем столбца.

Копирование формулы из ячейки **B1** в ячейку **C2** соответствует её смещению на один столбец правее и на одну строку ниже. Поскольку в ссылке имя столбца абсолютно, в формуле при таком копировании должен измениться только номер строки (увеличиться на 1). Этому соответствует вариант ответа 2 (в варианте 1 изменено абсолютное имя столбца в ссылке, а в вариантах 3 и 4 изменено значение константы 2).

Ответ: =2*$A2 (вариант №2).

Задача 4*. Дан фрагмент электронной таблицы:

	A	B	C
1	10	20	=A1+B$1
2	30	40	

Чему станет равным значение ячейки **C2**, если в неё скопировать формулу из ячейки **C1**?

Примечание: знак $ обозначает абсолютную адресацию.

1) 40 2) 50 3) 60 4) 70

Решение

В приведённой формуле первое слагаемое является относительной ссылкой, в которой может меняться и имя столбца, и номер строки.

Второе слагаемое представляет собой смешанную ссылку с абсолютным номером строки, т.е. номер строки при копировании формулы меняться не может, а имя столбца может измениться.

При копировании формулы из ячейки **C1** в ячейку **C2** имя столбца остаётся неизменным, а номер строки увеличивается на 1 (формула смещается на одну строку ниже). Поэтому в первом слагаемом при копировании формулы на 1 увеличивается номер строки, а имя столбца остаётся неизменным; во втором же слагаемом и имя столбца, и номер строки останутся неизменными. Таким образом, формула после копирования примет вид: **=A2+B$1**.

Учитывая числовые значения в левой части таблицы (в столбцах **A** и **B**), результат вычисления этой формулы в ячейке **C2** будет равен 30 + 20 = 50.

Ответ: 50 (вариант №2).

Задача 5*. В электронной таблице значение формулы **=СУММ(B1:B2)** равно 5. Чему равно значение ячейки **B3**, если значение формулы **=СРЗНАЧ(B1:B3)** равно 3?

1) 8 2) 2 3) 3 4) 4

Решение

Речь идёт о таблице, состоящей из трёх ячеек: **B1**, **B2** и **B3**. При этом **B1** + **B2** равно 5, а среднее значение, т.е. (**B1** + **B2** + **B3**)/3 равно 3. Требуется определить значение **B3**.

Подставив значение первой суммы во второе равенство получается:

$$(5 + B3)/3 = 3.$$

Остаётся решить полученное уравнение относительно значения (переменной) **B3**:

$$5 + B3 = 9;$$

отсюда **B3** = 9 − 5 = 4.

Ответ: 4 (вариант №4).

Задача 6*. В динамической (электронной) таблице приведены значения пробега автомашин (в км) и общего расхода дизельного топлива (в литрах) в четырёх автохозяйствах с 12 по 15 июля. В каком из хозяйств средний расход топлива на 100 км пути за эти четыре дня наименьший?

Название автохозяйства	12 июля		13 июля		14 июля		15 июля		За четыре дня	
	Пробег	Расход	Пробег	Расход	Пробег	Расход	Пробег	Расход	Пробег	Расход
Автоколонна №11	9989	2134	9789	2056	9234	2198	9878	2031	38890	8419
Грузовое такси	490	101	987	215	487	112	978	203	2942	631
Автобаза №6	1076	147	2111	297	4021	587	1032	143	8240	1174
Трансавтопарк	998	151	2054	299	3989	601	1023	149	8064	1200

1) Автоколонна № 11 3) Автобаза №6

2) Грузовое такси 4) Трансавтопарк

Решение

Необходимо сравнивать средний расход топлива на 100 км пути за четыре дня. Его можно вычислить как частное от деления значения в графе «Расход» на значение в графе «Пробег» из колонки таблицы «За четыре дня» (остальная информация для нас в данном случае избыточна).

Составляется таблица:

Название автохозяйства	Пробег за четыре дня	Расход за четыре дня	Средний расход (приблизительно)
Автоколонна №11	38890	8419	0,216
Грузовое такси	2942	631	0,214
Автобаза №6	8240	1174	0,142
Трансавтопарк	8064	1200	0,49

Вывод: наименьший средний расход топлива на 100 км пути за четыре дня — в автохозяйстве «Автобаза №6».

Ответ: Автобаза №6 (вариант ответа №3)

Задача 7*[1]. *Из правил соревнования по тяжелой атлетике:*

Тяжелая атлетика — это прямое соревнование, когда каждый атлет имеет три попытки в рывке и три попытки в толчке. Самый тяжелый вес поднятой штанги в каждом упражнении суммируется в общем зачёте. Если спортсмен потерпел неудачу во всех трёх попытках в рывке, он может продолжить соревнование в толчке, но уже не сможет занять какое-либо место по сумме 2-х упражнений.

Если два спортсмена заканчивают состязание с одинаковым итоговым результатом, высшее место присуждается спортсмену с меньшим весом. Если же вес спортсменов одинаков, преимущество отдаётся тому, кто первым поднял победный вес.

Таблица результатов соревнований по тяжёлой атлетике:

Фамилия И.О.	Вес спортсмена	Взято в рывке	Рывок с попытки	Взято в толчке	Толчок с попытки
Айвазян Г.С.	77,1	150,0	3	200,0	2
Викторов М.П.	79,1	147,5	1	202,5	1
Гордезиани Б.Ш.	78,2	147,5	2	200,0	1
Михальчук М.С.	78,2	147,5	2	202,5	3
Пай С.В.	79,5	150,0	1	200,0	1
Шапсугов М.Х.	77,1	147,5	1	200,0	1

Кто победил в общем зачёте (сумме двух упражнений)?

1) Айвазян Г.С.
2) Викторов М.П.
3) Михальчук М.С.
4) Пай С.В.

Решение

Формализуются правила оценивания соревнований:

1) «самый тяжёлый вес поднятой штанги в каждом упражнении суммируется в общем зачёте» — это означает, что в таблицу в столбцах «Взято в рывке» и «Взято в толчке» записывается максимальный успешно поднятый вес штанги из трёх сделанных попыток;

2) «если спортсмен потерпел неудачу во всех трёх попытках в рывке, он может продолжить соревнование в толчке, но уже не сможет занять какое-либо место по сумме 2-х упражнений» — означает, что из анализа результатов исключаются спортсмены, имеющие нулевое (либо пустое) значение в столбце «Взято в рывке»;

---

[1] Данная задача отличается плохо формализуемым условием и достаточно сложна; вместе с тем, по своему содержанию она во многом аналогична предыдущему заданию. Решение данной задачи приводится для общего ознакомления и показано как демонстрация практического использования электронных таблиц (можно ожидать, что с переходом к компьютерному проведению ЕГЭ характер заданий будет меняться именно в этом направлении). «Ручное» решение (без использования Excel) выполняется аналогично.

3) «если два спортсмена заканчивают состязание с одинаковым итоговым результатом, высшее место присуждается спортсмену с меньшим весом» — итоговым результатом является сумма значений в столбцах «Взято в рывке» и «Взято в толчке»; если эта сумма для двух спортсменов равна, то первым из них будет тот, у кого меньше значение в столбце «Вес спортсмена»;

4) «если же вес спортсменов одинаков, преимущество отдается тому, кто первым поднял победный вес» — данное условие формализовать сложнее всего. Считается, что если равны как сумма значений в столбцах «Взято в рывке» и «Взято в толчке», так и значение в столбце «Вес спортсмена», то первым из спортсменов будет тот, у кого суммарный вес в рывке и в толчке достигнут за меньшее значение суммы количеств попыток.

Таким образом, при определении победителя необходимо:

1) добавить в таблицу два дополнительных столбца — «Суммарный вес» и «Суммарное количество попыток» и заполнить их формулами вычисления сумм соответствующих значений;

2) выполнить автофильтрацию таблицы по признаку «непустые ячейки в столбце «Взято в рывке»;

3) выполнить сортировку отфильтрованной таблицы по убыванию значений в столбце «Суммарный вес», а при равных значениях — по возрастанию значений в столбце «Вес спортсмена»;

4) в полученной таблице произвести сортировку по возрастанию значений в столбце «Суммарное количество попыток».

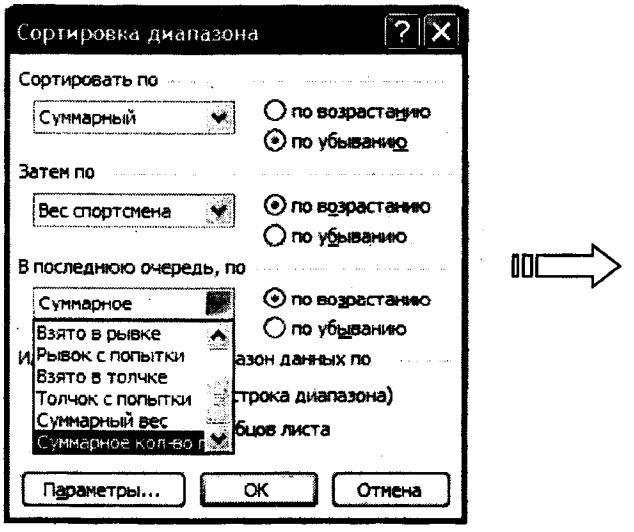
В результате фамилия спортсмена — победителя будет содержаться в первой строке таблицы.

1)

	A	B	C	D	E	F	G	H
1	Таблица результатов соревнований по тяжелой атлетике:							
2	Фамилия И.О.	Вес спортсмена	Взято в рывке	Рывок с попытки	Взято в толчке	Толчок с попытки	Суммарный вес	Суммарное кол-во попыток
3	Айвазян Г.С.	77,1	150	3	200	2	=C3+E3	=D3+F3
4	Викторов М.П.	79,1	147,5	1	202,5	1	=C4+E4	=D4+F4
5	Гордезиани Б.Ш.	78,2	147,5	2	200	1	=C5+E5	=D5+F5
6	Михальчук М.С.	78,2	147,5	2	202,5	3	=C6+E6	=D6+F6
7	Пай С.В.	79,5	150	1	200	1	=C7+E7	=D7+F7
8	Шапсугов М.Х.	77,1	147,5	1	200	1	=C8+E8	=D8+F8

2) в столбце «Взято в рывке» пустых или нулевых ячеек нет;

3) – 4)



	A	B	C	D	E	F	G	H
1	Таблица результатов соревнований по тяжелой атлетике:							
2	Фамилия И.О.	Вес спортсмена	Взято в рывке	Рывок с попытки	Взято в толчке	Толчок с попытки	Суммарный вес	Суммарное кол-во попыток
3	Айвазян Г.С.	77,1	150	3	200	2	350	5
4	Викторов М.П.	79,1	147,5	1	202,5	1	350	5
5	Гордезиани Б.Ш.	78,2	147,5	2	200	1	350	2
6	Михальчук М.С.	78,2	147,5	2	202,5	3	350	2
7	Пай С.В.	79,5	150	1	200	1	347,5	2
8	Шапсугов М.Х.	77,1	147,5	1	200	1	347,5	3

Ответ: Айвазян Г.С. (вариант №1).

Задача 8*. Три страны: Королевство Бельгия, Королевство Нидерланды и Великое Герцогство Люксембург образуют экономико-политический союз, который носит название Бенилюкс. Ниже приведён фрагмент электронной таблицы, характеризующий каждую из стран союза и союз в целом:

	A	B	C	D
1	Страна	Население (тыс. чел)	Площадь (кв. км)	Плотность населения (чел/кв.км)
2	Бельгия	10 415	30 528	341
3	Нидерланды	16 357	41 526	394
4	Люксембург	502	2 586	194
5	Бенилюкс в целом	27 274	74 640	

Какое значение должно стоять в ячейке D5?

1) 365
2) 929
3) 310
4) 2,74

Решение

Плотность населения вычисляется как частное от деления значения в графе «Население» на значение в графе «Площадь» в каждой строке таблицы.

Разделив в строке «Бенилюкс в целом» значение 27 274 000 (население указано в тысячах человек) на значение 74 640 (площадь), получается искомое значение в ячейке D5: 365,4.

Поскольку в списке ответов точное значение отсутствует, следует округлить его: $365{,}4 \approx 365$.

Ответ: 365 чел / кв.км. (вариант ответа №1).

Внимание! Нужно не забывать переводить приведённые в условии задачи значения в нужную размерность.

Если среди приведённых в задаче вариантов ответа отсутствует вычисленное точное значение, следует попытаться найти в вариантах ответа округленное значение.

Задачи для самостоятельного решения

1. В некотором царстве, в некотором государстве провели перепись населения во всех 7 уездах. Используя представленную таблицу, укажите номер уезда с наименьшую плотностью населения.

Название	Площадь (тыс. км^2)	Население (2002 г.)
1. Лапотный	1677,9	14158
2. Большие Сапоги	650,7	36482
3. Сермяжный	1038	31642
4. Луга	589,2	21471
5. Стольный Град	1788,9	12520
6. Таёжный	5114,8	20542
7. Далёкие Озёра	6515,9	7038

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

2. В электронной таблице значение формулы **=СУММ(A1:C1;A2:B2)** равно 15. Чему равно значение ячейки **C2**, если значение формулы **=СРЗНАЧ(A1:C2)** равно 3?
1) 1 2) 2 3) 3 4) 4

3. В электронной таблице значение формулы **=СУММ(F3:F5)** равно 14. Чему равно значение ячейки **F2**, если значение формулы **=СРЗНАЧ(A1:C2)** равно 5?
1) 8 2) 4 3) 4) 6

4. Дан фрагмент электронной таблицы:

	A	B	C
1		15	=C2+$B1
2	12	10	20

Чему станет равным значение ячейки **A1**, если в неё скопировать формулу из ячейки **C1**?
Примечание: знак $ обозначает абсолютную адресацию.
1) 35 2) 27 3) 32 4) 25

5. Дан фрагмент электронной таблицы:

	A	B	C
1	1	3	8
2	4		14
3	7	=C2/A3	5

Чему станет равным значение ячейки **B2**, если в неё скопировать формулу из ячейки **B3**?
Примечание: знак $ обозначает абсолютную адресацию.
1) 2 2) 3,5 3) 4 4) 6

6. Дан фрагмент электронной таблицы:

	A	B	C
1	18	3	
2	=$A$1/$C$2	6	9

Чему станет равным значение ячейки **C1**, если в неё скопировать формулу из ячейки **A2**?

Примечание: знак $ обозначает абсолютную адресацию.

1) 9 2) 6 3) 3 4) 2

7. Дан фрагмент электронной таблицы:

	A	B	C
1	2	8	11
2	=B$2+A1	5	9
3		4	7

Чему станет равным значение ячейки **A3**, если в неё скопировать формулу из ячейки **A2**?

Примечание: знак $ обозначает абсолютную адресацию.

1) 7 2) 12 3) 11 4) 10

8. Дан фрагмент электронной таблицы:

	A	B	C	D
1	=B3*D2	2	5	7
2	4	=A$2+B1	=C1+$C3	
3	3		1	8

Чему станет равным значение ячейки **A1**, если в ячейку **B3** скопировать формулу из ячейки **B2**, а в ячейку **D2** скопировать формулу из ячейки **C2**?

Примечание: знак $ обозначает абсолютную адресацию.

1) 48 2) 60 3) 36 4) 80

9. В ячейке **F3** записана формула **$A$2+G5**. Какой вид приобретёт формула после того, как ячейку **F3** скопируют в ячейку **B1**?

Примечание: знак $ используется для обозначения абсолютной адресации.

1) $A$2+C1 2) $A$2+C3 3) $A$2+B3 4) $B$1+C3

10. В ячейке **A3** записана формула **$B7*D$5**. Какой вид приобретёт формула после того, как ячейку **A3** скопируют в ячейку **C1**?

Примечание: знак $ используется для обозначения абсолютной адресации.

1) $D5*F$3 2) $B7*F$5 3) $B5*F$5 4) $B5*D$5

Ответы для самопроверки

Задача	Ответ
1	7
2	3
3	4
4	2
5	1
6	4
7	2
8	4
9	2
10	3

Обработка числовой информации

B5 Электронные таблицы. Графики и диаграммы

Конспект

Диаграммы

Диаграмма строится по числовым значениям в некотором диапазоне (блоке) таблицы. При этом подписи диаграммы обычно также берутся из соответствующих диапазонов таблицы (из её «шапки» и/или левого столбца с названиями строк).

При этом в таблице определяются исходные и зависимые величины. Исходные величины называют *категориями*, зависящие от них величины называют *рядами данных*.

Пример (таблица зависимости координаты тела, движущегося по прямой, от времени; для трёх различных экспериментов):

Категории

t	1	2	3	4	5	6	7	8
x_1	2	2,5	3	4	6	9	15	27
x_2	1,5	2	2,5	3	3,5	4	4,5	5
x_3	1	4	9	16	25	36	49	64

Ряды данных

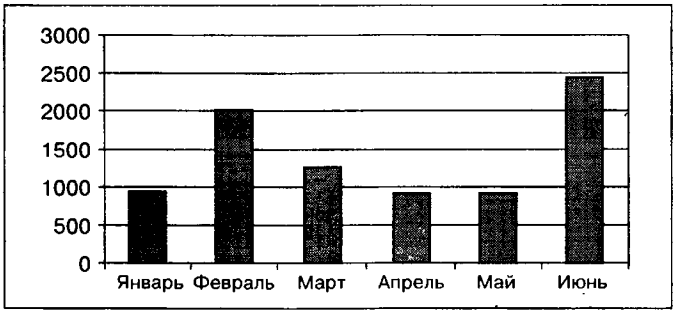
На диаграмме, в зависимости от её вида, может быть отображён только один ряд данных либо несколько рядов данных. Значения категорий при этом являются подписями к элементам диаграммы, соответствующим значениям ряда/рядов, либо определяют значения по оси X (тогда значения рядов данных — это значения по оси Y).

Имена рядов данных, как правило, отображаются на диаграмме в составе её *легенды* — таблицы, определяющей соотношение имени каждого ряда данных и его графического отображения (цвета линии, вида значков, соответствующих точкам и т.д.).

Основные виды диаграмм

1. **Гистограмма** — состоит из вертикальных прямоугольников. Каждый столбик соответствует одному значению таблицы для соответствующей категории.

Пример чтения гистограммы:



Ось категорий располагается по горизонтали (здесь: содержит названия месяцев).

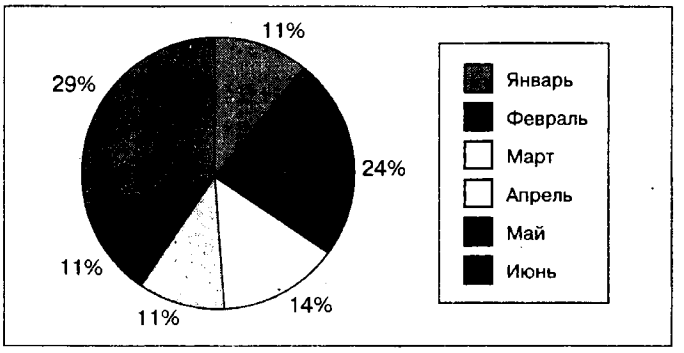
Ряд данных — один. Числовые значения соответствуют высотам столбиков и приближенно определяются по вертикальной оси (слева).

Месяц	январь	февраль	март	апрель	май	июнь
Значение	900	2000	1200	900	900	2500

В гистограмме (а также линейчатой диаграмме, графике, точечной диаграмме) для вычисления процентных долей значений ряда данных относительно суммы для всего ряда нужно сложить все значения ряда данных, определить тем самым общее значение и вычислять процентные доли делением каждого значения ряда данных на сумму их значений.

2. Круговая диаграмма — представляет собой набор секторов круга. Каждый сектор соответствует процентной доле одного значения таблицы для соответствующей категории относительно суммы всех значений ряда данных.

Пример чтения круговой диаграммы:



Категории определяются по легенде или надписям на самой диаграмме (здесь: названия месяцев).

Ряд данных — один (круговая диаграмма всегда отображает один ряд данных). Процентные доли каждого значения ряда данных, соответствующего той или иной категории, могут быть надписаны на диаграмме либо определяются приближенно как процентные доли углов соответствующих секторов относительно всего круга (360°).

Круговая диаграмма не даёт информации об абсолютных значениях ряда данных! Для их определения необходимо знать либо суммарное значение по всему ряду данных, либо абсолютное значение хотя бы одного элемента данных (дальнейший расчёт выполняется пропорционально).

Пусть известно, что июню (29%) соответствует числовое значение 870. Тогда можно по имеющейся диаграмме восстановить (рассчитать) значения для остальных категорий:

Месяц	январь	февраль	март	апрель	май	июнь	СУММАРНО
Доля, %	11	24	14	11	11	29	100
Значение	330	720	420	330	330	870	3000

3. Гистограмма для нескольких рядов данных — состоит из наборов соответствующих количеств столбцов разного цвета или фактуры (каждому ряду данных соответствует свой цвет или фактура, указанный в легенде).

Пример чтения гистограммы для нескольких рядов данных:

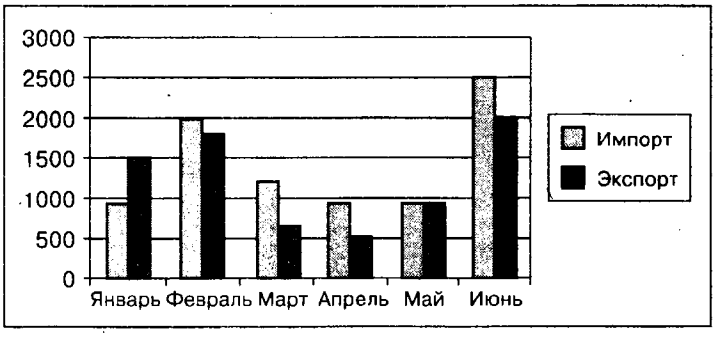


Диаграмма читается аналогично обычной гистограмме, но нужно отдельно считывать значения для столбиков каждого ряда данных.

Ось категорий располагается по горизонтали (здесь: содержит названия месяцев).

Рядов данных — два (первый, «импорт», — голубой цвет, второй, «экспорт», — лиловый цвет). Числовые значения соответствуют высотам столбиков и приближенно определяются по вертикальной оси (слева).

Месяц	январь	февраль	март	апрель	май	июнь
Импорт	900	2000	1200	900	900	2500
Экспорт	1500	1800	700	500	900	2000

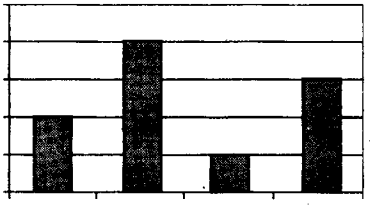
Разбор типовых задач

Задача 1*. Дан фрагмент электронной таблицы:

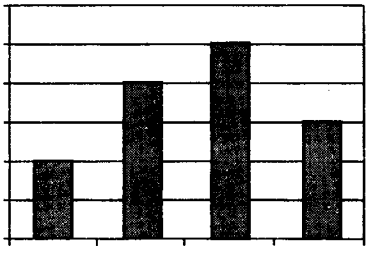
	A	B
1	=B1+1	1
2	=A1+2	2
3	=B2−1	
4	=A3	

После выполнения вычислений была построена диаграмма по значениям диапазона ячеек **A1:A4**. Укажите получившуюся диаграмму.

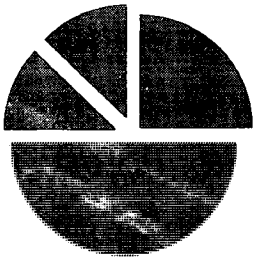
1)



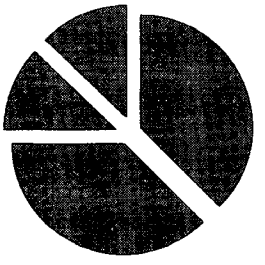
3)



2)



4)



Решение

Выполнив вручную вычисления, соответствующие указанным формулам, в ячейках получаются следующие числовые значения (приведён вид таблицы при последовательном выполнении вычислений над уже известными числовыми значениями; пунктирные стрелки указывают взаимное влияние ячеек):

	A	B
1	2 ◄ - - - - 1	
2	=A1+2	2
3	1 ◄	
4	=A3	

→

	A	B
1	2	1
2	4	2
3	1	
4	1	

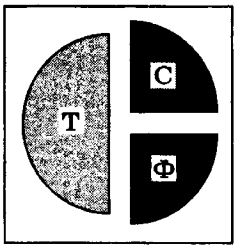
Диаграмма строится только по данным столбца А. При этом два последних значения одинаковы и равны 1, первое значение — вдвое больше, а второе — ещё вдвое больше. Такому соотношению не удовлетворяет ни одна столбчатая диаграмма (варианты №1 и №3). Что же касается круговых диаграмм, указанному соотношению исходных данных удовлетворяет только диаграмма варианта 2, так как вариант 4 предполагает две пары равных значений.

Ответ: вариант №2.

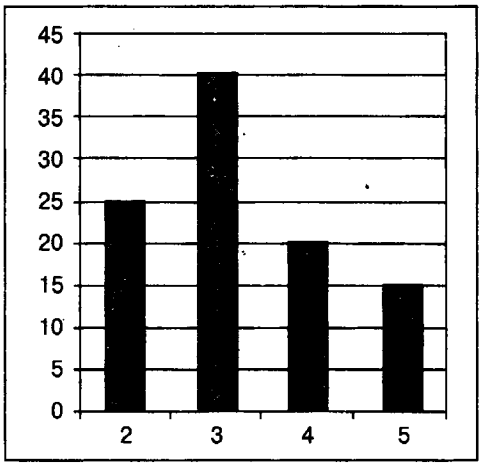
Задача 2*. В цехе трудятся рабочие трёх специальностей — токари (Т), слесари (С) и фрезеровщики (Ф). Каждый рабочий имеет разряд, не меньший второго и не больший пятого. На диаграмме I отражено количество рабочих с различными разрядами, а на диаграмме II — распределение рабочих по специальностям.

Каждый рабочий имеет только одну специальность и один разряд.

I



II



Имеются четыре утверждения:

А) Все рабочие третьего разряда могут быть токарями.

Б) Все рабочие третьего разряда могут быть фрезеровщиками.

В) Все слесари могут быть пятого разряда.

Г) Все токари могут быть четвёртого разряда.

Какое из этих утверждений следует из анализа обеих диаграмм?

1) А 2) Б 3) В 4) Г

Решение

Из диаграммы I следует, что больше всего — рабочих 3-го разряда (40 чел.); второй разряд имеют 25 чел.; четвёртый разряд имеют 20 чел.; меньше всего — рабочих 5-го разряда (15 чел.). Общее же число рабочих равно 40 + 25 + 20 + 15 = 100 чел.

Из диаграммы II следует, что токарей в цехе больше всего, их 50%, т. е. 50 чел. Количества же слесарей и фрезеровщиков одинаковы — по 25%, т.е. по 25 чел.

Последовательно анализируется справедливость каждого из приведённых утверждений в соответствии с обеими диаграммами.

А) Все рабочие третьего разряда могут быть токарями. Это возможно, так как токарей 50 чел., а рабочих 3-го разряда 40 чел.

Б) Все рабочие третьего разряда могут быть фрезеровщиками. Это невозможно: фрезеровщиков (как следует из диаграммы II) всего 25 чел., а 3-й разряд имеют 40 чел. Значит, только часть «третьеразрядников» может быть фрезеровщиками, но не все они.

В) Все слесари могут быть пятого разряда. Это невозможно: слесарей (как следует из диаграммы II) 25 чел., а 5-й разряд имеют только 15 чел. Значит, как минимум 10 слесарей должны иметь разряд, отличный от 5-го.

Г) Все токари могут быть четвёртого разряда. Это тоже невозможно: токарей 50 чел., а 4-й разряд имеют только 20 чел. Значит, как минимум 30 слесарей должны иметь разряд, отличный от 4-го.

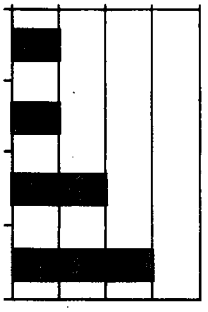
Вывод: истинным из четырёх приведённых в задаче высказываний является только высказывание А.

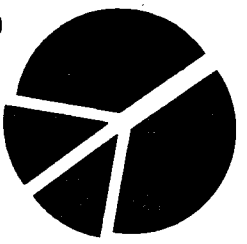
Ответ: высказывание А (вариант №1).

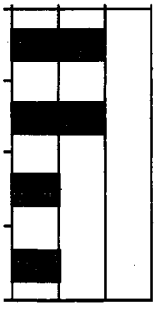
Задача 3*. Дан фрагмент электронной таблицы:

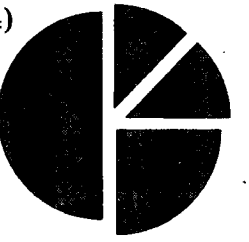
	A	B	C	D
1		3	4	
2	=C1–B1	=B1–A2*2	=C1/2	=B1+B2

После выполнения вычислений была построена диаграмма по значениям диапазона ячеек **A2:D2**. Укажите получившуюся диаграмму.

1) 

2) 

3) 

4) 

Решение

Прежде всего, вручную выполняются вычисления, соответствующие указанным формулам. В результате получаются в ячейках следующие числовые значения (приведён вид таблицы при последовательном выполнении вычислений над уже известными числовыми значениями; пунктирные стрелки указывают взаимное влияние ячеек):

	A	B	C	D
1		3	4	
2	1	=B1–A2*2	2	=B1+B2

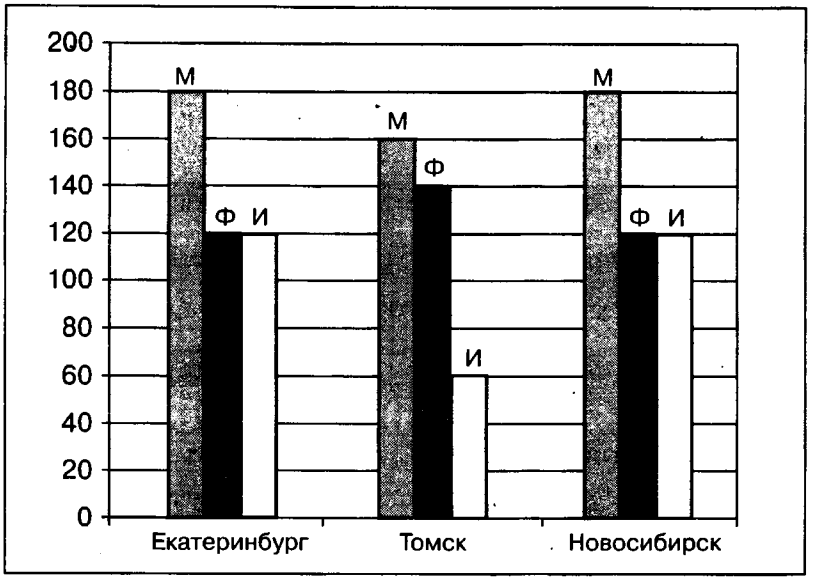
	A	B	C	D
1		3	4	
2	1	1	2	=B1+B2

	A	B	C	D
1		3	4	
2	1	1	2	4

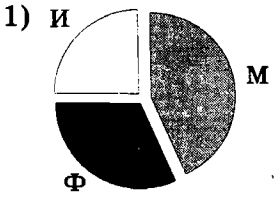
Диаграмма строится по значениям ячеек второй строки таблицы. При этом первые два значения равны друг другу, третье значение вдвое больше них, а четвёртое значение ещё в два раза больше. Такому соотношению удовлетворяет только круговая диаграмма 4: в столбиковой диаграмме 1 четвёртое значение равно 3 (если считать два первых равными 1), а в круговой диаграмме 2 и в столбиковой диаграмме 3 имеются две пары попарно равных значений.

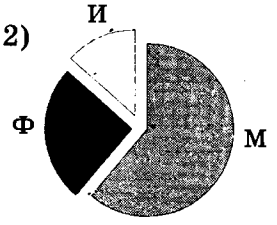
Ответ: вариант №4.

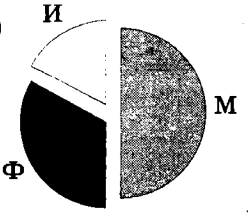
Задача 4*. На диаграмме показано количество призёров олимпиады по информатике (И), математике (М), физике (Ф) в трёх городах России.

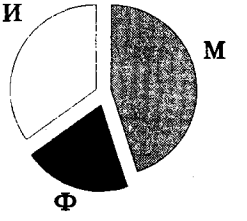


Какая из диаграмм правильно отражает соотношение общего числа призёров по каждому предмету для всех городов вместе?

1) 

2) 

3) 

4) 

Решение

Все необходимые данные имеются на исходной столбиковой диаграмме. Необходимо просуммировать полученные из этой диаграммы значения количеств призёров по информатике (И), математике (М) и по физике (Ф) для всех трёх городов:

М: 180 + 160 + 180 = 520;

Ф: 120 + 140 + 120 = 380;

И: 120 + 60 + 120 = 300.

Общее количество призёров (если считать, что один учащийся может быть призёром только по одному предмету[1]) равно: 520 + 380 + 300 = 1200.

Соотношения (в процентах) количеств призёров по каждому предмету к общему их количеству:

М: 520 / 1200 ≈ 0,43 ≈ 43%;

Ф: 380 / 1200 ≈ 0,32 ≈ 32%;

И: 300 / 1200 = 0,25 = 25%.

Следовательно, на круговой диаграмме сектор И должен точно соответствовать четверти круга, сектор Ф — быть чуть больше четверти, а сектор М — быть чуть меньше половины круга. Этим соотношениям удовлетворяет только диаграмма 1 (на всех других сектор И не равен $^1/_4$ круга).

Ответ: вариант №1.

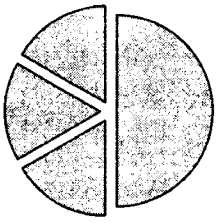
---

[1] Это условие весьма существенно (хотя и не оговорено явно), поскольку его несоблюдение делает задачу неоднозначной.

Задача 5*. Дан фрагмент электронной таблицы:

	A	B	C	D
1	3		3	2
2	=(C1+A1)/2	=C1−D1	=A1−D1	=B1/2

Какое число должно быть записано в ячейке B1, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:D2 соответствовала рисунку:



Решение

Ранее в подобных задачах требовалось по результатам расчётов в таблице указать, какая диаграмма по ней будет построена (и, соответственно, это была задача группы А). Теперь же задача стала сложнее — надо по виду диаграммы (причём «слепой» — без легенды и подписей!) определить недостающее значение в ячейке таблицы.

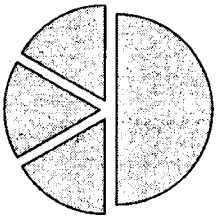
Решение следует начать с обозначения искомого числа в ячейке **B1** как x и выполнения (с учётом этой неизвестной) всех вычислений по предложенным в строке **2** формулам:

	A	B	C	D
1	3	x	3	2
2	(3 + 3) / 2	3 − 2	3 − 2	x/2

или

	A	B	C	D
1	3	x	3	2
2	3	1	1	x/2

Сравнивается полученный набор числовых значений в строке **2** с диаграммой:



На этой диаграмме имеются три равных сектора, в сумме дающих 180°, и значит, равных 60° каждый, и один сектор в 180° (т.е. равный трём остальным вместе). Очевидно, такая ситуация возможна, если большой сектор соответствует значению 3, а малые секторы — каждый значению 1.

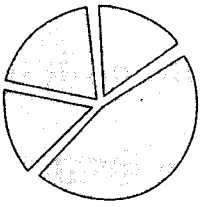
Тогда x / 2 = 1, откуда x = 2.

Ответ: в ячейке B1 должно быть записано число 2.

Задача 6. Дан фрагмент электронной таблицы:

	A	B
1	6	=A1−A2
2		=A3−A2
3	10	=A1/B1
4	18	=B2−B1

Какое число должно быть записано в ячейке **A2**, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек **B1:B4** соответствовала рисунку:



Решение

Эта задача по сути аналогична предыдущей, но предполагает более сложный анализ.

Решение начинается с обозначением искомого числа в ячейке **A2** как x и выполнения (с учётом этой неизвестной) всех вычислений по предложенным в столбце **B** формулам:

	A	B
1	6	=A1–A2
2	x	=A3–A2
3	10	=A1/B1
4	18	=B2–B1

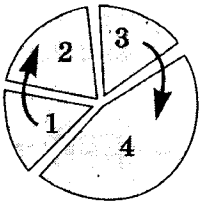
	A	B
1	6	$6 - x$
2	x	$10 - x$
3	10	$6/(6 - x)$
4	18	$(10 - x) - (6 - x)$

	A	B
1	6	$6 - x$
2	x	$10 - x$
3	10	$6/(6 - x)$
4	18	4

На исходной диаграмме имеется сектор, равный $^1/_4$ всего круга; большой сектор, равный почти $^1/_2$ круга (можно предположить, что до 180° ему не хватает 30°); сектор, чуть меньший $^1/_4$ (предположительно — 60°), а также сектор, примерно равный $^1/_8$ круга. Сопоставляются эти секторы ячейкам столбца **B** таблицы, оценивая его ячейки по величине и помня, что в круговой диаграмме секторы строятся по порядку ячеек в столбце сверху вниз (т. е. поняв, какой ячейке соответствует какой-то один сектор, можно сопоставить «своим» секторам и остальные ячейки).

Первый, наиболее «очевидный» способ — вычислить сумму значений во всех ячейках столбца **B**, чтобы узнать значение, которому соответствует весь круг диаграммы, а потом искать, сколько процентов от этой величины соответствуют каждому сектору (значению каждой ячейки столбца **B**), — слишком трудоёмок.

Поэтому можно попробовать решить задачу «обходным» путём. Очевидно, что для любого x будет истинным соотношение: $(6 - x) < (10 - x)$. Определяется, где при «движении» по диаграмме по часовой стрелке происходит переход от меньшего сектора к большему. Таких мест два: переход от сектора, которому условно присвоен номер 1, к сектору 2 и переход от сектора 3 к сектору 4.



1) Пусть верно первое предположение, и соответствие секторов (согласно введёнными номерам) и ячеек следующее:

Сектор	Примерный угол сектора, градусы	Примерная доля сектора, %	Ячейка	Значение в ячейке
1	45	12,5	B1	$6 - x$
2	90	25	B2	$10 - x$
3	60	16,6	B3	$6/(6 - x)$
4	150	41,6	B4	4

В этом случае, если самый большой сектор соответствует 4 единицам, то значение ячейки **B1** должно быть почти втрое меньше, т. е. $3 \cdot (6 - x) = 4 \Rightarrow 18 - 3x = 4 \Rightarrow 3x = 14 \Rightarrow x \approx 5$. Тогда соотношение величин секторов будет таким:

Сектор	Примерная доля сектора, %	Ячейка	Значение в ячейке	Примерная расчётная доля сектора, %
1	12,5	B1	6 – 5 = 1	6,25
2	25	B2	10 – 5 = 5	31,25
3	16,6	B3	6/(6 – 5) = 6	37,5
4	41,6	B4	4	25

Очевидно, что полученное соотношение долей секторов не соответствует диаграмме. Тогда проверяется второе предположение:

Сектор	Примерный угол сектора, градусы	Примерная доля сектора, %	Ячейка	Значение в ячейке
3	60	16,6	B1	$6 - x$
4	150	41,6	B2	$10 - x$
1	45	12,5	B3	$6/(6 - x)$
2	90	25	B4	4

В этом случае сектор 2, составляющий 25% долю всей диаграммы, соответствует 4 единицам. Тогда предшествующий ему сектор 1 (вдвое меньший) должен соответствовать 2 единицам: $6 / (6 - x) = 2 \Rightarrow x = 3$. Снова проверяется соотношение секторов:

Сектор	Примерная доля сектора, %	Ячейка	Значение в ячейке	Примерная расчётная доля сектора, %
3	16,6	B1	6 – 3 = 3	18,75
4	41,6	B2	10 – 3 = 7	43,75
1	12,5	B3	6/(6 – 3) = 2	12,5
2	25	B4	4	25

В данном случае для секторов 1 и 2 вычисленные значения долей в процентах совпали с оцененными «на глазок», а для секторов 3 и 4 получились значения, очень близкие к предварительно оцененным и, во всяком случае, верно отражающие примерное соотношение величин секторов.

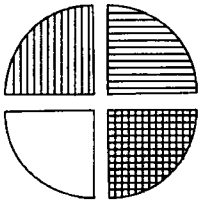
Ответ: в ячейке A2 должно быть записано значение 3.

Задачи для самостоятельного решения

1. Дан фрагмент электронной таблицы:

	A	B	C	D
1	1		2	3
2	=C1+D1	=B1−C1	=A2+C1−2·A1	=3·C1−A1

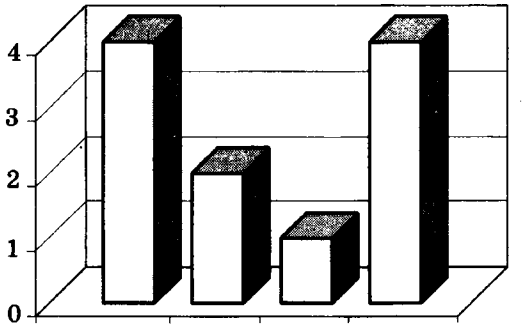
Какое число должно быть записано в ячейке B1, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:D2 соответствовала рисунку?



2. Дан фрагмент электронной таблицы:

	A	B	C	D
1	4	8	2	
2	=(B1−A1)/2+C1	=B1/A1	=D1−A2	=2·B2

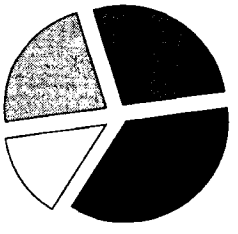
Какое число должно быть записано в ячейке D1, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:D2 соответствовала рисунку?



3. Дан фрагмент электронной таблицы:

	A	B	C	D
1		2	1	5
2	=A1+A3	=C1+C3	=B1+B3	=D1+D3
3	4	1	6	1

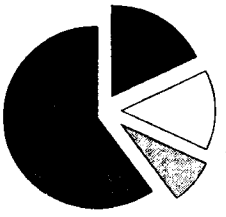
Какое число должно быть записано в ячейке A1, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:D2 соответствовала рисунку?



4. Дан фрагмент электронной таблицы:

	A	B	C	D
1	3	2	1	4
2	=СУММ(A1:D1)	=A3+C3	=СРЗНАЧ(A1:C1)	=B3*D3
3	4	2	6	

Какое число должно быть записано в ячейке D3, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:D2 соответствовала рисунку?

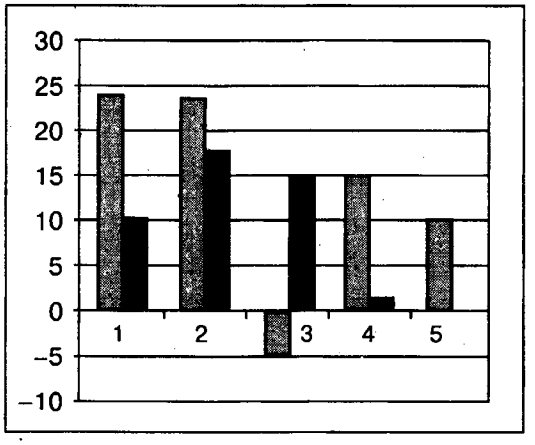


5. Дан фрагмент электронной таблицы:

	A	B	C	D	E
1	=A2+A4				
2	17	21	5	8	5
3	=A2−A4				
4	7	3		7	5

Формулу из ячейки A1 распространили путём автозаполнения вправо на ячейки B1:E1. Формулу из ячейки A3 аналогичным образом распространили вправо на ячейки B3:E3.

Какое число должно быть записано в ячейке C4, чтобы построенная после выполнения вычислений диаграмма по значениям диапазонов ячеек A1:E1 и A3:E3 соответствовала рисунку?



Ответы для самопроверки

Задача	Ответ
1	7
2	5
3	3
4	19
5	−10

Технологии поиска и хранения информации

A6 Базы данных. Сортировка данных. Запросы в базах данных

Конспект

Основой базы данных (БД) является **таблица**.

Поля БД — это характеристики объектов (сущностей), информация о которых хранится в БД. Поля БД соответствуют столбцам таблицы.

Записи БД — это информация о каждом из объектов (одному объекту соответствует одна запись), выраженная в виде значений соответствующих полей. Записям БД соответствуют строки таблицы.

Поля / Записи	Поле 1	Поле 2	...	Поле *n*
Запись 1				
Запись 2				
...				
Запись *m*				

Характеристики, отражённые в виде полей БД, являются едиными (общими) для всех объектов. Объекты в БД должны различаться хотя бы одним значением какой-либо характеристики.

Ключевое поле — поле БД, значения которого гарантированно различаются для разных объектов. По значению ключевого поля всегда можно однозначно выделить соответствующий объект.

Выборка данных из БД — операция отбора записей БД (строк таблицы), соответствующих заданному условию (**запросу на выборку**).

Условие (запрос) может быть **простым** (накладывается на значения какого-то одного поля либо выражено в сравнении двух каких-либо полей) или **составным** (простые условия объединяются при помощи логических операций **И, ИЛИ, НЕ**).

Реляционная БД состоит из нескольких взаимосвязанных таблиц. Связи («реляции») между двумя какими-либо таблицами осуществляются через общее для них по смыслу (не обязательно одинаковое по названию) поле. При этом возможны связи:

- «один к одному» — одной записи первой таблицы соответствует одна и только одна запись второй таблицы, и наоборот (пример: в ОС MS-DOS полному имени файла однозначно соответствует запись номера начального кластера);
- «один ко многим» — одной записи первой таблицы может соответствовать много записей второй таблицы или наоборот (пример: один и тот же учитель может вести уроки в нескольких классах);

- «многие ко многим» — много записей в первой таблице могут быть связаны с многими записями второй таблицы (пример: одного и того же ученика могут учить разные учителя, а один и тот же учитель может учить множество учеников). Подобный тип связей в реляционных БД не допускается и при необходимости реализуется как две связи «один ко многим» через промежуточную таблицу (в приведённом только что примере учитель связывается с учеником через номер класса и предмет).

Поиск данных в реляционной БД требует перехода от одной БД к другой в соответствии с имеющимися связями (в том числе — многократно).

Разбор типовых задач

Задача 1*. На городской олимпиаде по программированию предлагались задачи трёх типов: A, B и C. По итогам олимпиады была составлена таблица, в колонках которой указано, сколько задач каждого типа решил участник. Вот начало таблицы:

Фамилия	A	B	C
Иванов	3	2	1

За правильное решение задачи типа A участнику начислялся 1 балл, за решение задачи типа B — 2 балла и за решение задачи типа C — 3 балла. Победитель определялся по сумме баллов, которая у всех участников оказалась разная. Для определения победителя олимпиады достаточно выполнить следующий запрос:

1) Отсортировать таблицу по возрастанию значения поля C и взять первую строку.
2) Отсортировать таблицу по убыванию значения поля C и взять первую строку.
3) Отсортировать таблицу по убыванию значения выражения A + 2B + 3C и взять первую строку.
4) Отсортировать таблицу по возрастанию значения выражения A + 2B + 3C и взять первую строку.

Решение

Поскольку победитель определялся по сумме баллов, для определения победителя необходимо отсортировать таблицу по значению A + 2B + 3C, где A, B и C — количества решенных задач. Этому соответствуют варианты 3 и 4. Чтобы имя победителя (с максимальной суммой баллов) оказалось в первой строке таблицы, нужно отсортировать её по убыванию.

Ответ: отсортировать таблицу по убыванию значения выражения A + 2B + 3C и взять первую строку 3 (вариант ответа №3).

Задача 2*. Результаты тестирования представлены в таблице:

Фамилия	Пол	Математика	Русский язык	Химия	Информатика	Биология
Аганян	ж	82	56	46	32	70
Воронин	м	43	62	45	74	23
Григорчук	м	54	74	68	75	83
Роднина	ж	71	63	56	82	79
Сергеенко	ж	33	25	74	38	46
Черепанова	ж	18	92	83	28	61

Сколько записей в ней удовлетворяют условию **«Пол='ж' ИЛИ Химия>Биология»**?

1) 5 2) 2 3) 3 4) 4

Решение (способ 1)

1. В таблице помечаются (например, обводя карандашом) ячейки, соответствующие условию **Пол='ж'**:

Фамилия	Пол	Математика	Русский язык	Химия	Информатика	Биология
Аганян	(ж)	82	56	46	32	70
Воронин	м	43	62	45	74	23
Григорчук	м	54	74	68	75	83
Роднина	(ж)	71	63	56	82	79
Сергеенко	(ж)	33	25	74	38	46
Черепанова	(ж)	18	92	83	28	61

2. Отдельно помечаются в таблице (например, обводя карандашом) ячейки, соответствующие условию **Химия>Биология** (имеется в виду сравнение соответствующих числовых значений):

Фамилия	Пол	Математика	Русский язык	Химия	Информатика	Биология
Аганян	(ж)	82	56	46	32	70
Воронин	м	43	62	(45)	74	(23)
Григорчук	м	54	74	68	75	83
Роднина	(ж)	71	63	56	82	79
Сергеенко	(ж)	33	25	(74)	38	(46)
Черепанова	(ж)	18	92	(83)	28	(61)

3. Условия объединены логической операцией **ИЛИ**. Поэтому выбираются из таблицы строки, в которых отмечена обводкой хотя бы одна ячейка:

Фамилия	Пол	Математика	Русский язык	Химия	Информатика	Биология
Аганян	(ж)	82	56	46	32	70
Воронин	м	43	62	(45)	74	(23)
Григорчук	м	54	74	68	75	83
Роднина	(ж)	71	63	56	82	79
Сергеенко	(ж)	33	25	(74)	38	(46)
Черепанова	(ж)	18	92	(83)	28	(61)

Ответ: 5 строк (вариант ответа №1).

Решение (способ 2)

1. Для логических значений x и y выполняется соотношение: $x \vee y = \overline{\bar{x} \wedge \bar{y}}$. То есть исходное условие фильтрации **«Пол='ж' ИЛИ Химия>Биология»** можно заменить на эквивалентное условие **«НЕ(Пол<>'ж' И Химия<=Биология)»**, либо провести отбор строк по условию **«Пол='м' И Химия<=Биология»**. В этом случае можно сразу пропустить в таблице строки с буквой «ж» в графе «Пол».

2. Для оставшихся строк **3** и **4** проверяется второе условие. Ему соответствует единственная строка:

Фамилия	Пол	Математика	Русский язык	Химия	Информатика	Биология
~~Аганян~~	~~ж~~	~~82~~	~~56~~	~~46~~	~~32~~	~~70~~
Воронин	м	43	62	45	74	23
Григорчук	м	54	74	68	75	83
~~Роднина~~	~~ж~~	~~71~~	~~63~~	~~56~~	~~82~~	~~79~~
~~Сергеенко~~	~~ж~~	~~33~~	~~25~~	~~74~~	~~38~~	~~46~~
~~Черепанова~~	~~ж~~	~~18~~	~~92~~	~~83~~	~~28~~	~~61~~

3. Таким образом, обратному условию соответствует только одна строка. Всего в таблице 6 строк, следовательно, исходному условию (противоположному по смыслу) соответствуют 5 оставшихся строк.

Ответ: 5 строк (вариант ответа №1).

Решение (способ 3)

При помощи Excel решение данной задачи возможно при помощи опции «Расширенный фильтр» (меню **Данные → Фильтр → Расширенный фильтр**). Однако эта опция не позволяет задать в качестве условия фильтрации сравнение значений разных столбцов в каждой строке по отдельности (можно сравнивать значения в каждом из участвующих в фильтрации столбцов с константой или со значением, однократно вычисленным по некоторой формуле). Поэтому предварительно нужно добавить в таблицу ещё один столбец, содержащий разность значений ячеек в столбцах «Химия» и «Биология».

Далее нужно создать рядом с таблицей диапазон условий фильтрации по равенству значений ячеек столбца «Пол» константе «ж» или для дополнительного столбца разности значений — по признаку «значение ячейки больше нуля». При этом (согласно правилам построения диапазона условий для «Расширенного фильтра» Excel) для объединения двух указанных элементарных условий логической связкой ИЛИ (а не И) нужно ввести требуемые константы в *разных* строках диапазона условий.

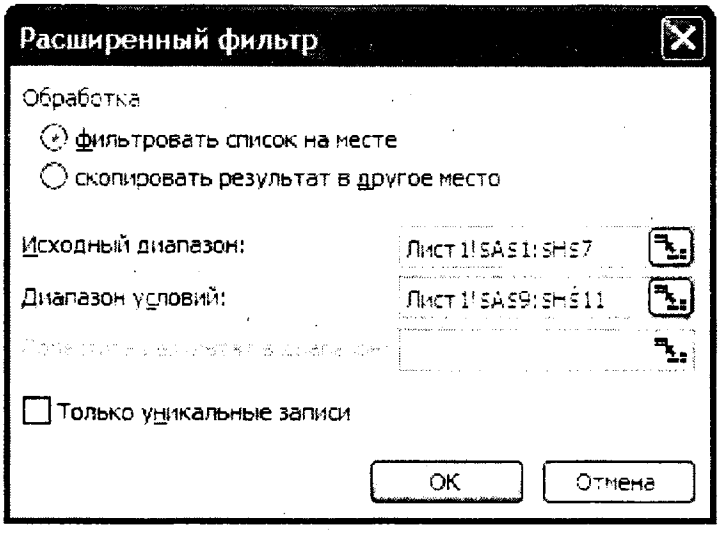
Формулы в столбце разности ячеек «Химия» и «Биология»

Дополнительный столбец

H2 ▼ fx =E2-G2

	A	B	C	D	E	F	G	H
1	Фамилия	Пол	Математика	Русский язык	Химия	Информатика	Биология	*Химия минус Биология*
2	Аганян	ж	82	56	46	32	70	*-24*
3	Воронин	м	43	62	45	74	23	*22*
4	Григорчук	м	54	74	68	75	83	*-15*
5	Роднина	ж	71	63	56	82	79	*-23*
6	Сергеенко	ж	33	25	74	38	46	*28*
7	Черепанова	ж	18	92	83	28	61	*22*
8								
9		*Пол*						*Химия минус Биология*
10		ж						
11								*>0*

Диапазон условий фильтрации



Результирующая таблица:

	A	B	C	D	E	F	G	H
1	**Фамилия**	**Пол**	**Математика**	**Русский язык**	**Химия**	**Информатика**	**Биология**	***Химия минус Биология***
2	Аганян	ж	82	56	46	32	70	*-24*
3	Воронин	м	43	62	45	74	23	*22*
4	Григорчук	м	54	74	**68**	75	**83**	*-15*
5	Роднина	ж	71	63	56	82	79	*-23*
6	Сергеенко	ж	33	25	74	38	46	*28*
7	Черепанова	ж	18	92	83	28	61	*22*

Здесь строки **2, 5, 6 и 7** включены в выборку как минимум по значению пола «ж», а строка **3** — потому, что значение соответствующей ячейки в дополнительном столбце разности — положительное (а значит, значение ячейки столбца «Химия» больше значения ячейки столбца «Биология»).

Что же касается пропущенной (не вошедшей в эту выборку) строки **4**:

4	Григорчук	м	54	74	**68**	75	**83**	*-15*

то в ней и значение ячейки в столбце «Пол» не равно «ж», и значение ячейки в столбце разности — отрицательное, т. е. не выполняются оба условия фильтрации.

Ответ: 5 строк (вариант ответа №1).

Задача 3*. В фрагменте базы данных представлены сведения о родственных отношениях. Определите на основании приведённых данных фамилию и инициалы бабушки Ивановой А.И.

1) Иванов Т.М.
2) Черных И.А.
3) Цейс Т.Н.
4) Петренко Н.Н.

Таблица 1

ID	Фамилия_И.О.	Пол
71	Иванов Т.М.	М
85	Петренко И.Т.	М
13	Черных И.А.	Ж
42	Петренко А.И.	Ж
23	Иванова А.И.	Ж
96	Петренко Н.Н.	Ж
82	Черных А.Н.	М
95	Цейс Т.Н.	Ж
10	Цейс Н.А.	М
	...	

Таблица 2

ID_Родителя	ID_Ребёнка
23	71
13	23
85	23
82	13
95	13
85	42
82	10
95	10
...	...

Решение

Первое, на что здесь нужно обратить внимание: каждому человеку, чьи ФИО и пол отражены в этой базе данных в таблице 1, присвоен конкретный индивидуальный номер.

Вторая особенность: таблица 2 определяет родственные связи между людьми (закодированными их индивидуальными номерами) по принципу «родитель — ребёнок».

Исходное лицо: *Иванова А.И.*

1) Ищется запись о ней в таблице 1 (проверяя, что значение поля «Пол» для неё должно быть равно «Ж») и определяется её индивидуальный номер: **23**.

Таблица 1

ID	Фамилия_И.О.	пол
71	Иванов Т.М.	М
85	Петренко И.Т.	М
13	Черных И.А.	Ж
42	Петренко А.И.	Ж
23	**Иванова А.И.**	**Ж**
96	Петренко Н.Н.	Ж
82	Черных А.Н.	М
95	Цейс Т.Н.	Ж
10	Цейс Н.А.	М
	...	

Таблица 2

ID_Родителя	ID_Ребёнка
23	71
13	23
85	23
82	13
95	13
85	42
82	10
95	10
...	...

2) Определяется по таблице 2, кто является родителем человека с найденным индивидуальным номером 23. Для этого ищутся в таблице 2 все строки с номером 23 в поле «ID_Ребёнка» и определяется, какие индивидуальные номера соответствуют в найденных записях полю «ID_Родителя». Это номера **13** и **85**: очевидно, мать и отец Ивановой А.И. (Вернувшись к таблице 1, по найденным номерам можно определить по значению поля «Пол», кто есть кто, хотя для решения данной задачи это не требуется.)

Таблица 1

ID	Фамилия_И.О.	Пол
71	Иванов Т.М.	М
85	Петренко И.Т.	М
13	Черных И.А.	Ж
42	Петренко А.И.	Ж
23	Иванова А.И.	Ж
96	Петренко Н.Н.	Ж
82	Черных А.Н.	М
95	Цейс Т.Н.	Ж
10	Цейс Н.А.	М
...		

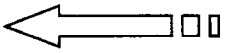


Таблица 2

ID_Родителя	ID_Ребёнка
23	71
13	23
85	23
82	13
95	13
85	42
82	10
95	10
...	...

3) Определили родителей Ивановой А.И., но по условию задачи требуется найти, кто является её бабушкой. Бабушка — это (в контексте рассматриваемого задания) мать одного из родителей Ивановой А.И., поэтому вышеописанную операцию поиска ID родителей по ID детей надо повторить для найденных на предыдущем шаге матери и отца Ивановой А.И.

- В таблице 2 в поле «ID_Ребёнка» ищется индивидуальный номер 85. Поскольку такое значение в указанном поле отсутствует, поиск по данной ветви генеалогического древа можно прекратить.
- В таблице 2 в поле «ID_Ребёнка» ищется индивидуальный номер 13 и определяются все значения индивидуальных номеров в поле «ID_Родителя», соответствующих указанному значению поля «ID_Ребёнка». Это номера 82 и 95 (мать и отец матери Ивановой А.И., т. е. её дед и искомая бабушка).

Таблица 1

ID	Фамилия_И.О.	Пол
71	Иванов Т.М.	М
85	Петренко И.Т.	М
13	Черных И.А.	Ж
42	Петренко А.И.	Ж
23	Иванова А.И.	Ж
96	Петренко Н.Н.	Ж
82	Черных А.Н.	М
95	Цейс Т.Н.	Ж
10	Цейс Н.А.	М
...		

Таблица 2

ID_Родителя	ID_Ребёнка
23	71
13	23
85	23
82	13
95	13
85	42
82	10
95	10
...	...

4) Найдя ID деда и бабушки, в таблице 1 по этим номерам и значениям поля «Пол» определяется, кто из них кто.

Таблица 1

ID	Фамилия_И.О.	Пол
71	Иванов Т.М.	М
85	Петренко И.Т.	М
13	Черных И.А.	Ж
42	Петренко А.И.	Ж
23	Иванова А.И.	Ж
96	Петренко Н.Н.	Ж
82	Черных А.Н.	М
95	Цейс Т.Н.	Ж
10	Цейс Н.А.	М
...		

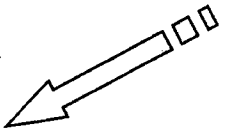


Таблица 2

ID_Родителя	ID_Ребёнка
23	71
13	23
85	23
82	13
95	13
85	42
82	10
95	10
...	...

Очевидно, что бабушкой Ивановой А.И. является Цейс Т.Н.

Ответ: Цейс Т.Н. (вариант ответа №3).

Задача 4. Дан фрагмент базы данных некоторого образовательного учреждения. Все объекты в этой базе имеют свой идентификационный код (4-значное число в 16-ричной системе счисления). Определите на основании приведённых данных № группы, в которой учится Смирнова Ю.

Таблица 1

Учащийся	ID
Бердыев А.	B8FE
Зинатуллина А.	2969
Круглова С.	F719
Кузнецов Ю.	34F4
Лебедева А.	9829
Лобчиков В.	4BF1
Морозова А.	0118
Мохначева А.	BFCE
Петрова А.	E641
Смирнова Ю.	156D
Тамкова В.	DDC8
Храповский М.	F46C
Чертакова Д.	F045
Шопша Н.	E8AC
Шубина Ж.	96F0
Щербакова Е.	E63D
...	...

Таблица 2

№ группы	ID
Группа 1	3D95
Группа 2	67BA
Группа 3	3668
Группа 4	5D6B
...	...

Таблица 3

ID_группы	ID_учащегося
3668	F46C
3668	96F0
3668	F8AC
3D95	F719
3D95	34F4
3D95	BFCE
3D95	2969
3D95	DDC8
5D6B	B8FE
5D6B	156D
5D6B	E641
5D6B	0118
67BA	F045
67BA	4BF1
67BA	9829
67BA	E63D

1) 1 2) 2 3) 3 4) 4

Решение

Хотя здесь речь идёт не о родственниках, принцип решения тот же.

Исходное лицо: *Смирнова Ю.*

1) По заданной фамилии и имени студентки определяется в таблице 1 её уникальный идентификационный номер: **156D**.

2) Определив ID учащегося, ищем его в соответствующем поле таблице 3, отражающей взаимосвязь кодов учащихся и учебных групп (внимание: одному коду группы может соответствовать много кодов учащихся, но каждому коду учащегося соответствует только один код группы; в предыдущей задаче одному коду ребёнка соответствовали один или два кода его родителей). Искомый код группы равен **5D6B**.

3) Обратившись к таблице 2, где сопоставлены коды групп и их порядковые номера, определяется искомый номер группы. Он равен **4**.

Таблица 1

Учащийся	ID
Бердыев А.	B8FE
Зинатуллина А.	2969
Круглова С.	F719
Кузнецов Ю.	34F4
Лебедева А.	9829
Лобчиков В.	4BF1
Морозова А.	0118
Мохначева А.	BFCE
Петрова А.	E641
Смирнова Ю.	156D
Тамкова В.	DDC8
Храповский М.	F46C
Чертакова Д.	F045
Шопша Н.	E8AC
Шубина Ж.	96F0
Щербакова Е.	E63D
...	...

Таблица 2

№ группы	ID
Группа 1	3D95
Группа 2	67BA
Группа 3	3668
Группа 4	5D6B
...	...

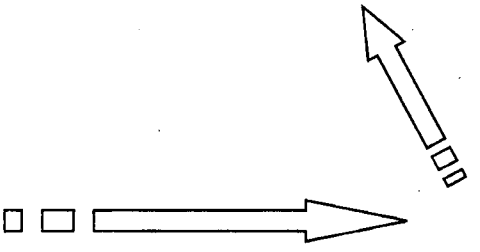


Таблица 3

ID_группы	ID_учащегося
3668	F46C
3668	96F0
3668	F8AC
3D95	F719
3D95	34F4
3D95	BFCE
3D95	2969
3D95	DDC8
5D6B	B8FE
5D6B	156D
5D6B	E641
5D6B	0118
67BA	F045
67BA	4BF1
67BA	9829
67BA	E63D

Ответ: группа 4 (вариант ответа №4).

Задача 5. Дан фрагмент базы данных некоторого образовательного учреждения. Все объекты в этой базе имеют свой идентификационный код (4-значное число в 16-ричной системе счисления). Определите на основании приведённых данных, какую отметку по физике имеет Кузнецов Ю.

Таблица 1

Учащийся	ID
Круглова С.	F719
Кузнецов Ю.	34F4
Лебедева А.	9829
Лобчиков В.	4BF1
Морозова А.	0118
Мохначева А.	BFCE
Петрова А.	E641
Смирнова Ю.	156D
...	...

Таблица 2

Предмет	ID
Русский язык	2969
Математика	DDC8
Физика	B8FE
Химия	156D
...	...

Таблица 3

ID_учащегося	ID_предмета	Отметка
9829	B8FE	5
BFCE	B8FE	3
F719	B8FE	4
E641	B8FE	5
E641	156D	5
4BF1	DDC8	2
0118	DDC8	4
E641	2969	5
BFCE	DDC8	3
34F4	B8FE	3
4BF1	2969	2
156D	2969	2
4BF1	B8FE	2
9829	2969	5
34F4	156D	3
4BF1	156D	2
156D	DDC8	2
0118	2969	4
156D	B8FE	2
F719	156D	4
F719	DDC8	4
BFCE	156D	3
0118	B8FE	4
E641	DDC8	5
156D	156D	2
0118	156D	4
9829	156D	5
F719	2969	4
BFCE	2969	3
9829	DDC8	5
34F4	2969	3
34F4	DDC8	3
...	...	...

1) 2 2) 3 3) 4 4) 5

Решение

Исходное лицо: *Кузнецов Ю.*

1) По таблице 1 определяется уникальный идентификационный номер Кузнецова Ю.: это код **34F4**.

2) Аналогично, по таблице 2 определяется идентификационный номер предмета «физика». Это код **B8FE**.

3) Зная коды учащегося и предмета, ищется в таблице 3 значение оценки. Внимание: в этой таблице одному коду учащегося может соответствовать несколько кодов учебных предметов и наоборот, одному коду учебного предмета может соответствовать несколько кодов учащихся, но *каждое сочетание кода учащегося и кода учебного предмета является уникальным*. Очевидно, ранее определённым кодам учащегося и предмета соответствует запись, содержащая в поле «Отметка» значение **3**.

Таблица 1

Учащийся	ID
Круглова С.	F719
Кузнецов Ю.	34F4
Лебедева А.	9829
Лобчиков В.	4BF1
Морозова А.	0118
Мохначева А.	BFCE
Петрова А.	E641
Смирнова Ю.	156D
...	...

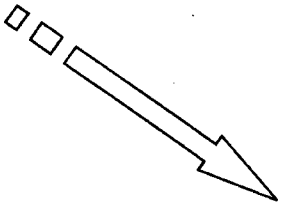


Таблица 2

Предмет	ID
Русский язык	2969
Математика	DDC8
Физика	B8FE
Химия	156D
...	...

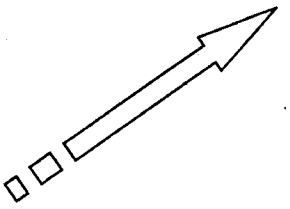


Таблица 3

ID_учащегося	ID_предмета	Отметка
9829	B8FE	5
BFCE	B8FE	3
F719	B8FE	4
E641	B8FE	5
E641	156D	5
4BF1	DDC8	2
0118	DDC8	4
E641	2969	5
BFCE	DDC8	3
34F4	B8FE	3
4BF1	2969	2
156D	2969	2
4BF1	B8FE	2
9829	2969	5
34F4	156D	3
4BF1	156D	2
156D	DDC8	2
0118	2969	4
156D	B8FE	2
F719	156D	4
F719	DDC8	4
BFCE	156D	3
0118	B8FE	4
E641	DDC8	5
156D	156D	2
0118	156D	4
9829	156D	5
F719	2969	4
BFCE	2969	3
9829	DDC8	5
34F4	2969	3
34F4	DDC8	3
...	...	...

Ответ: отметка 3 (вариант ответа №2).

Задача 6. В фрагменте базы данных представлены сведения о родственных отношениях. Определите на основании приведённых данных фамилию и инициалы племянника Черных Н.И.

Примечание: племянник — сын сестры или брата.

Таблица 1

ID	Фамилия_И.О.	Пол
85	Гуревич И.Т.	М
82	Гуревич А.И.	М
42	Цейс А.Т.	Ж
71	Петров Т.М.	М
23	Петров А.Т.	М
13	Цейс И.И.	Ж
95	Черных Т.Н.	Ж
10	Черных Н.И.	М
...		

Таблица 2

ID_Родителя	ID_Ребёнка
95	82
85	13
71	42
85	82
13	42
71	23
13	23
95	13
85	10
...	...

1) Петров А.Т. 2) Петров Т.М. 3) Гуревич А.И. 4) Гуревич И.Т.

Решение

Эта задача аналогична по формулировке и принципу решения ранее рассмотренной задаче 3, однако она заметно сложнее, поскольку её решение не так «прямолинейно».

Исходное лицо: *Черных Н.И.*

1) По таблице 1 определяется уникальный идентификационный номер Черных Н.И. Он равен **10**.

Таблица 1

ID	Фамилия_И.О.	Пол
85	Гуревич И.Т.	М
82	Гуревич А.И.	М
42	Цейс А.Т.	Ж
71	Петров Т.М.	М
23	Петров А.Т.	М
13	Цейс И.И.	Ж
95	Черных Т.Н.	Ж
10	**Черных Н.И.**	М
...		

Таблица 2

ID_Родителя	ID_Ребёнка
95	82
85	13
71	42
85	82
13	42
71	23
13	23
95	13
85	10
...	...

2) Определяется племянник Черных Н.И. Но в таблице 2 отражены родственные связи только типа «родитель — ребёнок», а нужно выявить родственную связь «сын сестры / брата». Как это сделать?

Нужно догадаться, что сестра или брат являются детьми тех же самых родителей, что и интересующего лица́. Поэтому, зная ID исходного человека (10), ищутся в таблице 2 все строки, для которых в поле «ID_Ребёнка» записан код 10. Такая запись в таблице 2 одна и содержит в поле «ID_Родителя» идентификационный номер **85**. (Вернувшись к таблице 1, можно по этому идентификатору определить, что речь идёт об отце Черных Н.И. — Гуревиче И.Т.)

Таблица 1

ID	Фамилия_И.О.	Пол
85	Гуревич И.Т.	М
82	Гуревич А.И.	М
42	Цейс А.Т.	Ж
71	Петров Т.М.	М
23	Петров А.Т.	М
13	Цейс И.И.	Ж
95	Чёрных Т.Н.	Ж
10	Черных Н.И.	М
...		

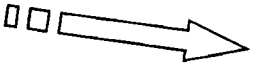


Таблица 2

ID_Родителя	ID_Ребёнка
95	82
85	13
71	42
85	82
13	42
71	23
13	23
95	13
85	10
...	...

3) Теперь нужно (опять-таки по таблице 2) определить для найденного ID родителя идентификационные номера *всех* его детей. Для этого в таблице 2 ищутся все записи (строки), где в поле «ID_Родителя» записан код 85. Соответствующие ID детей равны: **13, 82 и 10**.

Таблица 1

ID	Фамилия_И.О.	Пол
85	Гуревич И.Т.	М
82	Гуревич А.И.	М
42	Цейс А.Т.	Ж
71	Петров Т.М.	М
23	Петров А.Т.	М
13	Цейс И.И.	Ж
95	Черных Т.Н.	Ж
10	Черных Н.И.	М
...		

Таблица 2

ID_Родителя	ID_Ребёнка
95	82
85	13
71	42
85	82
13	42
71	23
13	23
95	13
85	10
...	...

4) Возвращаясь к таблице 1, для найденных идентификаторов определяется информация о людях (ФИО и пол):

- код 13 — Цейс И.И. (пол «Ж»);
- код 82 — Гуревич А.И. (пол «М»);
- код 10 — Черных Н.И. (пол «М»).

Из найденных людей Черных Н.И. — исходное лицо. Он из рассмотрения исключается.

Таблица 1

ID	Фамилия_И.О.	Пол
85	Гуревич И.Т.	М
82	**Гуревич А.И.**	М
42	Цейс А.Т.	Ж
71	Петров Т.М.	М
23	Петров А.Т.	М
13	**Цейс И.И.**	Ж
95	Черных Т.Н.	Ж
10	**Черных Н.И.**	М
…		

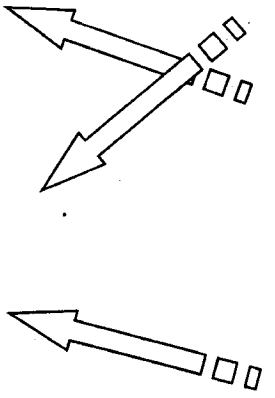


Таблица 2

ID_Родителя	ID_Ребёнка
95	82
85	13
71	42
85	82
13	42
71	23
13	23
95	13
85	10
…	…

5) После проделанных операций определены брат и сестра Черных Н.И. Теперь нужно найти племянника Черных Н.И., зная, что это сын его брата или сестры. Для этого вновь в таблице 2 ищутся записи (строки), где в поле «ID_Родителя» записано значение 13 или 82. (Записи для кода 82 в таблице отсутствуют, что облегчает задачу.) Таких записей две, в них в поле «ID_Ребёнка» записаны значения кодов **42** и **23**.

Таблица 1

ID	Фамилия_И.О.	Пол
85	Гуревич И.Т.	М
82	Гуревич А.И.	М
42	Цейс А.Т.	Ж
71	Петров Т.М.	М
23	Петров А.Т.	М
13	**Цейс И.И.**	Ж
95	Черных Т.Н.	Ж
10	Черных Н.И.	М
…		

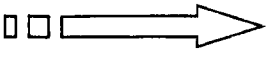


Таблица 2

ID_Родителя	ID_Ребёнка
95	82
85	13
71	42
85	82
13	**42**
71	23
13	**23**
95	13
85	10
…	…

6) Наконец, снова вернувшись к таблице 1, для найденных кодов определяются ФИО и пол соответствующих лиц:

- код 42 — Цейс А.Т. (пол «Ж»);
- код 23 — Петров А.Т. (пол «М»).

Таблица 1

ID	Фамилия_И.О.	Пол
85	Гуревич И.Т.	М
82	Гуревич А.И.	М
42	Цейс А.Т.	Ж
71	Петров Т.М.	М
23	Петров А.Т.	М
13	Цейс И.И.	Ж
95	Черных Т.Н.	Ж
10	Черных Н.И.	М
...		

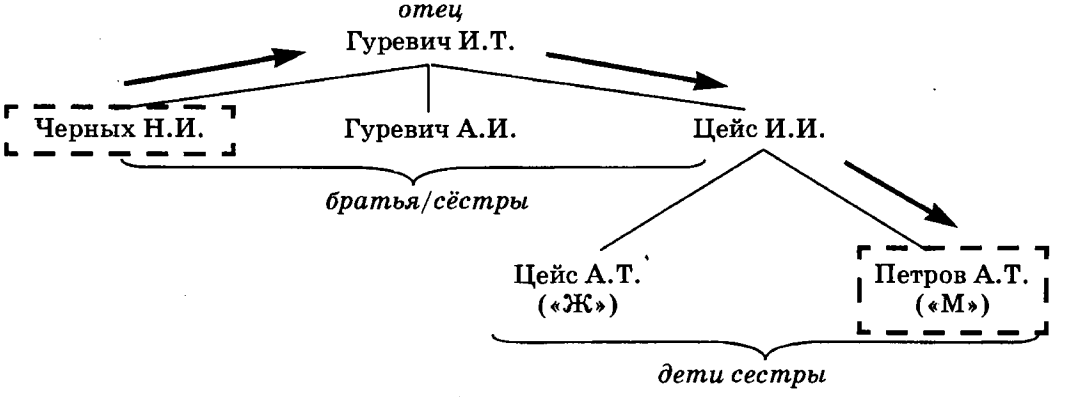
Таблица 2

ID_Родителя	ID_Ребёнка
95	82
85	13
71	42
85	82
13	42
71	23
13	23
95	13
85	10
...	...

Из указанных лиц Цейс А.Т. не подходит, поскольку по условию задачи нужно было определить племянника, а не племянницу. Поэтому единственный человек, подходящий в качестве решения задачи, — это Петров А.Т.

Ответ: Петров А.Т. (вариант ответа №1).

Примечание. Для наглядности можно изобразить «генеалогическое древо» всех рассмотренных в задаче лиц с указанием стрелками пути поиска племянника.



Задача 7. В фрагменте базы данных представлены сведения о родственных отношениях. Определите на основании приведённых данных фамилию и инициалы племянника Симоняна Н.И.

Примечание: племянник — сын сестры или брата.

Таблица 1

ID	Фамилия_И.О.	Пол
86	Седых И.Т.	М
83	Седых А.И.	М
50	Силис А.Т.	Ж
79	Симонов Т.М.	М
23	Симонов А.Т.	М
13	Силис И.И.	Ж
98	Симонян Т.Н.	Ж
11	Симонян Н.И.	М
...		

Таблица 2

ID_Родителя	ID_Ребёнка
98	83
86	13
79	50
86	83
13	50
79	23
13	23
98	13
86	11
...	...

1) Седых А.И. 2) Седых И.Т. 3) Симонов А.Т. 4) Симонов Т.М.

Решение

Исходное лицо: *Симонян Н.И.*

1) По таблице 1 определяется уникальный идентификационный номер Симоняна Н.И. Он равен **11**.

Таблица 1

ID	Фамилия_И.О.	Пол
86	Седых И.Т.	М
83	Седых А.И.	М
50	Силис А.Т.	Ж
79	Симонов Т.М.	М
23	Симонов А.Т.	М
13	Силис И.И.	Ж
98	Симонян Т.Н.	Ж
11	Симонян Н.И.	М
…		

Таблица 2

ID_Родителя	ID_Ребёнка
98	83
86	13
79	50
86	83
13	50
79	23
13	23
98	13
86	11
…	…

2) Для поиска сестры или брата Симоняна Н.И. в таблице 2 ищутся записи о его родителях (где в поле «ID_Ребёнка» записан код 11). Такая запись в таблице 2 одна и содержит в поле «ID_Родителя» идентификационный номер **86**. (Вернувшись к таблице 1, можно по этому идентификатору определить, что речь идёт об отце Симоняна Н.И. — Седых И.Т.)

Таблица 1

ID	Фамилия_И.О.	Пол
86	Седых И.Т.	М
83	Седых А.И.	М
50	Силис А.Т.	Ж
79	Симонов Т.М.	М
23	Симонов А.Т.	М
13	Силис И.И.	Ж
98	Симонян Т.Н.	Ж
11	Симонян Н.И.	М
…		

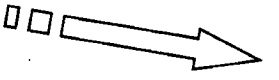


Таблица 2

ID_Родителя	ID_Ребёнка
98	83
86	13
79	50
86	83
13	50
79	23
13	23
98	13
86	11
…	…

3) Теперь (опять по таблице 2) определяются для найденного ID родителя идентификационные номера всех его детей. Для этого в таблице 2 ищутся все записи (строки), где в поле «ID_Родителя» записан код 86. Соответствующие ID детей равны: **13**, **82** и **10**.

Таблица 1

ID	Фамилия_И.О.	Пол
86	Седых И.Т.	М
83	Седых А.И.	М
50	Силис А.Т.	Ж
79	Симонов Т.М.	М
23	Симонов А.Т.	М
13	Силис И.И.	Ж
98	Симонян Т.Н.	Ж
11	Симонян Н.И.	М
…		

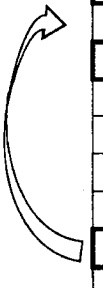


Таблица 2

ID_Родителя	ID_Ребёнка
98	83
86	**13**
79	50
86	**83**
13	50
79	23
13	23
98	13
86	**11**
…	…

4) В таблице 1 для найденных идентификаторов находится информация о людях (ФИО и пол), исключая из рассмотрения исходное лицо — Симоняна Н.И.:

- код 13 — Силис И.И. (пол «Ж»);
- код 83 — Седых А.И. (пол «М»).

Таблица 1

ID	Фамилия_И.О.	Пол
86	Седых И.Т.	М
83	**Седых А.И.**	**М**
50	Силис А.Т.	Ж
79	Симонов Т.М.	М
23	Симонов А.Т.	М
13	**Силис И.И.**	**Ж**
98	Симонян Т.Н.	Ж
11	Симонян Н.И.	М
...		

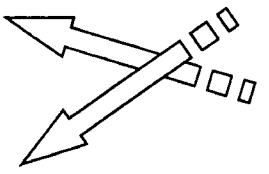


Таблица 2

ID_Родителя	ID_Ребёнка
98	83
86	**13**
79	50
86	**83**
13	50
79	23
13	23
98	13
86	11
...	...

5) В таблице 2 ищется племянник Симоняна Н.И., зная, что это сын его брата или сестры, где в поле «ID_Родителя» записано значение 13 (поскольку записи для кода 83 в таблице отсутствуют). Таких записей две, в них в поле «ID_Ребёнка» записаны значения кодов **50** и **23**.

Таблица 1

ID	Фамилия_И.О.	Пол
86	Седых И.Т.	М
83	Седых А.И.	М
50	Силис А.Т.	Ж
79	Симонов Т.М.	М
23	Симонов А.Т.	М
13	**Силис И.И.**	**Ж**
98	Симонян Т.Н.	Ж
11	Симонян Н.И.	М
...		

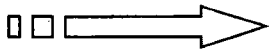


Таблица 2

ID_Родителя	ID_Ребёнка
98	83
86	13
79	50
86	83
13	**50**
79	23
13	**23**
98	13
86	11
...	...

6) Возвращаясь к таблице 1, для найденных кодов определяются ФИО и пол соответствующих лиц:

- код 50 — Силис А.Т. (пол «Ж»);
- код 23 — Симонов А.Т. (пол «М»).

Таблица 1

ID	Фамилия_И.О.	Пол
86	Седых И.Т.	М
83	Седых А.И.	М
50	**Силис А.Т.**	**Ж**
79	**Симонов Т.М.**	М
23	**Симонов А.Т.**	**М**
13	Силис И.И.	Ж
98	Симонян Т.Н.	Ж
11	Симонян Н.И.	М
...		

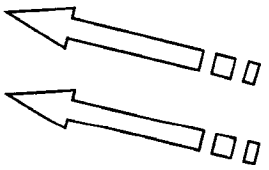


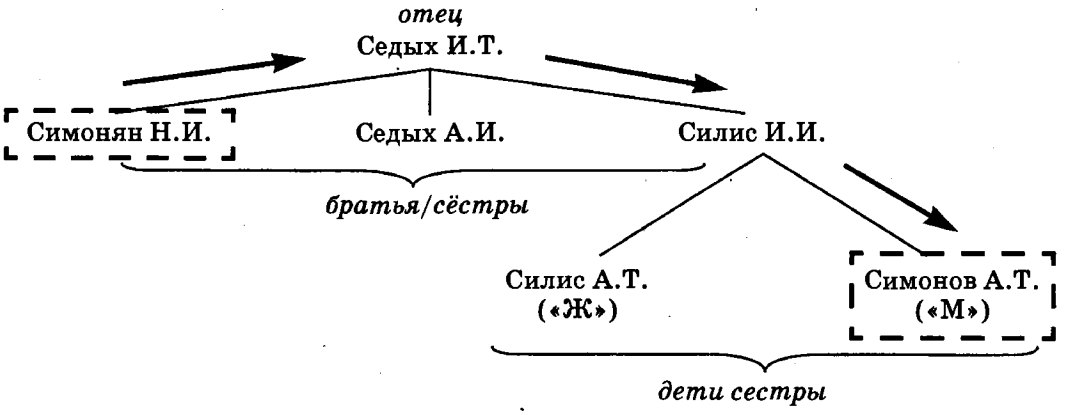
Таблица 2

ID_Родителя	ID_Ребёнка
98	83
86	13
79	50
86	83
13	**50**
79	23
13	**23**
98	13
86	11
...	...

Из указанных лиц Силис А.Т. не подходит (это — племянница). Поэтому единственный человек, подходящий в качестве решения задачи, — это Симонов А.Т.

Ответ: Симонов А.Т. (вариант ответа №3).

Примечание. «Генеалогическое древо» для данной задачи:



Задача 8*. Между четырьмя местными аэропортами: ОКТЯБРЬ, БЕРЕГ, КРАСНЫЙ и СОСНОВО ежедневно выполняются авиарейсы. Приведён фрагмент расписания перелётов между ними:

Аэропорт вылета	Аэропорт прилёта	Время вылета	Время прилёта
СОСНОВО	КРАСНЫЙ	06:20	08:35
КРАСНЫЙ	ОКТЯБРЬ	10:25	12:35
ОКТЯБРЬ	КРАСНЫЙ	11:45	13:30
БЕРЕГ	СОСНОВО	12:15	14:25
СОСНОВО	ОКТЯБРЬ	12:45	16:35
КРАСНЫЙ	СОСНОВО	13:15	15:40
ОКТЯБРЬ	СОСНОВО	13:40	17:25
ОКТЯБРЬ	БЕРЕГ	15:30	17:15
СОСНОВО	БЕРЕГ	17:35	19:30
БЕРЕГ	ОКТЯБРЬ	19:40	21:55

Путешественник оказался в аэропорту ОКТЯБРЬ в полночь (0:00). Определите самое раннее время, когда он может попасть в аэропорт СОСНОВО.

1) 15:40 2) 16:35 3) 17:15 4) 17:25

Решение

1. Исходный пункт — ОКТЯБРЬ. В таблице ищутся и выделяются все строки, в которых в графе «Аэропорт вылета» записано «ОКТЯБРЬ»:

Аэропорт вылета	Аэропорт прилёта	Время вылета	Время прилёта
СОСНОВО	КРАСНЫЙ	06:20	08:35
КРАСНЫЙ	ОКТЯБРЬ	10:25	12:35
ОКТЯБРЬ	**КРАСНЫЙ**	**11:45**	**13:30**
БЕРЕГ	СОСНОВО	12:15	14:25
СОСНОВО	ОКТЯБРЬ	12:45	16:35

Окончание табл.

Аэропорт вылета	Аэропорт прилёта	Время вылета	Время прилёта
КРАСНЫЙ	СОСНОВО	13:15	15:40
ОКТЯБРЬ	**СОСНОВО**	**13:40**	**17:25**
ОКТЯБРЬ	**БЕРЕГ**	**15:30**	**17:15**
СОСНОВО	БЕРЕГ	17:35	19:30
БЕРЕГ	ОКТЯБРЬ	19:40	21:55

2. Анализируется первая найденная строка:

ОКТЯБРЬ	КРАСНЫЙ	11:45	13:30

Здесь конечным пунктом является аэропорт «КРАСНЫЙ». В таблице ищутся строки, где пунктом отправления является «КРАСНЫЙ»:

КРАСНЫЙ	ОКТЯБРЬ	10:25	12:35
КРАСНЫЙ	СОСНОВО	13:15	15:40

Очевидно, строка «КРАСНЫЙ—ОКТЯБРЬ» не интересует, поскольку это — обратный путь. Вторая же строка соответствует требуемому пункту назначения.

Анализируются времена вылета/прилёта:

- «ОКТЯБРЬ—КРАСНЫЙ» — 11:45—**13:30**;
- «КРАСНЫЙ—СОСНОВО» — **13:15**—15:40.

Поскольку время прибытия в «КРАСНЫЙ» на 15 минут позже отлёта из «КРАСНЫЙ», данный маршрут не подходит.

В ЕГЭ условия подобных задач обычно составлены так, что в таблице всегда найдётся хотя бы один вариант, в котором время отлёта из промежуточного пункта больше, чем время прибытия в него (т.е. персонаж успевает пересесть с одного маршрута на другой), поэтому такие варианты маршрута просто отбрасываются. Однако вполне возможно, что составители заданий ЕГЭ могут усложнить их. Если вы обнаружите, что подобные «нестыковки по времени» присутствуют во всех возможных маршрутах, то необходимо будет не отбрасывать такие варианты, а анализировать их, учитывая, что путешественник, не успевший пересесть на другой маршрут сразу же, может сделать это на следующий день (т. е. в расчёты надо будет включать эту почти суточную задержку).

3. Анализируется вторая найденная строка:

ОКТЯБРЬ	СОСНОВО	13:40	17:25

В данном случае маршрут сразу содержит требуемый пункт назначения («СОСНОВО»). Время прибытия в этот конечный пункт — 17:25.

4. Анализируется третья найденная строка:

ОКТЯБРЬ	БЕРЕГ	15:30	17:15

Здесь конечным пунктом является аэропорт «БЕРЕГ». Ищутся в таблице строки, где пунктом отправления является «БЕРЕГ»:

БЕРЕГ	СОСНОВО	12:15	14:25
БЕРЕГ	ОКТЯБРЬ	19:40	21:55

Строка «БЕРЕГ—ОКТЯБРЬ» — обратный путь, поэтому она исключается из рассмотрения. Первая же строка соответствует требуемому пункту назначения.

Анализируются времена вылета/прилёта:

• «ОКТЯБРЬ—БЕРЕГ» — 15:30—**17:15**;

• «БЕРЕГ—СОСНОВО» — **12:15**—14:25.

Поскольку время прибытия в «БЕРЕГ» позже отлета из «БЕРЕГ», данный маршрут нам не подходит.

5. Таким образом, остаётся единственный возможный маршрут (в пределах текущих суток) — «ОКТЯБРЬ—СОСНОВО» с временем прибытия 17:25.

Ответ: 17:25 (вариант ответа №4).

Задача 9*. Путешественник пришёл в 08:00 на автостанцию населённого пункта КАЛИНИНО и обнаружил следующее расписание автобусов:

Пункт отправления	Пункт прибытия	Время отправления	Время прибытия
КАМЫШИ	КАЛИНИНО	08:15	09:10
КАЛИНИНО	БУКОВОЕ	09:10	10:15
РАКИТИНО	КАМЫШИ	10:00	11:10
РАКИТИНО	КАЛИНИНО	10:05	12:25
РАКИТИНО	БУКОВОЕ	10:10	11:15
КАЛИНИНО	РАКИТИНО	10:15	12:35
КАЛИНИНО	КАМЫШИ	10:20	11:15
БУКОВОЕ	КАЛИНИНО	10:35	11:40
КАМЫШИ	РАКИТИНО	11:25	12:30
БУКОВОЕ	РАКИТИНО	11:40	12:40

Определите самое ранее время, когда путешественник сможет оказаться в пункте РАКИТИНО согласно этому расписанию.

1) 12:25 2) 12:30 3) 12:35 4) 12:40

Решение

1. Исходный пункт — КАЛИНИНО. В таблице ищутся и выделяются все строки, в которых в графе «Пункт отправления» записано «КАЛИНИНО»:

Пункт отправления	Пункт прибытия	Время отправления	Время прибытия
КАМЫШИ	КАЛИНИНО	08:15	09:10
КАЛИНИНО	**БУКОВОЕ**	**09:10**	**10:15**
РАКИТИНО	КАМЫШИ	10:00	11:10
РАКИТИНО	КАЛИНИНО	10:05	12:25
РАКИТИНО	БУКОВОЕ	10:10	11:15
КАЛИНИНО	**РАКИТИНО**	**10:15**	**12:35**
КАЛИНИНО	**КАМЫШИ**	**10:20**	**11:15**
БУКОВОЕ	КАЛИНИНО	10:35	11:40
КАМЫШИ	РАКИТИНО	11:25	12:30
БУКОВОЕ	РАКИТИНО	11:40	12:40

2. Анализируется первая найденная строка:

КАЛИНИНО	БУКОВОЕ	09:10	10:15

Здесь конечный пункт — «БУКОВОЕ». Ищутся в таблице строки, где пунктом отправления является «БУКОВОЕ»:

БУКОВОЕ	КАЛИНИНО	10:35	11:40
БУКОВОЕ	РАКИТИНО	11:40	12:40

Очевидно, строка «БУКОВОЕ—КАЛИНИНО» не интересует, поскольку это — обратный путь. Вторая же строка соответствует требуемому пункту назначения.

Анализируются времена отправления/прибытия:

- «КАЛИНИНО—БУКОВОЕ» — 09:10—**10:15**;
- «БУКОВОЕ—РАКИТИНО» — **11:40**—12:40.

Время прибытия в «БУКОВОЕ» раньше, чем время отъезда из «БУКОВОЕ», следовательно, данный маршрут возможен. Время прибытия в конечный пункт «РАКИТИНО» — 12:40.

3. Анализируется вторая найденная строка:

КАЛИНИНО	РАКИТИНО	10:15	12:35

В данном случае маршрут сразу содержит требуемый пункт назначения («РАКИТИНО»). Время прибытия в этот конечный пункт — 12:35.

4. Анализируется третья найденная строка:

КАЛИНИНО	КАМЫШИ	10:20	11:15

Здесь конечный пункт — «КАМЫШИ». Ищутся в таблице строки, где пунктом отправления является «КАМЫШИ»:

КАМЫШИ	КАЛИНИНО	08:15	09:10
КАМЫШИ	РАКИТИНО	11:25	12:30

Строка «КАМЫШИ—КАЛИНИНО» — обратный путь, поэтому исключается из рассмотрения. Вторая же строка соответствует требуемому пункту назначения.

Анализируются времена отправления/прибытия:

- «КАЛИНИНО—КАМЫШИ» — 10:20—**11:15**;
- «КАМЫШИ—РАКИТИНО» — **11:25**—12:30.

Поскольку время прибытия в «КАМЫШИ» раньше, чем время отъезда из «КАМЫШИ», данный маршрут возможен. Время прибытия в конечный пункт «РАКИТИНО» — 12:30.

5. Таким образом, имеются три возможных маршрута из пункта «КАЛИНИНО» в пункт «РАКИТИНО» (один прямой и два — с пересадкой). Соответствующее им время прибытия в пункт «РАКИТИНО»:

- «КАЛИНИНО—БУКОВОЕ—РАКИТИНО» — 12:40;
- «КАЛИНИНО—РАКИТИНО» — 12:35;
- «КАЛИНИНО—КАМЫШИ—РАКИТИНО» — 12:30.

Самое раннее возможное время прибытия в пункт «РАКИТИНО» — 12:30.

Ответ: 12:30 (вариант ответа №2).

Внимание: прямой маршрут — не обязательно более быстрый (по времени прибытия в конечный пункт), чем маршруты с пересадками.

Задачи для самостоятельного решения

1*. Путешественник пришёл в 08:00 на автостанцию населённого пункта ЛИСЬЕ и обнаружил следующее расписание автобусов для всей районной сети маршрутов:

Пункт отправления	Пункт прибытия	Время отправления	Время прибытия
ЛИСЬЕ	ЗАЙЦЕВО	07:50	09:05
СОБОЛЕВО	ЛИСЬЕ	08:55	10:05
ЕЖОВО	ЛИСЬЕ	09:05	10:15
ЗАЙЦЕВО	ЕЖОВО	10:00	11:10
ЛИСЬЕ	СОБОЛЕВО	10:15	11:30
ЛИСЬЕ	ЕЖОВО	10:45	12:00
ЗАЙЦЕВО	ЛИСЬЕ	11:05	12:15
СОБОЛЕВО	ЗАЙЦЕВО	11:10	12:25
ЕЖОВО	ЗАЙЦЕВО	12:15	13:25
ЗАЙЦЕВО	СОБОЛЕВО	12:45	13:55

Определите самое раннее время, когда путешественник сможет оказаться в пункте ЗАЙЦЕВО согласно этому расписанию

1) 09:05
2) 12:15
3) 12:25
4) 13:25

2. Путешественник пришёл в 10:00 на станцию КОНЕВО и обнаружил следующее расписание пригородных электричек:

Пункт отправления	Пункт прибытия	Время отправления	Время прибытия
КОНЕВО	БЕЛЫЙ БОР	11:50	13:10
НЕЖИНО	ОЗЁРА	15:00	18:55
ЖУКИ	ОЗЁРА	13:05	15:20
КОНЕВО	ОЗЕРА	16:00	19:30
КОНЕВО	НЕЖИНО	12:15	14:50
ЖУКИ	НЕЖИНО	13:20	14:45
КОНЕВО	ЖУКИ	11:20	13:10
ОЗЁРА	КОНЕВО	15:30	18:35
ОЗЕРА	НЕЖИНО	12:20	14:05
БЕЛЫЙ БОР	КОНЕВО	15:40	17:25

Определите самое раннее время, когда путешественник сможет оказаться в пункте ОЗЁРА согласно этому расписанию

1) 19:30
2) 18:55
3) 15:20
4) 14:45

3. В фрагменте базы данных представлены сведения о родственных отношениях. Определите на основании приведённых данных фамилию и инициалы дяди Симонова А.Т.
Примечание: дядя — брат отца или матери.

Таблица 1

ID	Фамилия_И.О.	Пол
86	Седых И.Т.	М
83	Седых А.И.	М
50	Силис А.Т.	Ж
79	Симонов Т.М.	М
23	Симонов А.Т.	М
13	Силис И.И.	Ж
98	Симонян Т.Н.	Ж
11	Симонян Н.И.	М
...		

Таблица 2

ID_Родителя	ID_Ребёнка
98	83
86	13
79	50
86	83
13	50
79	23
13	23
98	13
86	11
...	...

1) Седых А.И.
2) Седых И.Т.
3) Симонов А.Т.
4) Симонов Т.М.

4. В фрагменте базы данных представлены сведения о родственных отношениях. Определите на основании приведённых данных, фамилию и инициалы племянницы Беловой Е.Т.
Примечание: племянница — дочь сестры или брата.

Таблица 1

ID	Фамилия_И.О.	Пол
85	Зорина Л.И.	Ж
82	Зорина И.Л.	Ж
42	Красных Н.А.	Ж
71	Семенов К.Л.	М
23	Семенов И.К.	М
13	Красных А.А.	Ж
95	Белова Е.Т.	Ж
10	Белов М.А.	М
...		

Таблица 2

ID_Родителя	ID_Ребёнка
95	82
85	13
71	42
85	82
13	42
71	23
13	23
95	13
85	95
...	...

1) Красных А.А.
2) Зорина Л.И.
3) Красных Н.А.
4) Зорина И.Л.

5. Дан фрагмент базы данных колледжа. Все объекты в этой базе имеют свой идентификационный код (2-значное число). Определите на основании приведённых данных № группы, в которой учится Мохначева А.

Таблица 1

Учащийся	ID
Бердыев А.	B8
Зинатуллина А.	29
Круглова С.	F7
Кузнецов Ю.	34
Лебедева А.	98
Лобчиков В.	4B
Морозова А.	01
Мохначева А.	BF
Петрова А.	E6
Смирнова Ю.	15
Тамкова В.	DD
Храповский М.	F4
Чертакова Д.	F0
Шопша Н.	E8
Шубина Ж.	96
Щербакова Е.	E3
...	...

Таблица 2

№ группы	ID
Группа 1	3D
Группа 2	67
Группа 3	36
Группа 4	5D
...	...

Таблица 3

ID_группы	ID_учащегося
36	F4
36	96
36	F8
3D	F7
3D	34
3D	BF
3D	29
3D	DD
5D	B8
5D	15
5D	E6
5D	01
67	F0
67	4B
67	98
67	E3

1) 1 2) 2 3) 3 4) 4

Ответы для самопроверки

Задача	Ответ
1	4
2	2
3	1
4	3
5	1

Технологии поиска и хранения информации

B12 Поиск информации в сети Интернет. Поисковые запросы

Конспект

Поисковый запрос для поисковой системы в Интернете представляет собой ключевое слово или несколько ключевых слов, соединенных между собой знаками логических операций **И, ИЛИ, НЕ**[1].

Процесс поиска в поисковой системе на основании поискового запроса аналогичен выборке данных в БД в соответствии с заданным условием отбора:

- если задано только одно ключевое слово, то производится поиск (выборка и включение в список найденного) всех web-страниц, в которых содержится данное ключевое слово;
- если ключевое слово задано с операцией **НЕ**, то производится поиск всех web-страниц, в которых не содержится данное ключевое слово;
- если ключевые слова связаны логической операцией **И**, то производится поиск web-страниц, в которых содержатся все эти ключевые слова;
- если ключевые слова связаны логической операцией **ИЛИ**, то производится поиск web-страниц, в которых содержится хотя бы одно ключевое слово.

Ранжирование поисковых запросов

Для разных поисковых запросов (содержащих различное число ключевых слов, связанных разными логическими операциями) количество найденных страниц будет разным. При этом важно помнить простые правила:

- **операция «И»** (&, $\wedge$) ***сокращает*** объём получаемого при поиске результата (уменьшает количество найденных сайтов), причём чем *больше* в ней задействовано *операндов*, тем *меньше* будет *объём* получаемого списка найденных сайтов;
- **операция «ИЛИ»** (|, $\vee$) ***увеличивает*** объём получаемого при поиске результата (увеличивает количество найденных сайтов), причём чем *больше* в ней задействовано *операндов*, тем *больше* будет *объём* получаемого списка найденных сайтов.

Запросы, состоящие из одного-единственного операнда (ключевого слова) и не содержащие логических операций можно рассматривать как запросы с операцией «ИЛИ» и с самым маленьким количеством операндов, т. е. в ранжированном списке такой запрос располагается непосредственно перед всеми остальными «ИЛИ»-запросами.

Таким образом, для поисковых запросов, включающих в себя только один вид логических операций (только «И» или только «ИЛИ») можно сразу определить их место в формируемом списке ранжирования — если в задаче требуется расположить запросы по возрастанию[2] количества найденных документов, то:

---

[1] Язык поисковых запросов, реализованный на почтовых сервисах (Яндекс, Google и пр.), включает также ряд других операций с ключевыми словами, позволяющих осуществлять поиск более гибко.

[2] Если требуется выстроить поисковые запросы по порядку *уменьшения* количества найденных документов, то задача решается аналогично, но расположение запросов в списке будет обратным.

- запросы с операцией «И» будут располагаться в начале списка, и чем больше в них задействовано операндов, тем такие запросы ближе к началу списка;
- запросы с операцией «ИЛИ» будут располагаться в конце списка, и чем больше в них задействовано операндов, тем такие запросы ближе к концу списка;
- запрос из одного-единственного ключевого слова будет располагаться в списке непосредственно перед запросами с операцией «ИЛИ».

Более сложен случай, когда поисковые запросы содержат оба типа логических операций — и «И», и «ИЛИ».

Иногда можно было получить правильный ответ просто методом исключения. Такой смешанный запрос располагается в ранжированном списке где-то посередине — между расположенным в его начале блоком запросов с «И» и расположенным в конце блоком запросов с «ИЛИ».

Возможны задачи, в которых смешанных запросов будет предложено несколько. В этом случае надо немного порассуждать, как ранжировать такие смешанные запросы между собой?

Пример:

Даны два смешанных запроса:

(кошки & собаки) | кролики

и

(кошки | собаки) & кролики

В первом случае будут найдены документы, в которых есть оба слова — «кошки» и «собаки», и к ним будут добавлены *все* документы со словом «кролики».

Во втором случае будут найдены документы, содержащие или слово «кошки», или слово «собаки», а из них будут отобраны только те документы, которые содержат также слово «кролики».

(Кошки & Собаки) | Кролики

(Кошки | Собаки) & Кролики

Можно считать, что «действие» (уменьшение или увеличение количества найденных документов) логических операций «И» и «ИЛИ» «ослабляется», когда они стоят в скобках, и «усиливается», когда эти операции расположены вне скобок. То есть, когда операция «И» стоит в скобках, а «ИЛИ» — вне скобок, будет найдено больше документов, чем когда операция «ИЛИ» стоит в скобках, а «И» — вне скобок.

Вычисление количеств найденных страниц

В подобных задачах считается, что существует некоторое ограниченное количество web-документов, часть которых (либо все они, либо ни один из них) может быть найдена при помощи определённого поискового запроса. В задаче рассматривается набор запросов, включающих одни и те же ключевые слова (полный или неполный набор), связанных различными логическими операциями. Для этих запросов указаны количества найденных документов, а по одному из запросов это количество требуется определить.

В этом случае найденные по каждому элементарному запросу (по какому-то одному ключевому слову) web-документы рассматриваются как пересекающиеся (операция **И**) либо объединяемые (операция **ИЛИ**) множества, а количества найденных документов вычисляются как объёмы этих множеств и/или их подмножеств.

Разбор типовых задач

Задача 1*. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдёт поисковый сервер по каждому запросу.

Для обозначения логической операции «ИЛИ» в запросе используется символ |, а для логической операции «И» — символ &.

А	чемпионы | (бег & плавание)
Б	чемпионы & плавание
В	чемпионы | бег | плавание
Г	чемпионы & Европа & бег & плавание

Решение

Запросы с операцией «И» в списке, ранжированном по возрастанию количества найденных документов, располагаются в самом начале, и чем больше в эти запросы включено операндов, тем они ближе к началу списка. Значит, список будут «открывать» запросы Г и Б.

Запросы с операцией «ИЛИ» располагаются в конце ранжированного списка, — значит, запрос В будет последним.

Смешанный (по типу операций) запрос А, по методу исключения, нужно расположить на третьем месте в списке.

Ответ: ГБАВ.

Задача 2*. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдёт поисковый сервер по каждому запросу.

Для обозначения логической операции «ИЛИ» в запросе используется символ |, а для логической операции «И» — символ &.

А	разведение & содержание & меченосцы & сомики
Б	содержание & меченосцы
В	(содержание & меченосцы) | сомики
Г	содержание & меченосцы & сомики

Решение

Запросы А, Г и Б содержат операцию «И», но разное количество операндов. «И»-запросы в ранжированном списке нужно располагать тем раньше, чем в них больше операндов. Поэтому начало списка — АГБ.

Оставшийся смешанный запрос В должен располагаться между «И»-запросами и «ИЛИ»-запросами. Но запросов с «ИЛИ» нет, поэтому запрос В просто завершает список.

Ответ: АГБВ.

Задача 3*. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдёт поисковый сервер по каждому запросу.

Для обозначения логической операции «ИЛИ» в запросе используется символ |, а для логической операции «И» — символ &.

А	волейбол | баскетбол | подача
Б	волейбол | баскетбол | подача | блок
В	волейбол | баскетбол
Г	волейбол & баскетбол & подача

Решение

Запрос Г, включающий в себя операции «И», будет первым в формируемом списке.

Что же касается остальных запросов, включающих в себя операцию «ИЛИ», то вспомним правило, что «ИЛИ»-запросы располагаются в конце списка, ранжированного по возрастанию количества найденных документов, и что чем больше в таких запросах используется операндов, тем ближе они к концу списка. Значит, первые три запроса надо выстроить в порядке ВАБ.

Ответ: ГВАБ.

Задача 4*. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдёт поисковый сервер по каждому запросу. Для обозначения логической операции «ИЛИ» в запросе используется символ |, а для логической операции «И» — символ &.

А	физкультура
Б	физкультура & подтягивания & отжимания
В	физкультура & подтягивания
Г	физкультура | фитнесс

Решение

Два запроса с операцией «И» будут располагаться в начале списка, причём первым будет идти запрос, содержащий больше всего операндов: БВ.

Запрос с операцией «ИЛИ» будет располагаться в конце списка.

Запрос из одного-единственного слова «физкультура» будет находиться непосредственно перед блоком запросов с «ИЛИ» (в данном случае этот блок состоит из одного запроса).

Ответ: БВАГ.

Задача 5*. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдёт поисковый сервер по каждому запросу. Для обозначения логической операции «ИЛИ» в запросе используется символ |, а для логической операции «И» — символ &.

1	принтеры & сканеры & продажа
2	принтеры & продажа
3	принтеры | продажа
4	принтеры | сканеры | продажа

Решение

Первыми в списке будут идти запросы с операцией «И», причём (согласно количеству используемых в них операндов) они будут идти в порядке 1, 2.

Последними в списке будут стоять запросы с «ИЛИ» и они (тоже согласно количеству операндов в них) будут следовать в порядке 3, 4.

Ответ: 1234.

Задача 6*. В языке запросов поискового сервера для обозначения логической операции «ИЛИ» используется символ «|», а для логической операции «И» — символ «&».

В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

Запрос	Найдено страниц (в тысячах)
Крейсер | Линкор	7000
Крейсер	4800
Линкор	4500

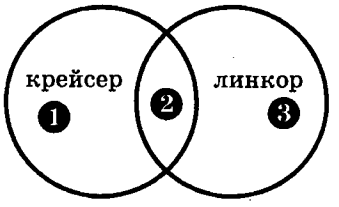
Какое количество страниц (в тысячах) будет найдено по запросу *Крейсер & Линкор* ?

Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Решение

Подобные задачи можно решать путем логических рассуждений. Однако существует универсальный способ их решения, который позволяет сделать решение всех таких задач типовым.

Строится примерная диаграмма Венна («примерная» — потому что в таких задачах *всегда* рисуются пересекающиеся области, хотя, как позже может выясниться по результатам вычислений, пересечение этих областей может и отсутствовать) и пронумеровываются по порядку все получившиеся при этом «элементарные» области:



Теперь возможно, считая порядковые номера выделенных областей своеобразными «переменными», под значениями которых понимаются объёмы соответствующих множеств, т.е. количества найденных документов, составить для каждого из указанных в таблице в условии задачи запросов систему уравнений. При этом следует помнить, что, например, область «крейсер» состоит из двух выделенных «элементарных» областей — ❶ и ❷, точно так же как область «линкор» состоит из двух «элементарных» областей ❷ и ❸.

$$\begin{cases} ❶ + ❷ + ❸ = 7000 \\ ❶ + ❷ = 4800; \\ ❷ + ❸ = 4500. \end{cases}$$

Аналогично, искомый запрос *Крейсер & Линкор* при этом трансформируется просто в значение «переменной» ❷.

Далее, например, можно подставить второе уравнение в первое и сразу получить значение «переменной» ❸:

$$\underbrace{❶ + ❷}_{❶ + ❷ = 4800} + ❸ = 7000 \quad \Rightarrow \quad 4800 + ❸ = 7000 \quad \Rightarrow \quad ❸ = 7000 - 4800 = 2200.$$

Теперь, подставив полученное значение «переменной» ❸ в третье уравнение, можно получить значение интересующей «переменной» ❷:

$$\underbrace{❷ + ❸ = 4500}_{❸ = 2200} \Rightarrow ❷ + 2200 = 4500 \Rightarrow ❷ = 4500 - 2200 = 2300.$$

Ответ: 2300.

Задача 7. В языке запросов поискового сервера для обозначения логической операции «ИЛИ» используется символ «|», а для логической операции «И» — символ «&».

В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

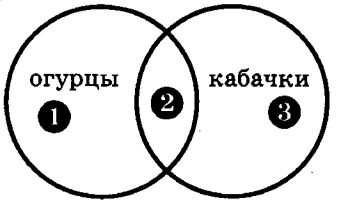
Какое количество страниц (в тысячах) будет найдено по запросу *Огурцы* ?

Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Запрос	Найдено страниц (в тысячах)
Огурцы | Кабачки	12000
Огурцы & Кабачки	6500
Кабачки	7700

Решение

Решение начинается с рисования примерной диаграммы Венна (внешне она будет точно такой же, как и в предыдущей задаче):



Составляется система уравнений, соответствующих указанным в таблице запросам:

$$\begin{cases} ❶ + ❷ + ❸ = 12000; \\ ❷ = 6500; \\ ❷ + ❸ = 7700. \end{cases}$$

Выражение же, соответствующее запросу *Огурцы*, выглядит так: ❶ + ❷.

Решается система уравнений. Для начала подставляется значение «переменной» ❷, известное по второму уравнению, в оба других уравнения системы, понижая тем самым её размерность до двух:

$$\begin{cases} ❶ + 6500 + ❸ = 12000; \\ 6500 + ❸ = 7700. \end{cases}$$

Из второго уравнения получившейся системы нетрудно получить значение «переменной» ❸: оно равно 7700 − 6500 = 1200.

Подставив его в первое уравнение, получается значение последней «переменной» — ❶:

❶ + 6500 + 1200 = 12000 ⇒ ❶ = 4300.

Тогда нетрудно вычислить, что запросу *Огурцы* соответствует:

❶ + ❷ = 4300 + 6500 = 10800 найденных документов.

Ответ: 10800

Задача 8. Некоторый сегмент сети Интернет состоит из 1000 сайтов. Поисковый сервер в автоматическом режиме составил таблицу ключевых слов для сайтов этого сегмента. Вот ее фрагмент:

Ключевое слово	Количество сайтов, для которых данное слово является ключевым
Скутер	200
Байк	250
Мопед	450

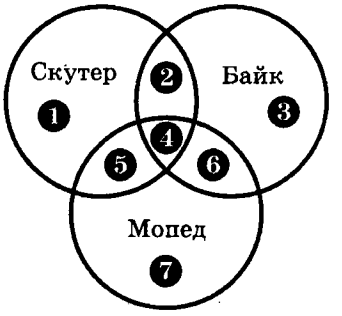
Сколько сайтов будет найдено по запросу

(Байк | Скутер) & Мопед,

если по запросу *Байк | Скутер* было найдено 450 сайтов, по запросу *Байк & Мопед* — 40, а по запросу *Скутер & Мопед* — 50 сайтов?

Решение

В этом случае вид диаграммы Венна будет другим — она будет состоять из трёх кругов:



Далее, как и раньше, составляется система уравнений для каждого заданного запроса (как в таблице, так и вне её), не забывая добавить условие, что всего сайтов было 1000:

$$\begin{cases} \textit{Скутер}: & ❶ + ❷ + ❹ + ❺ = 200; \\ \textit{Байк}: & ❷ + ❸ + ❺ + ❻ = 250; \\ \textit{Мопед}: & ❹ + ❺ + ❻ + ❼ = 450; \\ \textit{Байк | скутер}: & ❶ + ❷ + ❸ + ❹ + ❺ + ❻ = 450; \\ \textit{Байк \& мопед}: & ❺ + ❻ = 40; \\ \textit{Скутер \& мопед}: & ❹ + ❺ = 50; \\ \textit{Всего сайтов}: & ❶ + ❷ + ❸ + ❹ + ❺ + ❻ + ❼ = 1000. \end{cases}$$

Запрос *(Байк | Скутер) & Мопед* при этом соответствует выражению:

❹ + ❺ + ❻.

Данная система уравнений сложнее предыдущих, но при внимательном рассмотрении можно найти способ подстановки одних уравнений в другие, быстро приводящий к решению.

Из четвёртого уравнения вычитаются первые два:

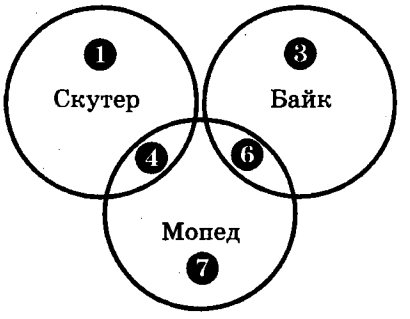
❶ + ❷ + ❸ + ❹ + ❺ + ❻ − (❶ + ❷ + ❹ + ❺) − (❷ + ❸ + ❺ + ❻) =

= 450 − 200 − 250 = 0 ⇒ − ❷ − ❺ = 0 ⇒ ❷ + ❺ = 0.

Помня, что значения наших «переменных», обозначенных цифрами в кружочках, — это *количества найденных документов* (т. е. числа натурального ряда), *которые не могут быть отрицательными*. Следовательно, если сумма таких «переменных» равна нулю, то *все* переменные, входящие в эту сумму (в данном случае — ❷ и ❺), также «обязаны» равняться нулю. Получается:

❷ = 0; ❺ = 0.

Кстати, это является объяснением тому, что диаграмма Венна называется «примерной»: равенство нулю этих двух «переменных» означает, что запросу *Байк & Скутер* не соответствует ни один документ (сайт), т. е. диаграмма Венна на самом деле должна выглядеть так:



Теперь складываются пятое и шестое уравнения. Одновременно в них подставляется уже известное нулевое значение «переменной» ❺:

❺ + ❻ + ❹ + ❺ = 40 + 50 ⇒ ❹ + ❻ = 90.

Нетрудно догадаться, что (опять-таки с учётом нулевого значения «переменной» ❺) это и есть значение искомого выражения для запроса *(Байк | Скутер) & Мопед*, — следовательно, задача решена (при необходимости, «расплетая» систему уравнений дальше, можно было бы вычислить значения всех используемых «переменных» — от ❶ до ❼ — и определить количество найденных документов для *любого* запроса с указанными ключевыми словами).

Ответ: 90.

Задачи для самостоятельного решения

1*. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдёт поисковый сервер по каждому запросу.

Для обозначения логической операции «ИЛИ» в запросе используется символ |, а для логической операции «И» — символ &.

1	Канарейки | Щеглы | Содержание
2	Канарейки & Содержание
3	Канарейки & Щеглы & Содержание
4	Разведение & Содержание & Канарейки & Щеглы

2. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдёт поисковый сервер по каждому запросу.

Для обозначения логической операции «ИЛИ» в запросе используется символ |, а для логической операции «И» — символ &.

1	Принтер & Матричный
2	Принтер & Матричный & Струйный
3	Принтер
4	Принтер | Матричный | Струйный

3. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдёт поисковый сервер по каждому запросу.
Для обозначения логической операции «ИЛИ» в запросе используется символ |, а для логической операции «И» — символ &.

1	(Хомяк | Мышь) & Суслик
2	Хомяк | (Мышь & Суслик)
3	Хомяк & Мышь & Суслик
4	Хомяк | Мышь | Суслик

4. В языке запросов поискового сервера для обозначения логической операции «ИЛИ» используется символ «|», а для логической операции «И» — символ «&».
В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.
Какое количество страниц (в тысячах) будет найдено по запросу *Нейтрон*?
Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Запрос	Найдено страниц (в тысячах)
Протон & Нейтрон	5100
Протон	9700
Протон | Нейтрон	14200

5*. В языке запросов поискового сервера для обозначения логической операции «ИЛИ» используется символ «|», а для логической операции «И» — символ «&».
В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

Запрос	Найдено страниц в тысячах)
Шахматы | Теннис	7770
Теннис	5500
Шахматы & Теннис	1000

Какое количество страниц (в тысячах) будет найдено по запросу *Шахматы*?
Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Ответы для самопроверки

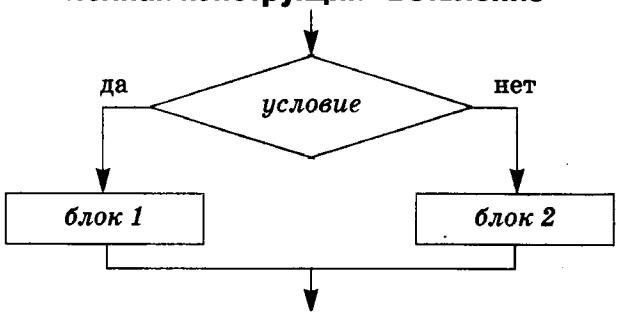
Задача	Ответ
1	4321
2	2134
3	3124
4	9600
5	3270

Программирование

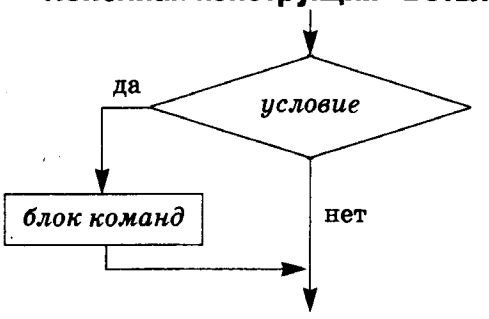
B6 Условный оператор

Конспект

Полная конструкция «Ветвление»



Неполная конструкция «Ветвление»



Конструкция ветвления в языке Паскаль

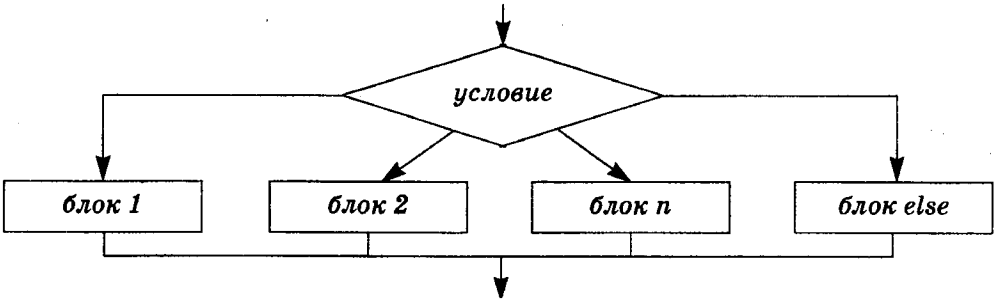
Полная	Неполная

if (<условие>) then
begin
<блок 1>
end
else
begin
<блок 2>
end;
	if (<условие>) then
	begin
	<блок>
	end;

Если блоки команд состоят только из одного оператора каждый, то использовать операторные скобки `begin ... end` не обязательно:

Полная	Неполная
if (<условие>) then <команда 1>; else <команда 2>;	if (<условие>) then <команда>;

Множественное ветвление (конструкция выбора)



Конструкция выбора в языке Паскаль

```
case <ключ> of
  <селектор 1>: begin <блок 1> end;
  <селектор 2>: begin <блок 2> end;
  . . . . .
  <селектор n>: begin <блок n> end;
  else begin <блок else>
end;
```

Ключ — выражение или переменная, имеющие целое или символьное значение.

В качестве **селектора** допускаются: константы, списки констант (через запятую), интервалы (через знак ..) и символы.

Разбор типовых задач

Задача 1*. Определите значение целочисленных переменных *a* и *b* после выполнения фрагмента программы (*ниже представлена одна и та же программа, записанная на разных языках программирования*):

Бейсик	Паскаль	Алгоритмический язык

a = 42
b = 14
a = a \ b
b = a * b
a = b \ a
\ – стандартная операция, вычисляющая результат деления нацело первого аргумента на второй
	a := 42;
	b := 14;
	a := a div b;
	b := a * b;
	a := b div a;
	{div – стандартная операция, вычисляющая результат деления нацело первого аргумента на второй}
		a := 42
		b := 14
		a := div(a, b)
		b := a * b
		a := div(b, a)
		|div – стандартная функция, вычисляющая результат деления нацело первого аргумента на второй|

1) $a = 42, b = 14$ 2) $a = 1, b = 42$ 3) $a = 0, b = 588$ 4) $a = 14, b = 42$

Решение

Подобные задачи решаются путём выполнения ручной трассировки (покомандного выполнения программы) в табличной форме. При этом в таблицу включаются все переменные, значения которых изменяются в программе. Дополнительно в левом столбце могут быть записаны соответствующие операторы программы.

В данном случае используются две переменные. Факт изменения значения той или иной переменной отмечен жирным подчёркнутым шрифтом и фоновым закрашиванием ячеек таблицы.

Оператор	*a*	*b*
a := 42;	**42**	–
b := 14;	42	**14**
a := a div b;	**3**	14
b := a * b;	3	**42**
a := b div a;	**14**	42

Таким образом, по завершении выполнения программы значения переменных равны: $a = 14, b = 42$.

Ответ: $a = 14, b = 42$ (вариант ответа №4).

При выполнении трассировки программ следует быть внимательными:
- до присваивания переменной первого значения содержимое этой переменной считается неопределённым и отмечается в таблице прочерком;
- при выполнении операторов присваивания, в которых одна и та же переменная стоит слева и справа нужно не ошибиться в определении, откуда берётся значение переменной для вычисления выражения после знака присваивания (из **предыдущей** строки таблицы) и в какой столбец записывается результат (в столбец, соответствующий переменной, которая записана **до знака присваивания**).

Задача 2*. Определите значение целочисленных переменных a и b после выполнения фрагмента программы:

Бейсик	Паскаль	Алгоритмический язык

a = 2468
b = (a MOD 1000) * 10
a = a \ 1000 + b
'\ и MOD — операции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно
	a := 2468;
	b := (a mod 1000) * 10;
	a := a div 1000 + b;
	{div и mod — операции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно}
		a := 2468
		b := mod(a, 1000) * 10
		a := div(a, 1000) + b
		|div и mod — функции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно|

1) $a = 22$, $b = 20$
2) $a = 4682$, $b = 4680$
3) $a = 8246$, $b = 246$
4) $a = 470$, $b = 468$

Решение

Оператор	a	b
a := 2468;	**2468**	—
b := (a mod 1000) * 10;	2468	**4680**
a := a div 1000 + b;	**4682**	4680

Таким образом, по завершении выполнения программы значения переменных равны: $a = 4682$, $b = 4680$.

Ответ: $a = 4682$, $b = 4680$ (вариант ответа №2).

Задача 3*. Определите значение переменной c после выполнения следующего фрагмента программы.

Бейсик	Паскаль	Алгоритмический язык

a = 5
a = a + 6
b = -a
c = a - 2 * b
	a := 5;
	a := a + 6;
	b := -a;
	c := a - 2 * b;
		a := 5
		a := a + 6
		b := -a
		c := a - 2 * b

1) $c = -11$
2) $c = 15$
3) $c = 27$
4) $c = 33$

Решение

Выполняется аналогично предыдущим задачам, но в последней команде программы по полученным значениям a и b дополнительно вычисляется значение переменной c.

Оператор	a	b
a := 5;	**5**	—
a := a + 6;	**11**	—
b := -a;	11	**–11**

Тогда значение переменной c равно: $a - 2 \cdot b = 11 - 2 \cdot (-11) = 11 + 22 = 33$.
Ответ: $c = 33$ (вариант ответа №4).

Задача 4*. Определите значение переменной c после выполнения следующего фрагмента программы (записанного ниже на разных языках программирования):

Бейсик

```
a = 100
b = 30
a = a - b * 3
IF a > b THEN
   c = a - b
ELSE c = b - a
ENDIF
```

Паскаль

```
a := 100;
b := 30;
a := a - b *3;
if a > b then
    c := a - b
else c := b - a;
```

Си

```
a = 100;
b = 30;
a = a - b * 3;
if (a > b)
    c = a - b;
else c = b - a;
```

Алгоритмический язык

```
a := 100
b := 30
a := a - b * 3
если a > b то
   c := a - b
иначе c := b - a
все
```

1) $c = 20$ 2) $c = 70$ 3) $c = -20$ 4) $c = 180$

Решение

Трассировка выполняется аналогично, но в таблицу дополнительно включается переменная c. Кроме того, в строках таблицы трассировки, соответствующих условному оператору, желательно отдельно отмечать истинность или ложность условия ветвления.

Оператор	a	b	c
a := 100;	**100**	—	—
b := 30;	100	**30**	—
a := a - b *3;	**10**	30	—
if a > b then Условие не выполнено — выполняется ветвь else	10 (<)	30	—
else c := b - a;	10	30	**20**

Ответ: $c = 20$ (вариант ответа №1).

Задача 5*. Определите значение переменной *c* после выполнения следующего фрагмента программы (записанного ниже на разных языках программирования):

Бейсик

```
a = 40
b = 10
b = -a / 2 * b
If a < b Then c = b - a
Else c = a - 2 * b
End If
```

Паскаль

```
a := 40;
b := 10;
b := -a / 2 * b;
if a < b then c : = b - a
else c : = a - 2 * b;
```

Си

```
a = 40;
b = 10;
b = -a / 2 * b;
if (a < b)
   c = b - a;
else
c = a - 2 * b;
```

Алгоритмический язык

```
a := 40
b := 10
b : = -a / 2 * b
если a < b
   то c : = b - a
   иначе c := a - 2 * b
все
```

Решение

Выполняется аналогично предыдущей задаче. Единственное отличие — здесь не приведены варианты ответов, поэтому при заполнении бланка ЕГЭ нужно указывать сам ответ.

Оператор	*a*	*b*	*c*
a := 40;	**40**	–	–
b := 10;	40	**10**	–
b := - a / 2 * b;	40	**–200**	–
if a < b then Условие не выполнено - выполняется ветвь else	40	–200	–
else c : = a - 2 * b;	40	–200	**440**

Ответ: $c = 440$.

Задача 6*. Следующая программа, содержащая, по крайней мере, одну ошибку, после устранения ошибок должна определять день недели для произвольного дня месяца. В ней считается, что первое число данного месяца — понедельник. Укажите в листе ответа все ошибки. Взяв эту программу за основу, напишите программу, которая будет решать ту же задачу при условии, что w_1 — день недели для первого числа месяца. Значение w_1 (целое число от 1 до 7) должно запрашиваться программой. Интересующее нас число месяца d (от 1 до 31) также должно запрашиваться. Предполагается, что ввод данных будет корректным.

Программа на языке Паскаль

```
Var d, w: integer;
begin
   readln(d);
   w := d div 7;
   case w of
   1:writeln('понедельник');
   2:writeln('вторник');
   3:writeln('среда');
   4:writeln('четверг');
   5:writeln('пятница');
   6:writeln('суббота');
   7:writeln('воскресенье');
   end
end.
```

Программа на языке Бейсик

```
DIM w, d AS INTEGER
INPUT d
w = d \ 7
IF w = 1 THEN PRINT "понедельник"
IF w = 2 THEN PRINT "вторник"
IF w = 3 THEN PRINT "среда"
IF w = 4 THEN PRINT "четверг"
IF w = 5 THEN PRINT "пятница"
IF w = 6 THEN PRINT "суббота"
IF w = 7 THEN PRINT "воскресенье"
END
```

Решение

Данная задача отличается от предыдущих. Во-первых, в ней рассматривается оператор множественного ветвления. Во-вторых, в ней требуется выполнить анализ листинга на наличие ошибок и затем написать свою программу (т.е. проверяются умения читать текст программы, анализировать его и писать собственные программы).

Составим таблицу анализа листинга:

Оператор	Комментарий
`Var d, w: integer;`	Объявление целочисленных переменных.
`begin`	Начало программы.
`readln(d);`	Считали значение *d* — число месяца (от 1 до 31).
`w := d div 7;`	Вычислили значение *w* как результат целочисленного деления *d* на 7.

`case w of`
`1:writeln('понедельник');`
`2:writeln('вторник');`
`3:writeln('среда');`
`4:writeln('четверг');`
`5:writeln('пятница');`
`6:writeln('суббота');`
`7:writeln('воскресенье');`
`end`
	Предполагая, что *w* содержит номер дня недели (от 1 до 7), выводится по нему название дня недели.

`end.`	Конец программы

Первая ошибка содержится в операторе `w := d div 7;` и заключается в том, что для вычисления по номеру дня в месяце номера дня недели требуется определять **остаток от деления *d* на 7**. Вторая ошибка — в том, что воскресенью тогда будет соответствовать остаток от деления, равный не 7, а нулю.

Проверить это можно, построив таблицу, например, для 15 первых дней месяца (по условию задачи $d = 1$ соответствует понедельнику, т. е. $w = 1$).

d	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
d div 7	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2
d mod 7	1	2	3	4	5	6	0	1	2	3	4	5	6	0	1

По этой таблице хорошо видно, что оператор `(d div 7) + 1` позволяет определить по номеру дня в месяце номер недели в месяце.

Тогда исправленная программа будет иметь вид (рассматривается вариант на языке Паскаль; исправленные строки выделены жирным шрифтом; исправления дополнительно выделены подчеркиванием):

```
Var d, w: integer;
begin
  readln(d);
  w := d mod 7;
  case w of
  1:writeln('понедельник');
  2:writeln('вторник');
  3:writeln('среда');
```

```
   4:writeln('четверг');
   5:writeln('пятница');
   6:writeln('суббота');
   0:writeln('воскресенье');
   end
end.
```

Вторая часть задания: написание программы, решающей ту же задачу при условии, что w_1 — день недели для первого числа месяца, где значение w_1 (целое число от 1 до 7), запрашивается программой.

Для этого в операторе, вычисляющем номер дня недели (`w := d mod 7`) нужно внести следующие изменения:

- прибавить к d значение w_1;
- вычесть из полученной суммы единицу, чтобы для $d = 1$ результат был равен w_1.

Получаемый вид оператора: `w := (d + w1 - 1) mod 7`.

Чтобы проверить его работу, составляется таблица (матрица) для различных значений w_1 и значений d от 1 до 8.

w_1 \ d	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	0	1
2	2	3	4	5	6	0	1	2
3	3	4	5	6	0	1	2	3
4	4	5	6	0	1	2	3	4
5	5	6	0	1	2	3	4	5
6	6	0	1	2	3	4	5	6
7	0	1	2	3	4	5	6	0

Как нетрудно видеть, номер дня недели вычисляется верно.

Тогда текст доработанной программы будет таким:

```
Var d, w: integer;
begin
   readln(d, w1);
   w := (d + w1 - 1) mod 7;
   case w of
   1:writeln('понедельник');
   2:writeln('вторник');
   3:writeln('среда');
   4:writeln('четверг');
   5:writeln('пятница');
   6:writeln('суббота');
   0:writeln('воскресенье');
   end
end.
```

Задача 7*. Требовалось написать программу, в которой нужно было проверить, лежит ли число x на числовой оси между числами a и b («между» понимается в строгом смысле, т.е. случай $x = a$ или $x = b$ недопустим). Числа x, a, b являются натуральными, и известно, что a отлично от b (но неизвестно: $a > b$ или $b > a$). Входная информация вводится с клавиатуры, а на выходе должно быть сообщение вида «x между a и b» (если это действительно так), в противном случае никакой выходной информации не выдаётся.

Программист торопился и написал программу некорректно.

Программа на Паскале	Программа на Бейсике

VAR a, b, x: integer;
BEGIN
readln(a, b, x);
if (a > x) AND (x > b) then
writeln('x между a, b');
END.
	CLS
	INPUT a, b, x
	IF (a > x) AND (x > b) THEN
	PRINT "x между a, b"
	END

Последовательно выполните три задания:

1) Приведите пример таких чисел a, b, x, при которых программа работает неправильно.

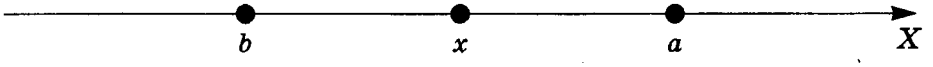
2) Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой способ доработки исходной программы).

3) Укажите, как можно доработать программу, соблюдая дополнительное условие: доработанная программа не должна использовать логических операций AND или OR.

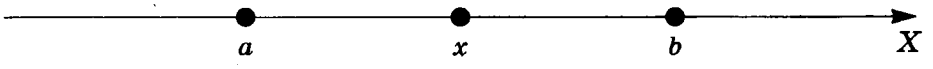
Решение

Очевидно, что поскольку все операторы, кроме `if`, ошибок явно не содержат, ошибка может содержаться только в операторе `if (a > x) AND (x > b) then`.

Обратим внимание на записанное в нем условие: `(a > x) И (x > b)`. Оно будет истинным в случае, если значение x не просто расположено на числовой оси между a и b, но притом $a > b$:



Но в случае, если числа a и b введены наоборот, т. е. $a < b$, для значения x, лежащего между ними, получается:



$(a > x)$ — ложно;

$(x > b)$ — ложно;

$(a > x)$ И $(x > b)$ — ложно.

В результате сообщение «`x между a, b`» выведено не будет (хотя и должно).

Пример таких значений: $a = 1$, $x = 2$, $b = 3$.

Как исправить эту ошибку?

Поскольку она возникает, если $a < b$, можно обеспечить, чтобы вводимые числа никогда не оказывались в таком соотношении, а всегда было $a > b$. Для этого можно сравнить введенные числа и в случае, если $a < b$, просто поменять их местами (добавив дополнительную переменную).

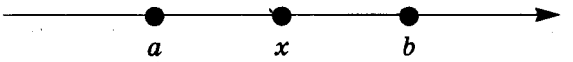
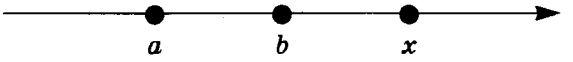
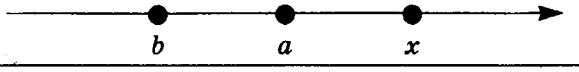
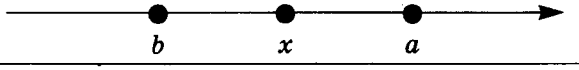
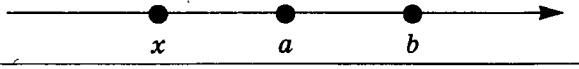
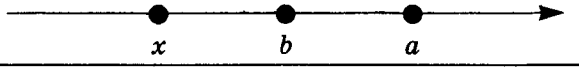
Пример исправленной программы:

```
VAR a, b, x: integer;
    p: integer;
BEGIN
 readln(a, b, x);
if a < b then begin p := a; a := b; b := p end;
 if (a > x) AND (x > b) then
    writeln('x между a, b');
END.
```

Как изменить это условие, чтобы избавиться от использования логических операций AND или OR?

Взаимное расположение двух точек на координатной оси можно оценить по знаку разности их координат. Например для точек C и D разность их координат $(c - d)$ будет положительной при $c > d$ (точка C расположена правее D) и отрицательной при $c < d$ (точка C левее D).

Определяются для всех возможных случаев взаимного расположения точек a, b и x знаки разностей $(a - x)$ и $(b - x)$.

Случай	Знак $(a - x)$	Знак $(b - x)$	x между a и b ?
	—	+	Да
	—	—	Нет
	—	—	Нет
	+	—	Да
	+	+	Нет
	+	+	Нет

Все возможные варианты расположения точек a, b и x можно получить как все варианты перестановок букв a, b, x (количество таких вариантов равно $3! = 1 \cdot 2 \cdot 3 = 6$):

abx
axb
bax
bxa
xab
xba

Проверка: каждая буква должна «побывать» на каждой позиции в слове дважды.

Строки таблицы, соответствующие искомой ситуации (x между a и b), имеют отличительную особенность — разные знаки разностей $(a - x)$ и $(b - x)$, тогда как в остальных строках знаки этих разностей одинаковы (оба «+» или оба «−»). Тогда в качестве искомого условия можно определять знак произведения этих разностей: если оно отрицательно, то x лежит между a и b, а если оно положительно, то вне интервала ab.

Тогда текст программы может быть таким:

```
VAR a, b, x: integer;
BEGIN
 readln(a, b, x);
 if (a - x) * (b - x) < 0 then
    writeln('x между a, b');
END.
```

Задача 8*. Требовалось написать программу, которая решает уравнение «$ax + b = 0$» относительно x для любых чисел a и b, введённых с клавиатуры. Все числа считаются действительными. Программист торопился и написал программу неправильно.

Программа на паскале	Программа на бейсике	Программа на си

var a, b, x: real;
begin
readln(a, b, x);
if b = 0 then
write('x = 0')
else
if a = 0 then
write('нет решений')
else
write('x =', -b / a);
end.
	INPUT a, b, x
	IF b = 0 THEN
	PRINT "x = 0"
	ELSE
	IF a = 0 THEN
	PRINT "нет решений"
	ELSE
	PRINT "x =", -b / a
	ENDIF
	ENDIF
	END
		void main(void)
		{ float a, b, x;
		scanf("%f%f%f", &a,&b,&x);
		if (b == 0)
		printf("x = 0");
		else
		if (a == 0)
		printf("нет решений");
		else
		printf("x=%f", -b / a);
		}

Последовательно выполните три задания:

1) Приведите пример таких чисел a, b, x, при которых программа неверно решает поставленную задачу.

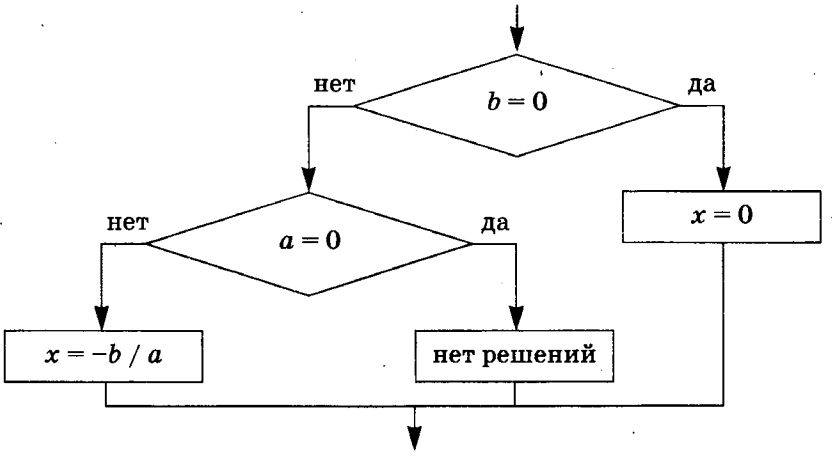
2) Укажите, какая часть программы является лишней.

3) Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой способ доработки исходной программы).

Решение

Лишней частью программы является ввод в операторе readln значения x, которое должно вычисляться в программе.

Для поиска ошибки вычерчивается блок-схема основной части задачи, реализованной операторами if:



В виде таблицы записываются все возможные результаты («отклики») такого алгоритма при различных вариантах значений a и b:

a	b	Результат
$<>0$	$<>0$	$x = -b / a$
$<>0$	$=0$	$x = 0$
$=0$	$<>0$	нет решений
$=0$	$=0$	$x = 0$

Очевидная ошибка — в четвёртой строке таблицы (результат, если $a = 0$ и $b = 0$): в этом случае уравнение имеет вид «$0x + 0 = 0$» и имеет бесконечное множество решений. Ошибка происходит из-за того, что первый же оператор if в случае, если $b = 0$, не анализирует уже значение a и сразу выдаёт результат «$x = 0$».

Доработка программы заключается в таком изменении структуры проверки условий, чтобы сначала анализировалось значение a.

Тогда возможный вид доработанной программы, не содержащей лишних фрагментов, может быть таким:

```
var a, b, x: real;
begin
readln(a,b);
if a = 0 then
  begin
    if b = 0 then
       write('любое число')
      else
       write('нет решений')
    end
  else
    write('x =', -b / a);
end.
```

Случай, когда $a <> 0$, $b = 0$, можно отдельно не рассматривать, так как в этом случае правильное значение x вычисляется по формуле $x = -b / a$.

Задачи для самостоятельного решения

1*. Определите значение целочисленных переменных a и b после выполнения фрагмента программы:

Бейсик	Паскаль	Алгоритмический язык

a = 1819
b = (a \ 100) * 10 + 9
a = (10*b - a) MOD 100
'\ и MOD — операции,
вычисляющие результат
деления нацело первого
аргумента на второй
и остаток от деления
соответственно
	a := 1819;
	b := (a div 100) * 10 + 9;
	a := (10 * b - a) mod 100;
	{div и mod — операции,
	вычисляющие результат
	деления нацело первого
	аргумента на второй
	и остаток от деления
	соответственно}
		a := 1819
		b := div(a,100) * 10 + 9
		a := mod(10 * b - a, 100)
		|div и mod — функции,
		вычисляющие результат
		деления нацело первого
		аргумента на второй
		и остаток от деления
		соответственно|

1) $a = 81$, $b = 199$ 2) $a = 81$, $b = 189$ 3) $a = 71$, $b = 199$ 4) $a = 71$, $b = 189$

2*. Определите значение целочисленных переменных a и b после выполнения фрагмента программы:

Бейсик	Паскаль	Алгоритмический язык

a = 3 + 8 * 4
b = (a \ 10) - 14
a = (b MOD 10) + 2
'\ и MOD — операции,
вычисляющие результат
деления нацело первого
аргумента на второй
и остаток от деления
соответственно
	a := 3 + 8 * 4;
	b := (a div 10) + 14;
	a := (b mod 10) + 2;
	{div и mod — операции,
	вычисляющие результат
	деления нацело первого
	аргумента на второй
	и остаток от деления
	соответственно}
		a := 3 + 8 * 4
		b := div(a, 10) + 14
		a := mod(b, 10) + 2
		|div и mod — функции,
		вычисляющие результат
		деления нацело первого
		аргумента на второй
		и остаток от деления
		соответственно|

1) $a = 0$, $b = 18$ 2) $a = 9$, $b = 17$ 3) $a = 11$, $b = 19$ 4) $a = 10$, $b = 18$

3*. Определите значение переменной c после выполнения следующего фрагмента программы, в котором a, b и c — переменные вещественного (действительного) типа.

Бейсик

```
a = 120
b = 100
a = a + b / 2
IF b < a / 2 THEN
c = b + a
ELSE
c = b + a / 2
END IF
```

Паскаль

```
a := 120;
b := 100;
a := a + b / 2;
if b < a / 2
then
c := b + a
else
c := b + a / 2;
```

Си

```
a = 120;
b = 100;
a = a + b / 2;
if (b < a / 2)
c = b + a;
else
c = b + a / 2;
```

Алгоритмический язык

```
a := 120
b := 100
a := a + b / 2
если b < a / 2
то c := b + a
иначе c := b + a / 2
все
```

1) $c = 105$ 2) $c = 160$ 3) $c = 185$ 4) $c = 210$

4. Определите значение переменной c после выполнения следующего фрагмента программы, в котором a, b и c — переменные вещественного (действительного) типа.

Бейсик

```
a = 15
b = 33
a = b MOD a
IF a <> (b \ 5) THEN
c = a * b
ELSE
c = (a + b) * 3
END IF
```

Паскаль

```
a := 15;
b := 33;
a := b mod a;
if a <> (b div 5)
then
c := a * b
else
c := (a + b) * 3;
```

1) $c = 27$ 2) $c = 99$ 3) $c = 18$ 4) $c = 108$

5*. Требовалось написать программу, которая решает уравнение «$a|x| = b$» относительно x для любых чисел a и b, введённых с клавиатуры. Все числа считаются действительными. Программист торопился и написал программу неправильно.

Программа на Паскале

```
var a, b, x: real;
begin
readln(a, b, x);
if a = 0 then
if b = 0 then
write ('любое число')
else
write ('нет решений')
else
if b = 0 then
write ('x = 0')
else
write ('x =', b / a, ' или
          x =', -b / a);
end.
```

Программа на Бейсике

```
INPUT a, b, x
IF a = 0 THEN
IF b = 0 THEN
PRINT "любое число"
ELSE
PRINT "нет решений"
ENDIF
ELSE
IF b = 0 THEN
PRINT "x = 0"
ELSE
PRINT "x =",b / a, "или
          x =", -b / a
ENDIF
ENDIF
END
```

Программа на Си

```
void main(void)
{float a, b, x;
scanf("%f%f%f", &a,&b,&x);
if(a == 0)
if(b == 0)
printf("любое число");
else
printf ("нет решений");
else
if(b == 0)
printf("x = 0");
else
printf("x=%f или x=%f",
          b / a,-b / a);
}
```

Последовательно выполните три задания:
1) Приведите пример таких чисел a, b, x, при которых программа неверно решает поставленную задачу.
2) Укажите, какая часть программы является лишней.
3) Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой способ доработки исходной программы).

Ответы для самопроверки

Задача	Ответ
1	4
2	2
3	3
4	2

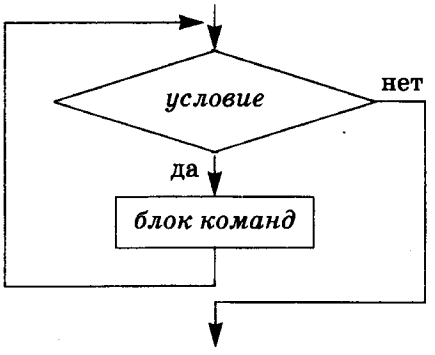
5	1. $a = 1$, $b = -1$, $x = 0$. Значение x может быть не указано. Значения a и b могут быть любыми ненулевыми числами с разными знаками. Также допустим ответ, что программа работает неправильно при любых ненулевых a и b, имеющих разные знаки.
	2. Лишняя часть: не нужно вводить x с клавиатуры. Верно: `readln(a,b);`
	3. Возможная доработка:
	`readln(a, b);`
	`if a = 0 then`
	`if b = 0 then write('любое число')`
	`else write('нет решений')`
	`else`
	`if b / a > 0 then`
	`write('x=', b / a, 'или x=', -b / a)`
	`else`
	`if b = 0 then write ('x=0')`
	`else write('нет решений');`

Программирование

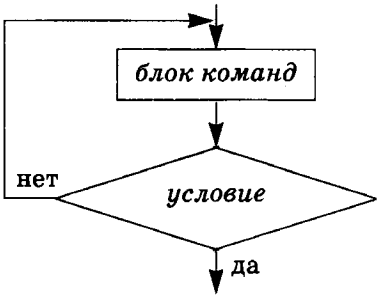
B3 Циклы: трассировка алгоритма

Конспект

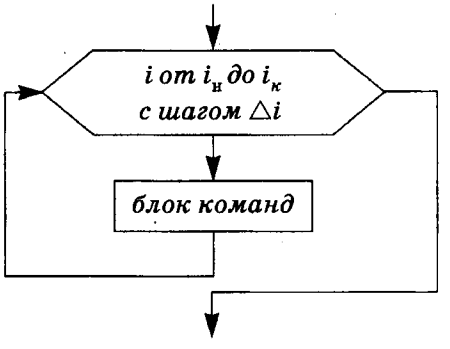
Цикл с предусловием (цикл ПОКА)



Цикл с постусловием (цикл ДО)



Цикл с параметром (цикл со счётчиком, цикл ДЛЯ)



Внимание!
В цикле с параметром количество выполнений тела цикла задано жёстко и однозначно. В циклах же ПОКА и ДО количество выполнений тела цикла заранее не известно и определяется ситуацией. Поэтому необходимо проследить, чтобы во время выполнения циклов ПОКА и ДО происходило какое-то изменение переменных, задействованных в условии цикла (чтобы в какой-то момент истинность условия изменилась и цикл завершился). Несоблюдение этого требования приводит к бесконечному выполнению цикла («зацикливанию» программы).

Конструкции циклов в языке Паскаль

Цикл с предусловием:

```
while <условие>
  begin
    <блок команд>
  end;
```

или

```
do while <условие> do <команда>;
```

Цикл с постусловием:

```
repeat
  <команды>
until <условие>;
```

Цикл со счётчиком:

Цикл с шагом 1	Цикл с шагом −1

`for <i> := <i`$_н$`> to <i`$_к$`> do`
`  begin`
`  <блок команд>`
`  end;`
	`for <i> := <i`$_н$`> downto <i`$_к$`> do`
	`  begin`
	`  < блок команд >`
	`  end;`

или	или
`for <i> := <i`$_н$`> to <i`$_к$`> do <команда>;`	`for <i> := <i`$_н$`> downto <i`$_к$`> do` `  <команда>;`

Тело цикла может включать в себя другой цикл. «Глубина вложенности» циклов теоретически может быть любой.

Правила построения вложенных циклов:

1) циклы не должны «пересекаться», т.е. внутренний цикл должен начинаться и полностью завершаться внутри тела внешнего цикла;

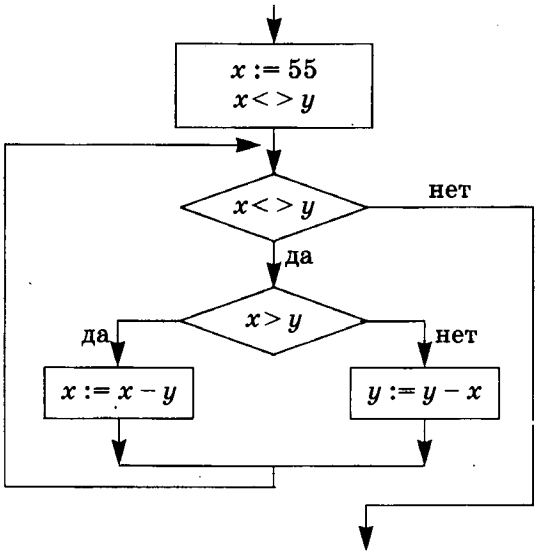
2) во внутреннем и во внешнем циклах ДЛЯ должны использоваться разные переменные-счётчики.

Операторы досрочного завершения цикла

Оператор языка Паскаль	Описание
`continue`	*Оператор продолжения* — выполнение данного оператора прекращает текущее выполнение тела цикла и передаёт управление на проверку условия цикла.
`break`	*Оператор прерывания* — выполнение данного оператора прекращает выполнение цикла и передаёт управление на следующий оператор после цикла.

Разбор типовых задач

Задача 1*. Определите значение целочисленной переменной x после выполнения следующего фрагмента программы:



1) 1 2) 5 3) 10 4) 15

Решение

Подобные задачи (так же, как ранее разобранные задания на ветвление) решаются путём построения таблицы трассировки.

Операторы в блоке	x	y
$x := 55$ $y := 75$	**55**	**75**
$x < > y$ Условие выполнено — выполняется ветвь «Да»	55	75
$x > y$ Условие не выполнено — выполняется ветвь «Нет»	55	75
$y := y - x$	55	**20**
$x < > y$ Условие выполнено — выполняется ветвь «Да»	55	20
$x > y$ Условие выполнено — выполняется ветвь «Да»	55	20
$x := x - y$	**35**	20
$x < > y$ Условие выполнено — выполняется ветвь «Да»	35	20
$x > y$ Условие выполнено — выполняется ветвь «Да»	35	20
$x := x - y$	**15**	20
$x < > y$ Условие выполнено — выполняется ветвь «Да»	15	20
$x > y$ Условие не выполнено — выполняется ветвь «Нет»	15	20

Окончание табл.

Операторы в блоке	x	y
$y := y - x$	15	5
$x <> y$ Условие выполнено — выполняется ветвь «Да»	15	5
$x > y$ Условие выполнено — выполняется ветвь «Да»	15	5
$x := x - y$	10	5
$x <> y$ Условие выполнено — выполняется ветвь «Да»	10	5
$x > y$ Условие выполнено — выполняется ветвь «Да»	10	5
$x := x - y$	5	5
$x <> y$ Условие не выполнено — выполняется ветвь «Нет»	5	5
Прекращение работы программы	5	5

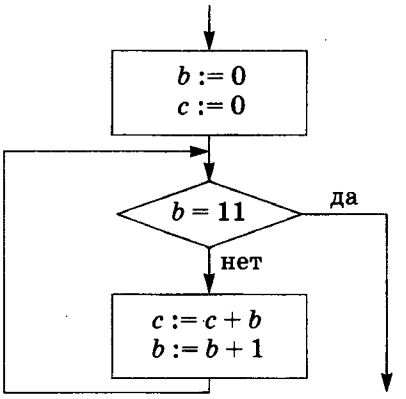
Итак, переменная x равна 5.

Ответ: $x = 5$ (вариант ответа №2).

Таблицу трассировки можно записывать упрощённо (подразумевая, что условие $x <> y$ выполняется, а при его первом же невыполнении трассировка прекращается):

Оператор в блоке	$x > y$?	x	y
$x := 55$ $y := 75$		55	75
$x > y$	Нет: $y := y - x$	55	20
$x > y$	Да: $x := x - y$	55	20
$x > y$	Да: $x := x - y$	15	20
$x > y$	Нет: $y := y - x$	15	5
$x > y$	Да: $x := x - y$	10	5
$x > y$	Да: $x := x - y$	5	5

Задача 2*. Определите значение переменной *c* после выполнения фрагмента алгоритма:



Примечание: знаком := обозначена операция присваивания.

1) 1 2) 45 3) 55 4) 66

Решение

Строится таблица трассировки (стрелками для большей ясности указано, откуда берутся значения при выполнении оператора *c* := *c* + *b*).

Операторы в блоке	*b*	*c*
b := 0 *c* := 0	**0**	**0**
b = 11 Условие не выполнено — выполняется ветвь «Нет»	0 ⇑	0
c := *c* + *b*;	0	**0**
b := *b* + 1;	**1**	0
b = 11 Условие не выполнено — выполняется ветвь «Нет»	1 ⇑	0
c := *c* + *b*;	1	**1**
b := *b* + 1;	**2**	1
b = 11 Условие не выполнено — выполняется ветвь «Нет»	2 ⇑	1
c := *c* + *b*;	2	**3**
b := *b* + 1;	**3**	3
b = 11 Условие не выполнено — выполняется ветвь «Нет»	3 ⇑	3
c := *c* + *b*;	3	**6**
b := *b* + 1;	**4**	6
b = 11 Условие не выполнено — выполняется ветвь «Нет»	4 ⇑	6
c := *c* + *b*;	4	**10**
b := *b* + 1;	**5**	10

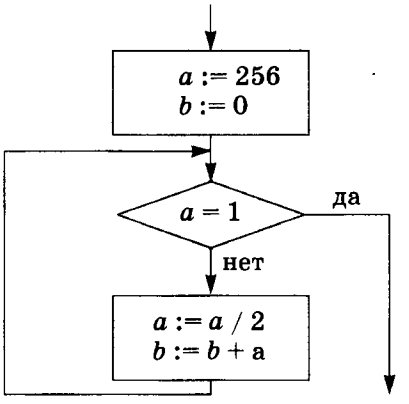
Окончание табл.

Операторы в блоке	b	c
$b = 11$ Условие не выполнено — выполняется ветвь «Нет»	5 ⇑	10
$c := c + b;$	5	**15**
$b := b + 1;$	**6**	15
$b = 11$ Условие не выполнено — выполняется ветвь «Нет»	6 ⇑	15
$c := c + b;$	6	**21**
$b := b + 1;$	**7**	21
$b = 11$ Условие не выполнено — выполняется ветвь «Нет»	7 ⇑	21
$c := c + b;$	7	**28**
$b := b + 1;$	**8**	28
$b = 11$ Условие не выполнено — выполняется ветвь «Нет»	8 ⇑	28
$c := c + b;$	8	**36**
$b := b + 1;$	**9**	36
$b = 11$ Условие не выполнено — выполняется ветвь «Нет»	9 ⇑	36
$c := c + b;$	9	**45**
$b := b + 1;$	**10**	45
$b = 11$ Условие не выполнено — выполняется ветвь «Нет»	10 ⇑	45
$c := c + b;$	10	**55**
$b := b + 1;$	**11**	55
$b = 11$ Условие выполнено — выполняется ветвь «Да»	11 ⇑	55
Прекращение работы программы	11	55

Итак, по завершении работы программы значение переменной c равно 55.

Ответ: $c = 55$ (вариант ответа №3).

Задача 3*. Запишите значение переменной b после выполнения фрагмента алгоритма:



Примечание: знаком ":=" обозначена операция присваивания. В бланк ответа впишите только число.

Решение

Строится таблица трассировки.

Операторы в блоке	a	b
$a := 256$ $b := 0$	**256**	**0**
a=1 Условие не выполнено — выполняется ветвь «Нет»	256 ⇑	0
$a := a / 2$	**128**	0
$b := b + a$;	0	**128**
$a = 1$ Условие не выполнено — выполняется ветвь «Нет»	128 ⇑	128
$a := a / 2$	**64**	128
$b := b + a$;	0	**192**
$a = 1$ Условие не выполнено — выполняется ветвь «Нет»	64 ⇑	192
$a := a / 2$	**32**	192
$b := b + a$;	0	**224**
$a = 1$ Условие не выполнено — выполняется ветвь «Нет»	32 ⇑	224
$a := a / 2$	**16**	224
$b := b + a$;	0	**240**
$a = 1$ Условие не выполнено — выполняется ветвь «Нет»	16 ⇑	240
$a := a / 2$	**8**	240
$b := b + a$;	0	**248**

Окончание табл.

Операторы в блоке	*a*	*b*
a = 1 Условие не выполнено — выполняется ветвь «Нет»	8 ⇑	248
a := *a* / 2	**4**	248
b := *b* + *a*;	0	**252**
a = 1 Условие не выполнено — выполняется ветвь «Нет»	4 ⇑	252
a := *a* / 2	**2**	252
b := *b* + *a*;	0	**254**
a := *a* / 2	**1**	
b := *b* + *a*;	0	**255**
a = 1 Условие выполнено — выполняется ветвь «Да»	1 ⇑	255
Прекращение работы программы	1	255

Таким образом, по окончании работы программы значение переменной *b* равно 255.

Ответ: *b* = 255.

Задача 4*. Определите, что будет напечатано в результате работы следующего фрагмента программы:

Бейсик

```
Dim k, s As Integer
s = 0
k = 0
While s < 1024
s = s + 10
k = k + 1
End While
Console.Write(k)
```

Паскаль

```
Var k, s: integer;
BEGIN
s := 0;
k := 0;
while s < 1024 do
begin
s := s + 10;
k := k + 1;
end;
write(k);
END.
```

Си

```
{
int к, s;
s = 0;
к = 0;
while (s < 1024) {
s = s + 10;
к = k + 1;
}
printf("%d", k); }
```

Алгоритмический язык

```
нач
цел к, s
s := 0
k := 0
нц пока s < 1024
s := s + 10; k := k + 1
кц
вывод к
кон
```

Решение

Строится таблица трассировки.

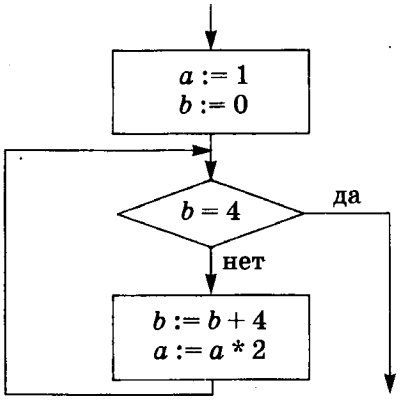
Операторы	s	k
$s := 0;$ $k := 0;$	**0**	**0**
while $s < 1024$ do	0	0
Условие истинно — цикл выполняется	⇑	
$s := s + 10;$	**10**	0
$k := k + 1;$	10	**1**
while $s < 1024$ do	10	1
Условие истинно — цикл выполняется	⇑	
$s := s + 10;$	**20**	1
$k := k + 1;$	20	**2**
while $s < 1024$ do	20	2
Условие истинно — цикл выполняется	⇑	
$s := s + 10;$	**30**	2
$k := k + 1;$	30	**3**
. . . . .	⇑	
while $s < 1024$ do	1000	100
Условие истинно — цикл выполняется	⇑	
$s := s + 10;$	**1010**	100
$k := k + 1;$	1010	**101**
while $s < 1024$ do	1010	101
Условие истинно — цикл выполняется	⇑	
$s := s + 10;$	**1020**	101
$k := k + 1;$	1020	**102**
while $s < 1024$ do	1020	102
Условие истинно — цикл выполняется	⇑	
$s := s + 10;$	**1030**	102
$k := k + 1;$	1030	**103**
while $s < 1024$ do	1030	103
Условие ложно — цикл прекращается	⇑	
write(k);		103

Таким образом, выводится значение переменной k, равное 10

Ответ: $k = 103$.

Задачи для самостоятельного решения

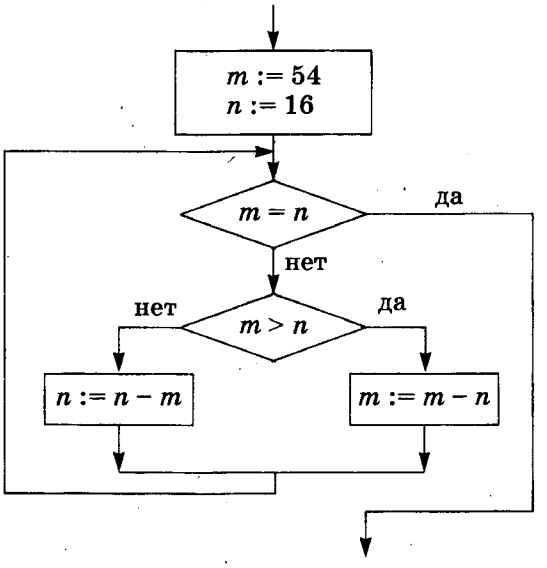
1*. Определите значение переменной a после выполнения фрагмента алгоритма:



Примечание: знаком * обозначено умножение, знаком := обозначена операция присваивания.

1) 8 2) 16 3) 32 4) 12

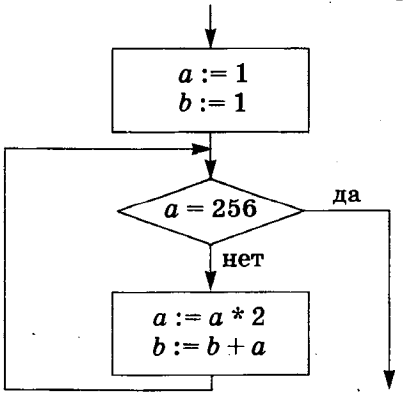
2*. Определите значение переменной m после выполнения фрагмента алгоритма.



Примечание: знаком := обозначена операция присваивания.

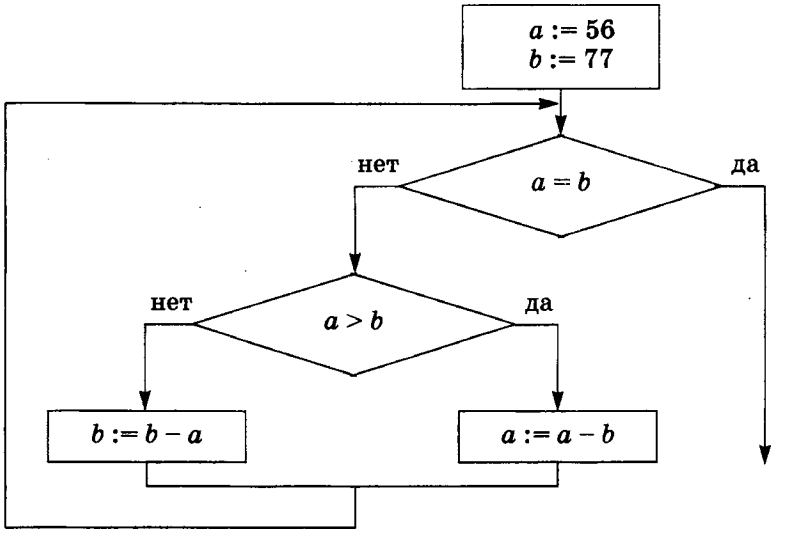
1) 1 2) 2 3) 6 4) 16

3*. Запишите значение переменной b после выполнения фрагмента алгоритма:



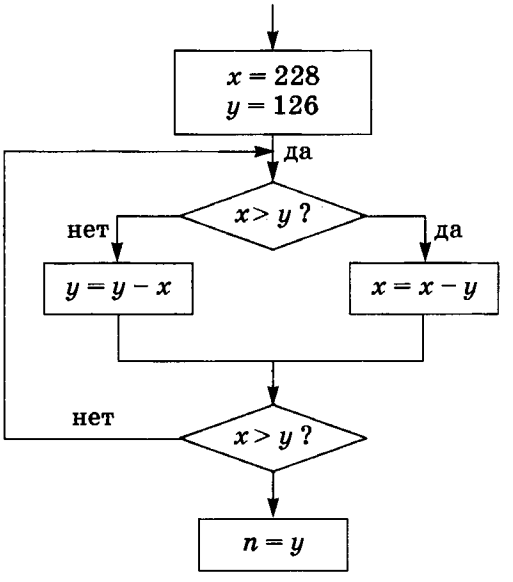
Примечание: знаком := обозначена операция присваивания, знаком * обозначена операция умножения.

4*. Запишите значение переменной a после выполнения фрагмента алгоритма:



Примечание: знаком := обозначена операция присваивания. В бланк ответов впишите только число.

5. Запишите значение переменной n после выполнения фрагмента алгоритма:



Ответы для самопроверки

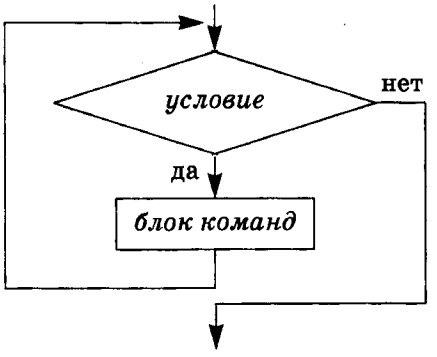
Задача	Ответ
1	2
2	2
3	511
4	7
5	6

Программирование

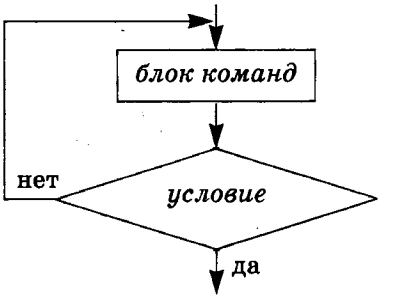
B7 Циклы: анализ алгоритмов

Конспект

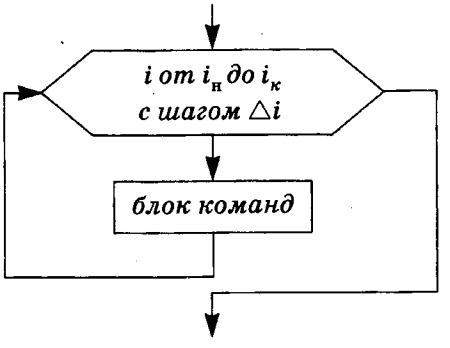
Цикл с предусловием (цикл ПОКА)



Цикл с постусловием (цикл ДО)



Цикл с параметром (цикл со счётчиком, цикл ДЛЯ)



Конструкции циклов в языке Паскаль

1. Цикл с предусловием:

```
while <условие>
  begin
   <блок команд>
  end;
```

или

```
do while <условие> do <команда>;
```

Цикл с постусловием:

```
repeat
  <команды>
until <условие>;
```

Цикл со счётчиком:

Цикл с шагом 1	Цикл с шагом −1

for <i> := <$i_н$> to <$i_к$> do
begin
<блок команд>
end;
или
for <i> := <$i_н$> to <$i_к$> do <команда>;
	for <i> := <$i_к$> downto <$i_н$> do
	begin
	<блок команд>
	end;
	или
	for <i> := <$i_к$> downto <$i_н$> do
	<команда>;

Операторы досрочного завершения цикла

Оператор языка Паскаль	Описание
continue	*Оператор продолжения* — выполнение данного оператора прекращает текущее выполнение тела цикла и передаёт управление на проверку условия цикла.
break	*Оператор прерывания* — выполнение данного оператора прекращает выполнение цикла и передаёт управление на следующий оператор после цикла.

Разбор типовых задач

Задача 1. Получив на вход число x, программа печатает два числа a и b. Укажите наибольшее из таких чисел x, при вводе которых программа напечатает сначала число 3, а потом число 35.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0;
  b := 1;
  while x > 0 do begin
    a := a + 1;
    b := b * (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

Решение

В данном случае трассировка бессмысленна, так как исходное данное неизвестно (его и нужно определить). Поэтому решение заключается в общем анализе алгоритма.

1. Программа должна первым напечатать значение 3. Поскольку первым стоит оператор вывода `writeln(a)`, по завершении выполнения цикла переменная a должна быть равна 3. Изначально $a = 0$. Изменение этой переменной в цикле производится оператором `a := a + 1`, т. е. значение a определяет количество проходов цикла.

2. Таким образом, цикл выполняется 3 раза. При этом второе печатаемое число (значение переменной b) равно 35.

Анализируется цикл целиком:

```
while x > 0 do begin
  a := a + 1;
  b := b *(x mod 10);
  x := x div 10;
end;
```

Конструкция, включающая в себя сначала вычисление остатка от деления числа x на 10, а затем меняющая значение x на результат целочисленного деления x на 10, в цикле пока $x > 0$, является типовой и реализует последовательное выделение цифр числа x.

Таким образом, данный цикл выполняет поочередное (справа налево — от младших разрядов к старшим) выделение цифр числа x и вычисление в переменной b произведения этих цифр.

3. Итак, нужно найти наибольшее значение числа x, у которого произведение цифр равно 35, а количество цифр равно трём (так как цикл выделения каждой цифры выполнялся трижды).

Число 35 можно разложить на множители как $5 \cdot 7$, причем эти множители простые. Чтобы получить три множителя, можно добавить к ним единицу: $35 = 5 \cdot 7 \cdot 1$.

Таким образом, для получения заданных результатов (чисел 3 и 35) годятся значения x, равные всем возможным комбинациям цифр 1, 5 и 7. Наибольшее же из таких чисел равно, очевидно, 751.

Ответ: $x = 751$.

Важно не забывать, что при определении исходного значения x надо учитывать не только разложение второго из заданных чисел (здесь — 35) на простые множители, но и количество походов цикла, определяемое первым из заданных чисел (здесь — 3).

Задача 2. Получив на вход число x, программа печатает два числа a и b. Укажите наименьшее из таких чисел x, при вводе которых программа напечатает сначала число 4, а потом число 24.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0;
  b := 1;
  while x > 0 do begin
    a := a + 1;
    b := b * (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

Решение

Решение заключается в общем анализе алгоритма.

1. Программа должна первым напечатать значение 4, значит, по завершении выполнения цикла переменная a должна быть равна 4. Изначально $a = 0$, изменение этой переменной в цикле производится оператором `a := a + 1`, значит цикл должен быть выполнен 4 раза. При этом второе печатаемое число (значение переменной b) равно 24.

2. Анализируется цикл целиком:

```
while x > 0 do begin
  a := a + 1;
  b := b * (x mod 10);
  x := x div 10;
end;
```

Конструкция, включающая в себя сначала вычисление остатка от деления числа x на 10, а затем меняющая значение x на результат целочисленного деления x на 10, в цикле пока $x > 0$, является типовой и реализует последовательное выделение цифр числа x.

Таким образом, данный цикл выполняет поочередное (справа налево — от младших разрядов к старшим) выделение цифр числа x и вычисление в переменной b произведения этих цифр.

3. Итак, нужно найти наименьшее значение числа x, у которого произведение цифр равно 24, а количество цифр равно четырём (так как цикл выделения каждой цифры выполнялся четыре раза).

Число 24 можно разложить на четыре однозначных множителя различными способами:

$24 = 4 \cdot 6 \cdot 1 \cdot 1 = 4 \cdot 3 \cdot 2 \cdot 1 = 2 \cdot 2 \cdot 6 \cdot 1 = 2 \cdot 2 \cdot 2 \cdot 3$.

Таким образом, в качестве исходного числа x подходят все возможные комбинации цифр: 4, 6, 1, 1 или 4, 3, 2, 1 или 2, 2, 6, 1 или 2, 2, 2, 3.

В каждой из полученных групп находятся наименьшие возможные значения x, которые можно получить из таких цифр, и сравниваются:

Цифры	4, 6, 1, 1	4, 3, 2, 1	2, 2, 6 ,1	2, 2, 2, 3
Наименьшее возможное число	1146	1234	1226	2223

Тогда наименьшее возможное значение x, удовлетворяющее условиям задачи, равно 1146.

Ответ: $x = 1146$.

Задача 3. Получив на вход число x, программа печатает два числа a и b. Укажите наибольшее из таких чисел x, при вводе которых программа напечатает сначала число 4, а потом число 7.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0;
  b := 0;
  while x > 0 do begin
    a := a + 1;
    b := b + (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

Решение

Решение заключается в общем анализе алгоритма.

1. Программа должна первым напечатать значение 4, значит, по завершении выполнения цикла переменная a должна быть равна 4. Изначально $a = 0$, изменение этой переменной в цикле производится оператором `a := a + 1`, значит цикл должен быть выполнен 4 раза.

2. Второе печатаемое число (значение переменной b) должно быть равно 7.

Анализируется цикл целиком:

```
while x > 0 do begin
  a := a + 1;
  b := b + (x mod 10);
  x := x div 10;
end;
```

Конструкция, включающая в себя сначала вычисление остатка от деления числа x на 10, а затем меняющая значение x на результат целочисленного деления x на 10, в цикле пока $x > 0$, является типовой и реализует последовательное выделение цифр числа x.

Таким образом, данный цикл выполняет поочередное (справа налево — от младших разрядов к старшим) выделение цифр числа x и вычисление в переменной b суммы этих цифр.

3. Итак, нужно найти наибольшее значение числа x, у которого сумма цифр равна 7, а количество цифр равно четырём (так как цикл выделения каждой цифры выполнялся четыре раза).

Число 7 можно представить как суммы различных цифр, например:

$7 = 1 + 1 + 1 + 4$;

$7 = 1 + 2 + 3 + 3$;

$7 = 2 + 2 + 2 + 1$; и т. д.

Наибольшее из возможных четырёхзначных значений x, сумма цифр которого была бы равна 7, равно 7000 (сумма цифр: $7 + 0 + 0 + 0$).

Ответ: $x = 7000$.

Задача 4. Получив на вход число x, программа печатает два числа a и b. Укажите наибольшее из таких чисел x, при вводе которых программа напечатает сначала число 3, а потом число 10.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0;
  b := 0;
  while x > 0 do begin
    a := a + 1;
    b := b + (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

Решение

Решение заключается в общем анализе алгоритма.

1. Программа должна первым напечатать значение 3, значит, по завершении выполнения цикла переменная a должна быть равна 3. Изначально $a = 0$, изменение этой переменной в цикле производится оператором `a := a + 1`, значит цикл должен быть выполнен 3 раза.

2. Второе печатаемое число (значение переменной b) должно быть равно 10.

Анализируется цикл целиком:

```
while x > 0 do begin
  a := a + 1;
  b := b + (x mod 10);
  x := x div 10;
end;
```

Конструкция, включающая в себя сначала вычисление остатка от деления числа x на 10, а затем меняющая значение x на результат целочисленного деления x на 10, в цикле пока $x > 0$, является типовой и реализует последовательное выделение цифр числа x.

Таким образом, данный цикл выполняет поочередное (справа налево — от младших разрядов к старшим) выделение цифр числа x и вычисление в переменной b суммы этих цифр.

3. Итак, нужно найти наибольшее значение числа x, у которого сумма цифр равна 10, а количество цифр равно трём (так как цикл выделения каждой цифры выполнялся три раза).

Очевидно, что воспользоваться той же идеей, что и в предыдущей задаче (когда сумма цифр формировалась как само значение суммы плюс требуемое количество нулей) здесь нельзя: 10 — это не цифра. Поэтому придётся искать разложение числа 10 как суммы трёх цифр, таких, что составленное из них число — максимально возможное.

Такое разложение выполняется: $10 = 9 + 1 + 0$.

Ответ: $x = 910$.

Задачи для самостоятельного решения

1. Получив на вход число x, программа печатает два числа a и b. Укажите наименьшее из таких чисел x, при вводе которых программа напечатает сначала число 5, а потом число 105.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0;
  b := 1;
  while x > 0 do begin
    a := a + 1;
    b := b * (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

2. Получив на вход число x, программа печатает два числа a и b. Укажите наибольшее из таких чисел x, при вводе которых программа дважды напечатает число 6.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0;
  b := 0;
  while x > 0 do begin
    a := a + 1;
    b := b + (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

3. Получив на вход число x, программа печатает два числа a и b. Укажите наименьшее из таких чисел x, при вводе которых программа напечатает два числа: 3 и 6.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0; b := 0;
  while x > 0 do begin
    a := a + 1;
    b := b * (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

4. Получив на вход число x, программа печатает два числа a и b. Укажите наименьшее из таких чисел x, при вводе которых программа дважды напечатает число 4.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0; b := 0;
  while x > 0 do begin
    a := a + 1;
    b := b + (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

5. Получив на вход число x, программа печатает два числа a и b. Укажите такое число x, которое представляет собой палиндром, является наименьшим из возможных и при вводе которого программа напечатает два числа: 5 и 45.
Примечание: число-палиндром — это симметричное число, которое читается одинаково слева направо и справа налево.

```
var x, a, b: integer;
  begin
  readln(x);
  a := 0; b := 0;
  while x > 0 do begin
    a := a + 1;
    b := b * (x mod 10);
    x := x div 10;
  end;
  writeln(a);
  write(b);
end.
```

Ответы для самопроверки

Задача	Ответ
1	11357
2	600000
3	123
4	1003
5	13531

Программирование

A12 Операции с массивами: анализ программ

Конспект

Массив

Во многих задачах требуется выполнять одни и те же операции с достаточно большим количеством данных одного и того же типа. Например, это может быть поиск требуемой фамилии в электронном телефонном справочнике, сортировка по возрастанию цен в прайс-листе интернет-магазина, вычисление средней годовой оценки в классе, изменение яркости растрового изображения путём изменения интенсивности цветовых составляющих для каждого пикселя и т. д.

Для таких применений предусмотрены составные типы данных, одним из наиболее широко используемых среди которых является тип «массив».

Одномерный массив представляет собой пронумерованную последовательность переменных одного и того же типа, имеющих общее **имя**. Обращение к конкретной переменной (**элементу массива**) производится по **имени массива** и порядковому номеру (**индексу**) нужного элемента в нём.

Двумерный массив можно рассматривать как одномерный массив, элементами которого являются одномерные массивы. Другое представление двумерного массива — **матрица**: прямоугольная или квадратная таблица, в которой элементы массива соответствуют ячейкам, а номера строк и столбцов представляют собой два индекса каждого элемента.

Аналогично, трёхмерный массив может быть рассмотрен как куб данных, состоящий из элементарных кубиков (элементов массива), каждый из которых имеет три индекса (т.е. трехмерный массив рассматривается как одномерный массив «слоёв» — матриц).

По тому же принципу могут быть определены массивы большей размерности.

Объявление массива в языке Паскаль

array [<диапазон индекса 1>, ..., <диапазон индексаN>] **of** <базовый тип>, где <диапазон индекса> — записанные через две точки начальное и конечное значения индексов элементов по соответствующей размерности (целые числа или символы); <базовый тип> — тип элементов массива.

Обмен местами элементов массива

Производится аналогично обмену значений двух обычных переменных (обычно — при помощи дополнительной «буферной» переменной).

Обработка элементов массива (определение максимума/минимума, вычисление суммы, произведения, среднего и пр.)

В цикле (для многомерного массива — во вложенных циклах) производится перебор элементов массива (полный или частичный — для фрагмента массива).

- *При поиске максимума/минимума* — за предполагаемый максимум/минимум берётся первый элемент массива либо константа, заведомо меньшая/большая любого элемента. Далее каждый очередной элемент массива сравнивается с предполагаемым максимумом/минимумом, и если этот элемент больше/меньше предполагаемого максимума/минимума, то значение этого элемента запоминается в качестве нового предполагаемого максимума/минимума. (Дополнительно при этом в отдельной переменной (переменных) может перезапоминаться индекс (индексы) очередного предполагаемого максимума/минимума.)
- *При вычислении суммы/произведения* — вначале переменной, выделенной для накопления суммы/произведения присваивается инициализационное значение (нуль — для суммы, единица — для произведения). Затем в цикле (либо вложенных циклах) выполняется перебор элементов массива и текущее значение суммы/произведения складывается/умножается на текущий элемент массива.
- *При вычислении среднего значения* — выполняется суммирование элементов массива, а затем полученная сумма делится на количество элементов в массиве.
- *При определении максимума/минимума, суммы, произведения, среднего значения элементов, удовлетворяющих заданному условию* (например, только положительных) — дополнительно добавляется условный оператор с соответствующим условием, и требуемое действие (проверка и переприсваивание предполагаемого максимума/минимума, сложение, умножение) выполняется только при истинности этого условия. При вычислении среднего значения также предусматривается отдельная переменная-счётчик, которая увеличивается на 1 каждый раз, когда к сумме добавляется очередной удовлетворяющий условию элемент массива, и после вычисления суммы она делится на значение этого счётчика (количество вошедших в сумму элементов).

Сортировка массива

Выполняется путём многократного просмотра массива, попарного сравнения его элементов между собой и при необходимости — соответствующей перестановки этих элементов местами (при сортировке по убыванию на первые места переносятся элементы с бóльшими значениями; при сортировке по убыванию — наоборот).

Существует несколько методов сортировки массивов, различающихся сложностью их алгоритма и скоростью работы.

Разбор типовых задач

Задача 1*. Значения двумерного массива задаются с помощью вложенного оператора цикла в представленном фрагменте программы

Бейсик	Паскаль	Алгоритмический язык

FOR n = 1 TO 5
FOR k = 1 TO 5
B(n,k) = n + k
NEXT k
NEXT n
	for n := 1 to 5 do
	for k := 1 to 5 do
	B[n,k] := n + k;
		нц для n от 1 до 5
		нц для k от 1 до 5
		B[n,k] = n + k
		кц
		кц

Чему будет равно значение B(2,4)?

1) 9 2) 8 3) 7 4) 6

Решение

Общая схема решения задач с массивами такова.

Для задач с массивами трассировку удобнее проводить на *модели массива*, имеющей вид таблицы: для одномерного массива — в виде одной строки с нужным числом пронумерованных ячеек, для двумерного — в виде таблицы-матрицы с пронумерованными строками и столбцами. После этого анализируется диапазон изменения в цикле индексов элемента массива и соответствующие элементы (если обрабатывается не весь массив, а его часть) выделяются на рисунке массива.

Наконец, указанное в цикле действие поочередно выполняется для каждого обрабатываемого элемента массива. В итоге получается искомый массив.

Если же, как в данной задаче, требуется определить значение только одного элемента, формируемое при первичном заполнении массива, то достаточно лишь подставить в соответствующий оператор нужные значения переменных-индексов и сразу получить ответ.

Пусть требуется определить значение элемента B(2,4).

Первый индекс равен 2 и в записи обращения к элементу массива выражен переменной n.

Второй индекс равен 4 и в записи обращения к элементу массива выражен переменной k.

Запись оператора, формирующего значения элементов массива: `B[n,k] := n + k`. В него подставляются нужные значения n и k: `B[2,4] := 2 + 4`. Отсюда видно, что значение элемента B(2,4) равно 6.

Ответ: B(2,4) = 6 (вариант ответа № 4).

Задача 2*. Следующий фрагмент программы записывает в переменную *Max* максимальный элемент в двумерном массиве *Dist* размера $N \times N$,заполненном целыми неотрицательными числами:

```
Max := 0;
for i := 1 to N do
for j := 1 to N do
     if Dist [i,j] > Max then Max := Dist [i,j];
```

На очень медленном компьютере эта программа при $N = 1000$ работала 5 секунд. Оцените время работы этой программы на том же компьютере при $N = 2000$.

1) 10 сек. 2) 20 сек. 3) 30 сек. 4) 40 сек.

Решение

Данная задача затрагивает достаточно сложный вопрос определения временной эффективности алгоритма. Подобные вопросы рассматриваются в курсе информатики далеко не всегда, — видимо, поэтому, разработчики ЕГЭ в более поздние годы не предлагали такие задания. Однако возможно, что подобные задачи будут возвращены в ЕГЭ после принятия нового ФГОС.

В данном случае решение не очень сложно.

Нетрудно видеть, что оператор `if Dist [i,j] > Max then Max := Dist [i,j]` (по смыслу — поиск максимума в массиве) выполняется в теле вложенных циклов, каждый из которых выполняется N раз. То есть, сложность данного алгоритма — квадратичная: количество выполняемых операций `if Dist [i,j] > Max then Max := Dist [i,j]` равно N^2.

Условно считается, что реализация самих циклов выполняется мгновенно; факт, что переприсваивание `Max := Dist [i,j]` выполняется не каждый раз, а только при соблюдении условия в `if`, игнорируется.

Тогда, если для $N = 1000$ время работы программы составило 5, то для вдвое бóльшего значения $N = 2000$ время работы программы возрастёт в 4 раза, т. е. составит 20 с.

Ответ: 20 с (вариант ответа №2).

Задача 3*. Все элементы двумерного массива A размером 10 × 10 элементов первоначально были равны 0. Затем значения элементов меняются с помощью вложенного оператора цикла в представленном фрагменте программы (*ниже представлена одна и та же программа, записанная на разных языках программирования*).

Бейсик	Паскаль	Алгоритмический язык

FOR n = 1 TO 4
FOR k = n TO 4
A(n,k)= A(n,k) + 1
A(k,n)= A(k,n) + 1
NEXT k
NEXT n
	for n := 1 to 4 do
	for k := n to 4 do
	begin
	A[n,k] := A[n,k] + 1;
	A[k,n] := A[k,n] + 1;
	end
		нц для n от 1 до 4
		нц для k от n до 4
		A[n,k] := A[n,k] + 1
		A[k,n] := A[k,n] + 1
		кц
		кц

Сколько элементов массива в результате будут равны 1?

1) 0 2) 16 3) 12 4) 4

Решение

1. Модель двумерного массива:

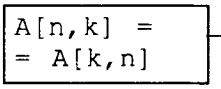
n \ k	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

2. Выделяется обрабатываемая область:

n \ k	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

3. Заполяется массив поэлементно, отслеживая изменение индексных переменных (ниже приводится только нужный фрагмент модели массива; при его выделении нужно быть внимательным — в зависимости от записи индексов элемента возможен выход значений индексов за пределы изменения значений индексных переменных в цикле).

1)

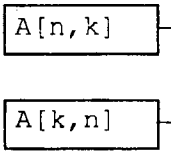


n \ k	1	2	3	4
1	2	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

$n = 1,\ k = n = 1$

Вначале на единицу увеличено значение A[n,k], т. е. A[1,1]. Затем на единицу должно быть увеличено значение A[k,n], т. е. A[1,1]. Поскольку это тот же самый элемент, его значение увеличивается на 1 ещё раз и становится равным 2. Это, очевидно, будет происходить всегда, когда $n = k$.

2)

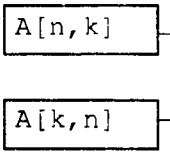


n \ k	1	2	3	4
1	2	1	0	0
2	1	0	0	0
3	0	0	0	0
4	0	0	0	0

$n = 1,\ k = 2$

Вначале на единицу увеличено значение A[n,k], т. е. A[1,2]. Затем на единицу увеличено значение A[k,n], т. е. A[2,1].

3)

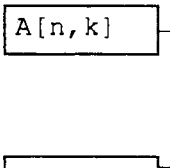


n \ k	1	2	3	4
1	2	1	1	0
2	1	0	0	0
3	1	0	0	0
4	0	0	0	0

$n = 1,\ k = 3$

Вначале на единицу увеличено значение A[n,k], т. е. A[1,3]. Затем на единицу увеличено значение A[k,n], т. е. A[3,1].

4)

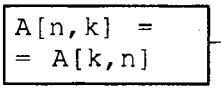


n \ k	1	2	3	4
1	2	1	1	1
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0

$n = 1,\ k = 4$

Вначале на единицу увеличено значение A[n,k], т. е. A[1,4]. Затем на единицу увеличено значение A[k,n], т. е. A[4,1].

5) Внутренний цикл завершён, выполняется новый проход внешнего цикла и внутренний цикл запускается заново:

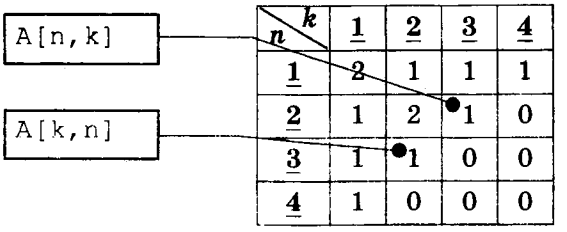


n \ k	1	2	3	4
1	2	1	1	1
2	1	2	0	0
3	1	0	0	0
4	1	0	0	0

$n = 2,\ k = n = 2$

Вначале на единицу увеличено значение A[n,k], т. е. A[2,2]. Затем на единицу должно быть увеличено значение A[k,n], т. е. снова A[2,2].

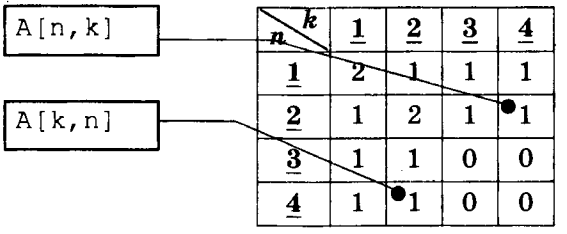
6)



$n = 2,\ k = 3$

Вначале на единицу увеличено значение A[n,k], т. е. A[2,3]. Затем на единицу увеличено значение A[k,n], т. е. A[3,2].

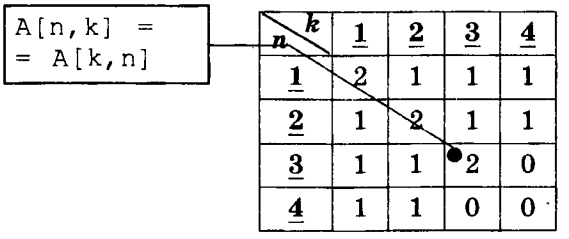
7)



$n = 2,\ k = 4$

Вначале на единицу увеличено значение A[n,k], т. е. A[2,4]. Затем на единицу увеличено значение A[k,n], т. е. A[4,2].)

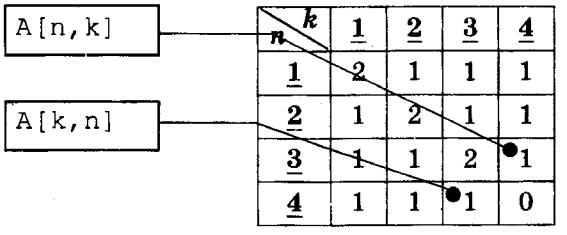
8) Внутренний цикл завершён, выполняется новый проход внешнего цикла и внутренний цикл запускается заново:



$n = 3,\ k = n = 3$

Вначале на единицу увеличено значение A[n,k], т. е. A[3,3]. Затем на единицу должно быть увеличено значение A[k,n], т. е. снова A[3,3].

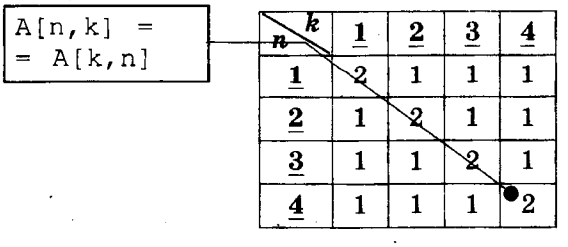
9)



$n = 3,\ k = 4$

Вначале на единицу увеличено значение A[n,k], т. е. A[3,4]. Затем на единицу увеличено значение A[k,n], т. е. A[4,3].

10) Внутренний цикл завершён, выполняется новый проход внешнего цикла и внутренний цикл запускается заново:



$n = 4,\ k = n = 4$

Вначале на единицу увеличено значение A[n,k], т. е. A[4,4]. Затем на единицу должно быть увеличено значение A[k,n], т. е. снова A[4,4].

11) Внутренний цикл завершён, внешний цикл также завершен.

4. В полученном массиве подсчитывается количество полученных единиц:
- всего ненулевых ячеек массива: $4 \cdot 4 = 16$ (остальные ячейки остались равны нулю и не интересуют);
- двойками заняты 4 ячейки;
- тогда единицы содержат $16 - 4 = 12$ ячеек.

Ответ: 12 элементов (вариант ответа №3).

Задача 4*. Стандартный алгоритм вычисления среднего арифметического элементов числового массива работает на массиве из миллиона элементов 0,5 сек. Оцените время работы того же алгоритма на том же компьютере, если длина массива 3 миллиона.

1) 1 сек. 2) 1,5 сек. 3) 3 сек. 4) 4,5 сек.

Решение

Это также задача на оценку сложности алгоритма.

1. Поскольку не указано, что массив многомерный, подразумевается одномерный массив.

2. При вычислении среднего значения выполняется однократный проход по массиву и текущий его элемент прибавляется к значению переменной, в которой накапливается сумма. По завершении цикла выполняется однократное деление суммы на количество элементов в массиве. Таким образом, время выполнения программы зависит от количества элементов массива линейно (т. е. прямо пропорционально N).

3. Если для $N = 1000000$ элементов программа работала 0,5 с, то для $N = 3000000$ элементов (т. е. второе большего) длительность работы программы также будет втрое больше, т. е. 1,5 сек.

Ответ: 1,5 сек. (вариант ответа №2).

Задача 5*. Значения двумерного массива размера 7×7 задаются с помощью вложенного оператора цикла в представленном фрагменте программы

Бейсик	Паскаль	Алгоритмический язык

FOR n = 1 TO 7
FOR k = 1 TO 7
B(n,k) = k - n
NEXT k
NEXT n
	for n := 1 to 7 do
	for k := 1 to 7 do
	B[n,k] := k - n;
		нц для n от 1 до 7
		нц для k от 1 до 7
		B[n,k] = k - n
		кц
		кц

Сколько элементов массива будут иметь положительные значения?

1) 49 2) 28 3) 21 4) 7

Решение

1. Модель двумерного массива:

n \ k	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0

2. Массив обрабатывается весь (диапазон изменений обеих цикловых переменных совпадает с размерами массива).

3. Заполняется массив, «пробегая» по строкам (внешний цикл), а в пределах каждой строки — слева направо (внутренний цикл). При этом отслеживаются значения цикловых переменных и вычисляется разность $k - n$.

n \ k	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6
2	−1	0	1	2	3	4	5
3	−2	−1	0	1	2	3	4
4	−3	−2	−1	0	1	2	3
5	−4	−3	−2	−1	0	1	2
6	−5	−4	−3	−2	−1	0	1
7	−6	−5	−4	−3	−2	−1	0

Итого получается:

- на главной диагонали матрицы (главная диагональ — ячейки, в которых оба индекса равны друг другу) все ячейки содержат нули;
- в верхней части матрицы (над главной диагональю) все ячейки содержат положительные значения;
- в нижней части матрицы (под главной диагональю) все ячейки содержат отрицательные значения.

Всего ячеек (элементов массива) равно $7 \cdot 7 = 49$.

Главную диагональ составляют 7 элементов, тогда остальных элементов 42.

Верхняя и нижняя части матрицы по количеству элементов одинаковы и составляют 21 элемент каждая.

Тогда количество элементов массива с положительными значениями равно 21.

Ответ: 21 элемент (вариант ответа №3).

Задача 6*. Значения двух массивов A[1..100] и B[1..100] задаются с помощью следующего фрагмента программы:

Бейсик	Паскаль	Алгоритмический

FOR n = 1 TO 100
A(n) = n - 10
NEXT n
FOR n = 1 TO 100
B(n) = A(n) * n
NEXT n
	for n := 1 to 100 do
	  A[n] := n - 10;
	for n := 1 to 100 do
	  B[n] := A[n] * n
		нц для n от 1 до 100
		  A[n] = n - 10
		кц
		нц для n от 1 до 100
		  B[n] = A[n] * n
		кц

Сколько элементов массива B будут иметь положительные значения?

1) 10 2) 50 3) 90 4) 100

Решение

Модели массивов из-за их большой длины полностью вычертить сложно. Поэтому нужно постараться обойтись устными рассуждениями.

1. При заполнении массива A значение индекса n меняется от 1 до 100, и в n-й элемент записывается значение, равное n — 10. Тогда:

- в элементы массива A с индексами от 1 до 9 будут записаны отрицательные значения (−9, −8, −7, −6, −5, −4, −3, −2, −1);
- в элемент массива A с индексом 10 будет записан нуль;
- во все остальные элементы массива A будут записаны положительные элементы (начиная с 1 для A[11] и до 90 в A[100]).

2. При заполнении массива B в каждый его n-й элемент записывается произведение n-го элемента массива A и значения n. Поскольку значение n всегда положительно (n меняется в цикле от 1 до 100), знак такого произведения определяется знаком соответствующего элемента массива A, т. е. количества положительных, отрицательных и нулевых элементов в массиве B точно такие же, как в массиве A.

3. В массиве A положительными являются элементы с A[11] до A[100] (т.е. все элементы массива A, кроме первых десяти). Их 90. Следовательно, в массиве B положительными также будут 90 элементов (с B[11] по B[100]).

Ответ: 90 элементов (вариант ответа №3).

Задача 7*. Значения двух массивов A[1..100] и B[1..100] задаются с помощью следующего фрагмента программы:

Бейсик	Паскаль	Алгоритмический язык
FOR n = 1 TO 100 A(n) = (n - 80) * (n - 80) NEXT n FOR n = 1 TO 100 B(101-n) = A(n) NEXT n	for n := 1 to 100 do A[n] := (n - 80) * n - 80); for n := 1 to 100 do B[101-n] := A[n];	нц для n от 1 до 100 A[n] = (n - 80) * n - 80) кц нц для n от 1 до 100 B[101-n] = A[n] кц

Какой элемент массива B будет наибольшим?

1) B[1]
2) B[21]
3) B[80]
4) B[100]

Решение

Как и в предыдущей задаче, постараемся обойтись устными рассуждениями.

1. При заполнении массива A значение n меняется от 1 до 100. При этом значение $(n - 80)$ меняется от −79 до 20. Однако при вычислении значения n-го элемента массива A значение $(n - 80)$ умножается само на себя (т. е. возводится в квадрат); при этом число всегда получается положительным, а его величина зависит от модуля значения $(n - 80)$.

Наибольшее по модулю значение разность $(n - 80)$ имеет для $n = 1$ (все остальные значения этой разности для n от 2 до 100 меньше 79). Следовательно, наибольшим будет элемент массива A[1].

2. Заполнение массива B производится элементами массива A, взятыми в обратном порядке:

n	1	2	3	...	98	99	100
B[101−n]:=A[n]	B[100]:=A[1]	B[99]:=A[2]	B[98]:=A[3]	...	B[3]:=A[98]	B[2]:=A[99]	B[1]:=A[100]

Следовательно, если в массиве A наибольшее значение имел элемент A[1], то в массиве B он попадёт на последнее место. Значит, в массиве B наибольшим будет элемент B[100].

Ответ: элемент B[100] (вариант ответа №4).

Задача 8*. Дан фрагмент программы, обрабатывающей двухмерный массив A размера n × n.

Бейсик	Паскаль	Алгоритмический язык

k = 1
FOR i = 1 TO n
c = A(i,i)
A(i,i) = A(k,i)
A(k,i) = c
NEXT i
	k := 1;
	for i := 1 to n do
	begin
	c := A[i,i]:
	A[i,i] := A[k,i];
	A[k,i] := c
	end
		k := 1
		нц для i от 1 до n
		c := A[i,i]
		A[i,i] := A[k,i]
		A[k,i] := c
		кц

Представим массив в виде квадратной таблицы, в которой для элемента массива A[i,j] величина *i* является номером строки, а величина *j* — номером столбца, в котором расположен элемент. Тогда данный алгоритм меняет местами

1) два столбца в таблице
2) две строки в таблице
3) элементы диагонали и *k*-ой строки таблицы
4) элементы диагонали и *k*-го столбца таблицы

Решение

Анализируется структура программы:

```
k := 1;
for i := 1 to n do
begin
  c := A[i,i]:
  A[i,i] := A[k,i];
  A[k,i] := c
end
```

Массив задан двумерный, цикл имеется только один, и он меняет от 1 до *n* значение переменной-индекса *i*. Значение же переменной-индекса *k* задано константой (1) и не меняется.

Тогда тело цикла:

```
c := A[i,i];
A[i,i] := A[k,i];
A[k,i] := c,
```

очевидно, реализует обмен местами (через буферную переменную *c*) элементов массива A[i,i] и A[k,i], т. е. элементов A[i,i] (лежащих на главной диагонали) и A[1,i] (элементов первой строки, так как по условию задачи, первый индекс массива — это номер строки).

Следовательно, правилен вариант ответа №3: программа меняет элементы диагонали и *k*-й строки таблицы.

Ответ: вариант ответа №3.

Задача 9*. В программе используется одномерный целочисленный массив A с индексами от 0 до 10. Ниже представлен фрагмент программы, записанный на разных языках программирования, в котором значения элементов сначала задаются, а затем меняются.

Бейсик	Паскаль

FOR i = 0 TO 10
A(i) = i
NEXT i
FOR i = 0 TO 10
A(10-i) = A(i)
A(i) = A(10-i)
NEXT i
	for i := 0 to 10 do
	  A[i] := i;
	for i := 0 to 10 do
	begin
	  A[10-i] := A[i];
	  A[i] := A[10-i];
	end;

Си	Алгоритмический язык

`for (i = 0; i <= 10; i++)`
`  A[i] = i;`
`for (i = 0; i <= 10; i++)`
`{`
`  A[10-i] = A[i];`
`  A[i] = A[10-i];`
`}`
	`нц для i от 0 до 10`
	`  A[i] := i`
	`кц`
	`нц для i от 0 до 10`
	`  A[10-i] := A[i]`
	`  A[i] := A[10-i]`
	`кц`

Чему будут равны элементы этого массива после выполнения фрагмента программы?

1) 10 9 8 7 6 5 4 3 2 1 0
2) 0 1 2 3 4 5 6 7 8 9 10
3) 10 9 8 7 6 5 6 7 8 9 10
4) 0 1 2 3 4 5 4 3 2 1 0

Решение

1. Модель массива A:

0	1	2	3	4	5	6	7	8	9	10

2. Массив обрабатывается весь (диапазон изменения цикловой переменной совпадает с размером массива).

3. Заполнение массива: каждому элементу присваивается значение, равное его индексу:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9	10

4. Изменение массива выполняется стоящей в теле второго цикла парой операторов:

```
A[10-i] := A[i];
A[i] := A[10-i];
```

Эта алгоритмическая конструкция *похожа* на обмен местами двух элементов (i-го и $(10-i)$-го), но **не является обменом**, поскольку здесь не используется буферная переменная! Здесь просто сначала в $(10-i)$-й элемент заносится значение i-го элемента (а прежнее значение теряется), а затем выполняется обратное присваивание, которое, очевидно, уже ничего не меняет.

Эта операция выполняется последовательно для всех значений i:

• $i = 0$:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9	0

• $i = 1$:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	1	0

• $i = 2$:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	2	1	0

• $i = 3$:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	3	2	1	0

• $i = 4$:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	**4**	5	**4**	3	2	1	0

• $i = 5$:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	**5**	4	3	2	1	0

При обмене элемента самого с собой никаких изменений не происходит.

• $i = 6$:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	**4**	5	**4**	3	2	1	0

Очевидно, при дальнейших операциях массив не изменяется.

• $i = 7$:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	**3**	4	5	4	**3**	2	1	0

• $i = 8$:

0	1	2	3	4	5	6	7	8	9	10
0	1	**2**	3	4	5	4	3	**2**	1	0

• $i = 9$:

0	1	2	3	4	5	6	7	8	9	10
0	**1**	2	3	4	5	4	3	2	**1**	0

• $i = 10$:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	4	3	2	1	**0**

Таким образом, в результате получается массив из элементов: 0, 1, 2, 3, 4, 5, 4, 3, 2, 1, 0.
Ответ: массив 0, 1, 2, 3, 4, 5, 4, 3, 2, 1, 0 (вариант ответа №4).

Задача 10*. В программе используется одномерный целочисленный массив A с индексами от 0 до 9. Ниже представлен фрагмент программы, записанный на разных языках программирования, в котором значения элементов сначала задаются, а затем меняются.

Бейсик	Паскаль

`For i = 0 To 9`
`A.SetValue(9-i, i)`
`Next`
`For i = 0 To 4`
`K = A.GetValue(i)`
`A.SetValue(A.GetValue(9-i), i)`
`A.SetValue(k, 9-i)`
`Next`
	`for i := 0 to 9 do`
	`A[i] := 9-i;`
	`for i := 0 to 4 do`
	`begin`
	`k := A[i];`
	`A[i] := A[9-i];`
	`A[9-i] := k;`
	`end;`

Си	Алгоритмический язык

`for (i = 0; i <= 9; i++)`
`A[i] = 9 - i;`
`for (i = 0; i <= 4; i++)`
`{`
`k = A[i];`
`A[i] = A[9-i];`
`A[9-i] = k;`
`}`
	`нц для i от 0 до 9`
	`A[i]:=9-i`
	`кц`
	`нц для i от 0 до 4`
	`k := A[i]`
	`A[i] := A[9-i]`
	`A[9-i] := k`
	`кц`

Чему будут равны элементы этого массива после выполнения фрагмента программы?

1) 9 8 7 6 5 4 3 2 1 0
2) 0 1 2 3 4 5 6 7 8 9
3) 9 8 7 6 5 5 6 7 8 9
4) 0 1 2 3 4 4 3 2 1 0

Решение

1. Модель массива A:

0	1	2	3	4	5	6	7	8	9

2. Массив при заполнении обрабатывается весь (диапазон изменения цикловой переменной совпадает с размером массива), а при изменении обрабатывается только часть массива (для i от 0 до 4).

3. Заполнение массива: каждому элементу присваивается значение, равное разности 9 и его индекса:

0	1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1	0

4. Изменение массива выполняется стоящей в теле второго цикла парой операторов:

```
k := A[i];
A[i] := A[9-i];
A[9-i] := k;
```

Эта алгоритмическая конструкция — обмен местами двух элементов (i-го и $(9-i)$-го) через буферную переменную k.

Эта операция выполняется последовательно для всех значений i:

• $i = 0$:

0	1	2	3	4	5	6	7	8	9
0	8	7	6	5	4	3	2	1	**9**

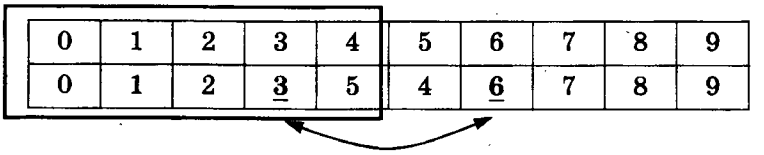
• $i = 1$:

0	1	2	3	4	5	6	7	8	9
0	**1**	7	6	5	4	3	2	**8**	9

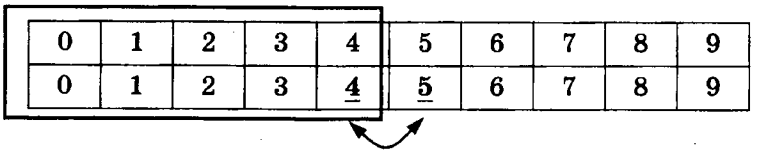
• $i = 2$:

0	1	2	3	4	5	6	7	8	9
0	1	**2**	6	5	4	3	**7**	8	9

• $i = 3$:



• $i = 4$:



То есть элементы массива меняются местами первый — с последним, второй — с предпоследним и т.д. В результате получается массив 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Ответ: массив 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (вариант ответа №2).

Будьте внимательны при определении обрабатываемой части массива! Если бы данный массив был обработан целиком, то начиная с значения $i = 5$ происходил бы обратный обмен значений элементов, в результате чего получился бы вновь исходный массив.

Задача 11. В программе используется одномерный целочисленный массив A с индексами от 1 до 10. Ниже представлен фрагмент программы, записанный на Паскале, в котором значения элементов сначала задаются, а затем меняются.

Паскаль

```
for i := 1 to 10 do A[i] := i;
for i := 1 to 5 do A[i] := A[6-i];
for i := 6 to 10 do A[i] := A[11-i];
```

Чему будут равны элементы этого массива после выполнения фрагмента программы?

1) 5 4 3 4 5 5 4 3 4 5
2) 5 4 3 2 1 10 9 8 7 6
3) 1 2 3 4 5 5 4 3 2 1
4) 5 4 3 2 1 1 2 3 4 5

Решение

Для решения этой задачи нужно «расчертить» на бумаге табличную строку из 10 ячеек (соответствующую искомому массиву), а затем для каждого из трех имеющихся в приведенной программе циклов аккуратно и внимательно заполнить ячейки этой таблицы вычисленными значениями.

1. Цикл `for i := 1 to 10 do A[i] := i;`

Здесь все ячейки массива заполняются своими порядковыми номерами:

1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

2. Цикл `for i := 1 to 5 do A[i] := A[6-i];`

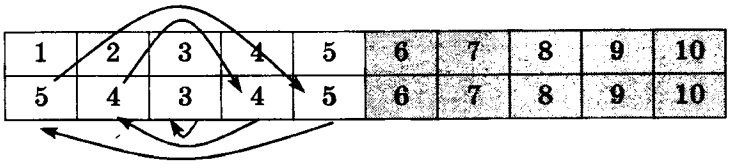
Обрабатываются (т. е. перезаписываются, меняя свои значения) элементы с номерами от 1 до 5, тогда как остальные элементы в правой части массива (с номерами от 6 до 10) остаются неизменными.

В ходе обработки каждый i-й элемент (для $i = 1 \ldots 5$) заменяется на значение элемента с номером, равным $6 - i$, тогда:

• в 1-й элемент записывается значение $(6 - 1 = 5)$-го элемента,
• во 2-й элемент — значение $(6 - 2 = 4)$-го,

- в 3-й — значение (6 – 3 = 3)-го, т. е. 3-й элемент остается без изменения,
- в 4-й элемент нужно записать значение (6 – 4 = 2)-го элемента, но при этом следует помнить, что ранее он уже был изменен; в него перед этим занесли первоначальное значение того же самого 4-го элемента;
- в 5-й элемент надо занести значение (6 – 5 = 1)-го элемента, которое ранее тоже было заменено на первоначальное значение элемента №5.

Получается массив:



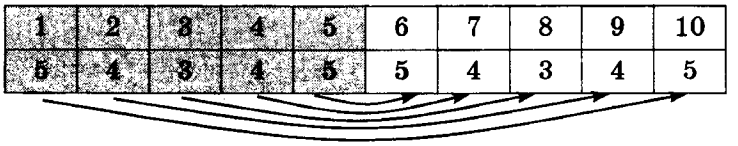
3. Цикл `for i := 6 to 10 do A[i] := A[11-i];`

Аналогично, в данном цикле обрабатываются элементы только части массива — с номерами от 6 до 10, тогда как ранее обработанные элементы №№ 1—5 в левой части массива остаются неизменными.

В ходе обработки каждый i-й элемент (для $i = 6 \ldots 10$) заменяется на значение элемента с номером, равным $11 - i$:

- в 6-й элемент записывается значение (11 – 6 = 5)-го элемента,
- в 7-й элемент записывается значение (11 – 7 = 4)-го элемента,
- в 8-й элемент записывается значение (11 – 8 = 3)-го элемента,
- в 9-й элемент записывается значение (11 – 9 = 2)-го элемента,
- в 10-й элемент записывается значение (11 – 10 = 1)-го элемента.

Здесь всё обстоит гораздо проще — значения берутся из неизменяемой части массива, поэтому путаницы не возникнет. Получается массив:



Ответ: массив 5 4 3 4 5 5 4 3 4 5 (вариант ответа №1).

Задача 12. В программе описан одномерный целочисленный массив с индексами от 0 до 10. Ниже представлен записанный на разных языках программирования фрагмент одной и той же программы, обрабатывающей данный массив:

Бейсик	`s = 0`
	`n = 10`
	`FOR i = 0 TO n`
	`IF i = n - i THEN`
	`s = s + A(i) + A(i+1)`
	`END IF`
	`NEXT i`

Паскаль	`s := 0;`
	`n := 10;`
	`for i := 0 to n do begin`
	`if i = n - i then`
	`s := s + A[i] + A[i+1];`
	`end;`

Окончание табл.

Си	`s = 0;`
	`n = 10;`
	`for (i = 0; i <= n; i++)`
	`if (i == n-i) s = s + A[i] + A[i+1];`

Алгоритмический язык	`s := 0`
	`n := 10`
	`нц для i от 0 до n`
	`  если i = n - i`
	`  то s := s + A[i] + A[i+1]`
	`  все`
	`кц`

В начале выполнения этого фрагмента в массиве находились числа 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, т. е. A[0] = 0, A[1] = 1 и т. д. Чему будет равно значение переменной *s* после выполнения данной программы?

1) 45 2) 11 3) 5 4) 0

Решение

Из анализа текста программы видно, что речь идёт о вычислении суммы элементов массива по следующим правилам: к текущей сумме прибавляются i-й и $(i + 1)$-й элементы, но делается это только в том случае, когда значение i совпадает с $(n - i)$.

Для решения задачи можно начертить модель массива и честно просмотреть все значения i от 0 до n. Однако, поскольку n является константой (равно 10), нетрудно догадаться, что указанное условие будет за весь цикл выполнено только один раз. Соответствующее значение i можно найти из уравнения:

$i = n - i \Rightarrow i = 10 - i \Rightarrow 2i = 10 \Rightarrow i = 5$.

Тогда вычисленная сумма будет равна A[i] + A[i+1] = A[5] + A[6] = 5 + 6 = 11.

Ответ: $s = 11$ (вариант ответа №2).

Задачи для самостоятельного решения

1*. В программе описан одномерный целочисленный массив A с индексами от 0 до 10. Ниже представлен фрагмент этой программы, записанный на разных языках программирования, в котором значения элементов массива сначала задаются, а затем меняются.

Бейсик	Паскаль

`FOR i = 0 TO 10`
`A(i) = i - 1`
`NEXT i`
`FOR i = 10 TO 1 STEP -1`
`A(i-1) = A(i)`
`NEXT i`
	`for i := 0 to 10 do`
	`A[i] := i - 1;`
	`for i := 10 downto 1 do`
	`A[i-1] := A[i];`

Чему окажутся равны элементы этого массива?

1) 9 9 9 9 9 9 9 9 9 9
2) 0 1 2 3 4 5 6 7 8 9 9
3) 0 1 2 3 4 5 6 7 8 9 10
4) −1 −1 0 1 2 3 4 5 6 7 8

2. В программе описан одномерный целочисленный массив A с индексами от 0 до 10. Ниже представлен фрагмент этой программы, записанный на разных языках программирования, в котором значения элементов массива сначала задаются, а затем меняются.

Бейсик	Паскаль

FOR i = 0 TO 10
A(i) = i \ 2
NEXT i
FOR i = 1 TO 5 STEP 1
A(i) = A(2*i)
NEXT i
	for i := 0 to 10 do
	A[i] := i div 2;
	for i := 1 to 5 do
	A[i] := A[2*i];

Чему окажутся равны элементы этого массива?

1) 0 0 1 1 2 2 3 3 4 4 5
2) 0 1 2 3 4 5 3 3 4 4 5
3) 0 1 2 3 4 5 6 7 8 9 10
4) −5 −4 −3 −2 −1 0 1 2 3 4 5

3. В программе описан одномерный целочисленный массив A с индексами от 0 до 10. Ниже представлен фрагмент этой программы, записанный на разных языках программирования, в котором значения элементов массива сначала задаются, а затем меняются.

Бейсик	Паскаль

FOR i = 0 TO 10
A(i) = i - 5
NEXT i
FOR i = 1 TO 10 STEP 1
t = A(i)
A(i) = A(i-1)
A(i-1) = t
NEXT i
	for i := 0 to 10 do
	A[i] := i - 5;
	for i := 1 to 10 do begin
	t := A[i];
	A[i] := A[i-1];
	A[i-1] := t;
	end;

Чему окажутся равны элементы этого массива?

1) −5 −4 −3 −2 −1 0 1 2 3 4 5
2) −4 −5 −2 −3 0 −1 2 1 4 3 5
3) 5 4 3 2 1 0 −1 −2 −3 −4 −5
4) −4 −3 −2 −1 0 1 2 3 4 5 −5

4. Значения предварительно обнуленного двумерного массива размера 5 × 5 задаются с помощью вложенного оператора цикла в представленном фрагменте программы.

Бейсик	Паскаль

FOR n = 1 TO 5
FOR k = 1 TO 5
B(n,6-n) = n - k + 2
NEXT k
NEXT n
	for n := 1 to 5 do
	for k := 1 to 5 do
	B[n,6-n] := n - k + 2;

Сколько элементов массива будут иметь ненулевые значения?

1) 24 2) 20 3) 4 4) 1

5. Значения двумерного массива размера 5 × 5 задаются с помощью вложенного оператора цикла в представленном фрагменте программы

Бейсик	Паскаль

FOR n = 1 TO 5
FOR k = 1 TO 5
B(n,k) = 1
NEXT k
NEXT n
FOR n = 1 TO 5
FOR k = n TO 5
B(n,k) = B(n,k) + B(k,n)
NEXT k
NEXT n
	for n := 1 to 5 do
	for k := 1 to 5 do
	  B[n,k] := 1;
	for n := 1 to 5 do
	for k := n to 5 do
	  B[n,k] := B[n,k] + B[k,n];

Сколько элементов массива будут равны двум?

1) 10 2) 15 3) 25 4) 5

Ответы для самопроверки

Задача	Ответ
1	1
2	2
3	4
4	3
5	2

Программирование

A12 Операции с массивами: задачи группы «С»

Конспект

Обмен местами элементов массива

Производится аналогично обмену значений двух обычных переменных (обычно — при помощи дополнительной «буферной» переменной).

Обработка элементов массива (определение максимума/минимума, вычисление суммы, произведения, среднего и пр.)

В цикле (для многомерного массива — во вложенных циклах) производится перебор элементов массива (полный или частичный — для фрагмента массива).

- *При поиске максимума/минимума* — за предполагаемый максимум/минимум берётся первый элемент массива либо константа, заведомо меньшая/большая любого элемента. Далее каждый очередной элемент массива сравнивается с предполагаемым максимумом/минимумом, и если этот элемент больше/меньше предполагаемого максимума/минимума, то значение этого элемента запоминается в качестве нового предполагаемого максимума/минимума. (Дополнительно при этом в отдельной переменной (переменных) может перезапоминаться индекс (индексы) очередного предполагаемого максимума/минимума.)

Поиск максимума	Поиск минимума

`Max := Mas[1];`
`NoMax := 1;`
`for i := 2 to N do`
`  if Mas[i] > Max then begin`
`  Max := Mas[i];`
`  NoMax := i;`
`  end;`
`end;`
	`Min := Mas[1];`
	`NoMin := 1;`
	`for i := 2 to N do`
	`  if Mas[i] < Min then begin`
	`  Min := Mas[i];`
	`  NoMin := i;`
	`  end;`
	`end;`

- *При вычислении суммы/произведения* — вначале переменной, выделенной для накопления суммы/произведения присваивается инициализационное значение (нуль — для суммы, единица — для произведения). Затем в цикле (либо вложенных циклах) выполняется перебор элементов массива и текущее значение суммы/произведения складывается/умножается на текущий элемент массива.

Вычисление суммы	Вычисление произведения

`S := 0;`
`for i := 1 to N do`
`  S := S + Mas[i];`
	`P := 1;`
	`for i := 1 to N do`
	`  P := P * Mas[i];`

- *При вычислении среднего значения* — выполняется суммирование элементов массива, а затем полученная сумма делится на количество элементов в массиве.
```
S := 0;
for i := 1 to N do S := S + Mas[i];
Sredn := S / N;
```
- *При определении максимума/минимума, суммы, произведения, среднего значения элементов, удовлетворяющих заданному условию (например, только положительных)* — дополнительно добавляется условный оператор с соответствующим условием, и требуемое действие (проверка и переприсваивание предполагаемого максимума/минимума, сложение, умножение) выполняется только при истинности этого условия. При вычислении среднего значения также предусматривается отдельная переменная-счетчик, которая увеличивается на 1 каждый раз, когда к сумме добавляется очередной удовлетворяющий условию элемент массива, и после вычисления суммы она делится на значение этого счётчика (количество вошедших в сумму элементов).

Разбор типовых задач

Задача 1*. Опишите на русском языке или на одном из языков программирования алгоритм поиска второго по величине (т. е. следующего по величине за максимальным) элемента в числовом массиве из 30 различных элементов.

Решение

Наиболее простое решение — вначале найти «основной» максимум и запомнить его в отдельной переменной *Max*, а потом повторно провести поиск максимума, добавив в оператор `if` к условию «текущий элемент больше предполагаемого максимума» дополнительное условие «если текущий элемент не равен основному максимуму *Max*»:
```
if (Mas[i] > Max2) AND (Mas[i] <> Max) then Max2 := Mas[i];
```
Однако можно построить программу более оптимально, реализовав в ней только один цикл. При этом:

- объявляем две переменные: Max — для запоминания предполагаемого «основного» максимума, Max2 — для запоминания искомого второго максимума;
- изначальные значения Max и Max2 определяем следующим образом: сравниваем между собой 1-й и 2-й элементы массива и больший из них присваиваем переменной Max, а меньший — переменной Max2;
- просмотр массива производится с третьего элемента (считаем, то первые два мы уже «обработали»);
- если очередной элемент больше, чем Max, то этот элемент копируем в Max, а перед этим прежнее значение Max копируем в Max2; иначе дополнительно проверяем: если текущий элемент больше, чем Max2, то копируем его в Max2;
- очевидно, что после просмотра всего массива мы получаем в переменной Max «основной» максимум, а в переменной Max2 — искомый второй максимум.

Полный текст программы:
```
const N = 30;
var Mas: array[1..N] of integer;
Max, Max2, i: integer;
begin
  if Mas[1] > Mas[2] then begin
  // если первый элемент больше второго,
```

```
Max := Mas[1]; Max2 := Mas[2]; end
// то первый элемент — это «основной» максимум, а второй элемент —
// второй максимум,
else begin
  Max := Mas[2]; Max2 := Mas[1];
// иначе — наоборот
end;
for i := 3 to N do begin
  if Mas[i] > Max then begin
  // если очередной элемент больше предполагаемого «основного» максимума, то
Max2 := Max; Max := Mas[i]; end
// бывший «основной» максимум становится предполагаемым вторым максимумом,
// а текущий элемент — предполагаемым «основным» максимумом
else
  if Mas[i] > Max2 then Max2 := Mas[i];
// иначе если очередной элемент больше предполагаемого второго максимума,
// то этот элемент становится новым вторым максимумом
  end;
  writeln(Max2);
  // Max2 — это искомый второй максимум
end.
```

Пример для массива (3,2,5,6,8,4,7):

1) первоначально (Mas[1] = 3 больше, чем Mas[2] = 2):

1	2	3	4	5	6	7
3	2	4	6	5	1	7

Max	3
Max2	2

2) $i = 3$: $Mas[3] = 5$: больше, чем *Max*:

1	2	3	4	5	6	7
3	2	4	6	5	1	7

Max	4
Max2	3

3) $i = 4$: $Mas[4] = 6$: больше, чем *Max*:

1	2	3	4	5	6	7
3	2	4	6	5	1	7

Max	6
Max2	4

4) $i = 5$: $Mas[5] = 5$: меньше, чем *Max*, но больше, чем *Max2*:

1	2	3	4	5	6	7
3	2	4	6	5	1	7

Max	6
Max2	5

5) $i = 6$: $Mas[6] = 1$: меньше, чем Max, меньше, чем $Max2$:

1	2	3	4	5	6	7
3	2	4	6	5	1	7

Max	6
Max2	5

6) $i = 7$: $Mas[7] = 7$: больше, чем Max:

1	2	3	4	5	6	7
3	2	4	6	5	1	7

Max	7
Max2	6

Результат:

Max2	6

Задача 2*. Опишите на русском языке или одном из языков программирования алгоритм поиска номера первого из двух последовательных элементов в целочисленном массиве из 30 элементов, сумма которых максимальна (если таких пар несколько, то можно выбрать любую из них).

Решение

Алгоритм поиска максимального значения суммы пар элементов в массиве в целом подобен алгоритму поиска максимального элемента. Кроме того, нужно предусмотреть отдельную переменную *Num* для сохранения номера (индекса) первого элемента пары, сумма которой предполагается максимальной.

В целом алгоритм может быть построен следующим образом:

- изначально предполагается, что максимальной является сумма первого и второго элемента (соответственно, в переменной *Num* запоминаем значение 1);
- просмотр массива ведётся со второго элемента и до предпоследнего, так как при используемом способе выделения пар («текущий элемент плюс следующий за ним) последний элемент массива пары не имеет;
- если сумма текущего и последующего за ним элементов больше, чем предполагаемое значение максимальной суммы, то в *Num* запоминается номер текущего элемента, а эта сумма запоминается как новое значение предполагаемой максимальной суммы;
- по завершении просмотра массива в переменной *Num* будет содержаться искомый номер первого элемента пары с максимальной суммой.

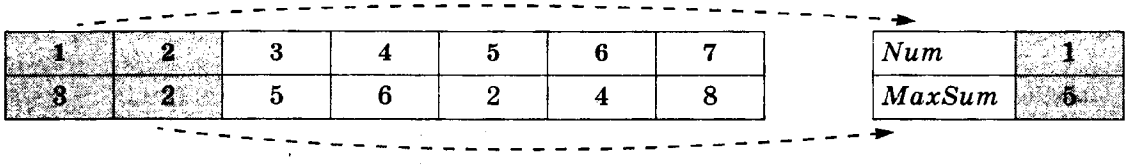
Полный текст программы:

```
const N = 30;
var Mas: array[1..N] of integer;
MaxSum, Num, i: integer;
begin
  MaxSum := Mas[1] + Mas[2];
  Num := 1;
    // предполагаем, что максимальна сумма 1-го и 2-го элемента
  for i := 2 to N - 1 do begin
  // просмотр массива со второго элемента до предпоследнего
    if Mas[i] + Mas[i+1] > MaxSum then begin
    // если сумма текущего элемента и следующего
    // за ним больше предполагаемой максимальной суммы, то
```

```
            Num := i;
          // запоминаем индекс текущего элемента (начала новой предполагаемой
          // пары с максимальной суммой)
            MaxSum := Mas[i] + Mas[i+1];
          // и записываем текущую сумму пары в качестве новой предполагаемой
          // максимальной суммы
          end;
      end;
      writeln(Num);
end.
```

Пример для массива (3,2,5,6,2,4,8):

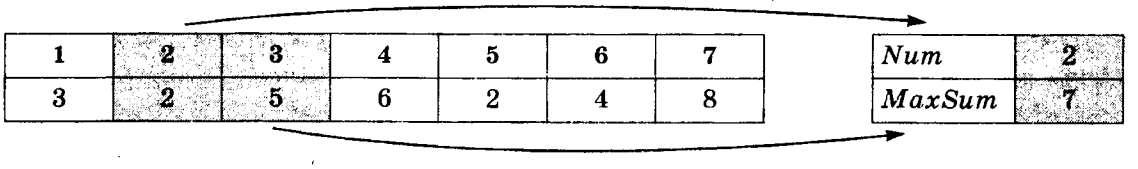
1) первоначально:



1	2	3	4	5	6	7
3	2	5	6	2	4	8

Num	1
MaxSum	5

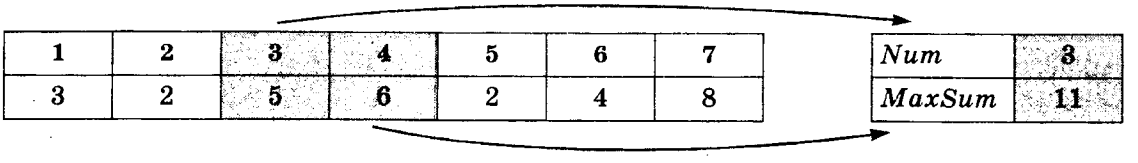
2) $i = 2$: $Mas[2] + Mas[3] = 2 + 5 = 7$: **больше, чем** *MaxSum*:



1	2	3	4	5	6	7
3	2	5	6	2	4	8

Num	2
MaxSum	7

3) $i = 3$: $Mas[3] + Mas[4] = 5 + 6 = 11$: **больше, чем** *MaxSum*:



1	2	3	4	5	6	7
3	2	5	6	2	4	8

Num	3
MaxSum	11

4) $i = 4$: $Mas[4] + Mas[5] = 6 + 2 = 8$: **меньше, чем** *MaxSum*:

1	2	3	4	5	6	7
3	2	5	6	2	4	8

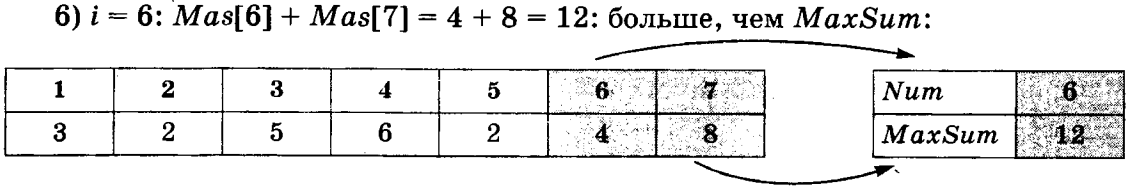
Num	3
MaxSum	11

5) $i = 5$: $Mas[5] + Mas[6] = 2 + 4 = 6$: **меньше, чем** *MaxSum*:

1	2	3	4	5	6	7
3	2	5	6	2	4	8

Num	3
MaxSum	11

6) $i = 6$: $Mas[6] + Mas[7] = 4 + 8 = 12$: **больше, чем** *MaxSum*:



1	2	3	4	5	6	7
3	2	5	6	2	4	8

Num	6
MaxSum	12

Результат:

Num	6

Задача 3*. Опишите на русском языке или одном из языков программирования алгоритм подсчёта максимального количества подряд идущих совпадающих элементов в целочисленном массиве длины 30.

Решение

Алгоритм подсчёта максимального количества подряд идущих совпадающих элементов может быть следующим:

- объявляются отдельные переменные: *Num* — для подсчёта текущего количества подряд идущих одинаковых элементов; *Max* — для подсчёта максимального значения *Num*;
- обеим этим переменным изначально присваивается минимально возможное значение 1 (любой элемент можно рассматривать как группу одинаковых элементов, состоящую всего из одного элемента);
- просматривается массив начиная со второго элемента;
- сравнивается текущий элемент с предыдущим: если они равны, то счётчик количества одинаковых чисел *Num* увеличивается на 1;
- иначе (понимая, что группа подряд идущих одинаковых элементов кончилась) — проверяется: если текущее значение *Num* больше предполагаемого *Max* (т. е. найдена группа подряд идущих одинаковых чисел ещё большей длины, чем ранее), то записывается *Num* как новое значение *Max*; после этого сбрасывается *Num* снова в 1 для будущего подсчёта длины уже новой последовательности одинаковых элементов;
- по завершении цикла ещё раз проверяется: если *Num* (т. е. длина последней цепочки одинаковых символов) больше прежнего *Max*, то записывается *Num* в качестве *Max*;
- в итоге переменная *Max* содержит искомое значение максимального количества подряд идущих совпадающих элементов.

Полный текст программы:

```
const N = 30;
var Mas: array[1..N] of integer;
    Max, Num, i: integer;
begin
  Max := 1; Num := 1;
   // изначально присваиваем обеим переменным минимально возможное
   // значение длины цепочки одинаковых символов
  for i := 2 to N do begin // просмотр массива со второго элемента
    if Mas[i] = Mas[i-1] then Num := Num + 1;
   // если очередной элемент равен предыдущему, то счетчик длины после
   // довательности таких одинаковых символов увеличиваем на 1
    else begin
       if Num > Max then Max := Num;
   // иначе (группа одинаковых элементов кончилась):
   // если полученная длина последовательности больше длины
   // ранее найденной такой цепочки, то эту полученную длину
   // запоминаем как новый максимум,
       Num := 1;
   // и после этого сбрасываем счетчик длин цепочек в 1 для будущего
   // отсчета длины новой цепочки одинаковых элементов
    end; // конец if
  end;    // конец цикла
  if Num > Max then Max := Num;
   // проверяем: если последнее найденное значение Num больше
   // чем ранее найденное максимальное значение длины цепочки
   // одинаковых символов, то это Num и записываем в качестве Max
  writeln(Max); // переменная Max содержит искомое значение
end.
```

Пример для массива (3,3,1,1,1,5,5):

1) первоначально:

1	2	3	4	5	6	7
3	3	1	1	1	5	5

Num	1
Max	1

2) $i = 2$: $Mas[2] = Mas[1]$:

1	2	3	4	5	6	7
3	3	1	1	1	5	5

Num	2
Max	1

3) $i = 3$: $Mas[3] \neq Mas[2]$, *Num* больше чем *Max*:

1	2	3	4	5	6	7
3	3	1	1	1	5	5

Num	1
Max	2

4) $i = 4$: $Mas[4] = Mas[3]$:

1	2	3	4	5	6	7
3	3	1	1	1	5	5

Num	2
Max	2

5) $i = 5$: $Mas[5] = Mas[4]$:

1	2	3	4	5	6	7
3	3	1	1	1	5	5

Num	3
Max	2

6) $i = 6$: $Mas[6] \neq Mas[5]$, *Num* больше чем *Max*:

1	2	3	4	5	6	7
3	3	1	1	1	5	5

Num	1
Max	3

7) $i = 7$: $Mas[7] = Mas[6]$:

1	2	3	4	5	6	7
3	3	1	1	1	5	5

Num	2
Max	3

7) финальная проверка: *Num* меньше чем *Max*:

1	2	3	4	5	6	7
3	3	1	1	1	5	5

Num	2
Max	3

Результат:

Max	3

Задача 4*. Опишите на русском языке или одном из языков программирования алгоритм получения из заданного целочисленного массива размером 30 элементов другого массива, который будет содержать модули значений элементов первого массива (не используя специальной функции, вычисляющей модуль числа).

Решение

По сравнению с предыдущими, данное задание является очень лёгким.

Для замены специальной функции, вычисляющей модуль числа, проще всего сравнивать каждый элемент исходного массива с нулём и для отрицательных чисел менять знак с помощью унарной операции «–».

Полный текст программы:

```
const N = 30;
var Mas1, Mas2: array[1..N] of integer;
    i: integer;
begin
  for i := 1 to N do
    // просматриваем весь исходный массив
    if Mas1[i] < 0 then Mas2[i] := -Mas1[i]
    // если очередной исходный элемент меньше нуля, то записываем
    // этот элемент во второй массив с обратным знаком
        else Mas2[i] := Mas1[i];
    // иначе записываем во второй массив сам этот элемент
end.
```

Пример для массива (1,−2,3,−4,5).

1) $i = 1$: $Mas1[1] = 1$ (> 0):

1	2	3	4	5
1	−2	3	−4	5

1	2	3	4	5
1				

2) $i = 2$: $Mas1[2] = -2$ (< 0):

1	2	3	4	5
1	−2	3	−4	5

−Mas[]

1	2	3	4	5
1	2			

3) $i = 3$: $Mas1[3] = 3$ (> 0):

1	2	3	4	5
1	−2	3	−4	5

1	2	3	4	5
1	2	3		

4) $i = 4$: $Mas1[4] = -4$ (< 0):

1	2	3	4	5
1	−2	3	−4	5

−Mas[]

1	2	3	4	5
1	2	3	4	

5) $i = 5$: $Mas1[5] = 5$ (> 0):

1	2	3	4	5
1	−2	3	−4	5

1	2	3	4	5
1	2	3	4	5

Задача 5*. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от 0 до 100 — баллы учащихся выпускного класса за итоговый тест по информатике. Для получения положительной оценки за тест требовалось набрать не менее 20 баллов. Опишите на русском языке или на одном из языков программирования алгоритм, который позволяет найти и вывести минимальный балл среди учащихся, получивших за тест положительную оценку. Известно, что в классе хотя бы один учащийся получил за тест положительную оценку.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из них.

Паскаль	Бейсик

const N = 30;
var
a: array [1..N] of integer;
i, j, min: integer;
begin
for i := 1 to N do readln(a[i]);
...
end.
	N = 30
	DIM A(N) AS INTEGER
	DIM I, J, MIN AS INTEGER
	FOR I = 1 TO N
	INPUT A(I)
	NEXT I
	...
	END

Си	**Естественный язык**

#include <stdio.h>
#define N 30
void main(void)
{
int a[N];
int i, j, min;
for (i = 0; i < N; i++)
scanf("%d", &a[i]);
}
	Объявляем целочисленные переменные I, J, MIN.
	В цикле от 1 до 30 вводим элементы массива A с 1-го по 30-й.
	...

В качестве ответа Вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например, Borland Pascal 7.0) или в виде блок-схемы. В этом случае вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

Решение

Данная задача подобна поиску минимума в массиве, — но с дополнительным условием: искать минимум надо только среди элементов, равных или превышающих 20.

Соответственно этому, в типовой алгоритм поиска минимума следует внести следующие изменения:

- **поскольку в качестве предполагаемого минимума нельзя первоначально использовать первый элемент (он может оказаться не удовлетворяющим условию «⩾20»), нужно в качестве первого значения предполагаемого минимума брать число, заведомо превышающее любое значение в массиве (в данном случае это может быть число 101);**
- **в оператор if, проверяющий, не является ли текущий элемент просматриваемого массива меньшим, чем предполагаемый минимум, надо добавить условие «текущий элемент больше или равен 20».**

Полный текст программы:

```
const N = 30;
var a: array [1..N] of integer;
    i, min: integer; // переменная j не используется
begin
  for i := 1 to N do readln(a[i]); // считали массив
  min := 101;
   // в качестве исходного предполагаемого минимума берется значение,
   // превышающее максимально возможное значение элемента в массиве
```

```
  for i := 1 to N do // полный просмотр массива
    if (a[i] >= 20) and (a[i] < min) then min := a[i];
    // если текущий элемент удовлетворяет условию (больше или равен 20)
    // и этот элемент меньше предполагаемого минимума, то этот элемент
    // запоминаем как новый предполагаемый минимум
writeln(min); // вывод найденного минимума на экран
end.
```

Пример для массива (80,18,43,5,50,28,30).

1) первоначально:

1	2	3	4	5	6	7
80	18	43	5	50	28	30

min	101

2) $i = 1$: $Mas[1] = 80$ — больше 20 и меньше, чем *min*:

1	2	3	4	5	6	7
80	18	43	5	50	28	30

min	80

3) $i = 2$: $Mas[2] = 18$ — меньше 20:

1	2	3	4	5	6	7
80	18	43	5	50	28	30

min	80

4) $i = 3$: $Mas[3] = 43$ — больше 20 и меньше, чем *min*:

1	2	3	4	5	6	7
80	18	43	5	50	28	30

min	43

5) $i = 4$: $Mas[4] = 5$ — меньше 20:

1	2	3	4	5	6	7
80	18	43	5	50	28	30

min	80

6) $i = 5$: $Mas[5] = 50$ — больше 20, но больше, чем *min*:

1	2	3	4	5	6	7
80	18	43	5	50	28	30

min	80

7) $i = 6$: $Mas[6] = 28$ — больше 20 и меньше, чем *min*:

1	2	3	4	5	6	7
80	18	43	5	50	28	30

min	28

8) $i = 7$: $Mas[7] = 30$ — больше 20, но больше, чем *min*:

1	2	3	4	5	6	7
80	18	43	5	50	28	30

min	28

Результат:

min	28

Задача 6*. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать значения от 0 до 1000. Опишите на русском языке или на одном из языков программирования алгоритм, который позволяет подсчитать и вывести среднее арифметическое элементов массива, имеющих нечётное значение. Гарантируется, что в исходном массиве хотя бы один элемент имеет нечётное значение.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из них.

Паскаль

```
const N = 30;
var
a: array [1..N] of integer;
i, x, y: integer;
s: real;
begin
for i := 1 to N do readln(a[i]);
...
end.
```

Бейсик

```
N = 30
DIM A(N) AS INTEGER
DIM I, X, Y AS INTEGER
DIM S AS SINGLE
FOR I = 1 TO N
INPUT A(I)
NEXT I
...
END
```

Си

```
#include <stdio.h>
#define N 30
void main(void)
{int a[N];
int i, x, y;
float s;
for (i = 0; i < N; i++)
scanf("%d", &a[i]);
...}
```

Естественный язык

```
Объявляем массив A из 30 элементов.
Объявляем целочисленные переменные
I, X, Y.
Объявляем вещественную переменную S.
В цикле от 1 до 30 вводим элементы
массива A с 1-го по 30-й.
...
```

В качестве ответа Вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например, Borland Pascal 7.0) или в виде блок-схемы. В этом случае вы должны использовать переменные, аналогичные переменным, используемым в алгоритме, записанном на естественном языке, с учетом синтаксиса и особенностей используемого вами языка программирования.

Решение

Речь идёт о вычислении среднего арифметического элементов массива, удовлетворяющих заданному условию (здесь: нечётность). Для этого нужно в цикле просмотра массива подсчитывать сумму только элементов, удовлетворяющих этому условию, и одновременно подсчитывать количество таких элементов.

Полный текст программы:

```
const N = 30;
var a: array [1..N] of integer;
    i, x, y: integer;
    s: real;
begin
  for i := 1 to N do readln(a[i]); // считали массив
  x := 0; y := 0;
   // x – переменная для подсчета суммы элементов, удовлетворяющих
   // заданному условию;
```

```
    // y – переменная для подсчета количества этих элементов;
    // изначально обе переменные обнуляются
    for i:=1 to N do // полный просмотр массива
      if (a[i] mod 2 = 1) then begin
    // если текущий элемент удовлетворяет условию (нечетный), то
        x := x + a[i];
    // прибавляем этот элемент к накапливаемой сумме
        y := y + 1;
    // и увеличиваем количество просуммированных элементов на 1
  end;
  s := x / y;
    // по окончании цикла вычисляем среднее арифметическое,
    // деля полученную сумму нечетных чисел на их количество
    writeln(s); // вывод среднего арифметического значения
    // нечетных элементов

end.
```

Пример для массива (4,65,2,3,1).

1) первоначально:

1	2	3	4	5
4	65	2	3	1

x	0
y	0

2) $i = 1$: Mas[1] = 4 — чётное:

1	2	3	4	5
4	65	2	3	1

x	0
y	0

3) $i = 2$: Mas[2] = 65 — нечётное:

1	2	3	4	5
4	65	2	3	1

x	65
y	1

4) $i = 3$: Mas[3] = 2 — чётное:

1	2	3	4	5
4	65	2	3	1

x	65
y	1

5) $i = 4$: Mas[4] = 3 — нечётное:

1	2	3	4	5
4	65	2	3	1

x	65
y	2

6) $i = 5$: Mas[5] = 1 — нечётное:

1	2	3	4	5
4	65	2	3	1

x	69
	3

Результат:

x	69
y	3

$\Rightarrow$ s = 69 / 3 = 23

Задача 7*. Дан целочисленный массив из 20 элементов. Элементы массива могут принимать целые значения от 0 до 1000. Опишите на русском языке или на одном из языков программирования алгоритм, позволяющий найти и вывести минимальное значение среди элементов массива, которые имеют чётное значение и не делятся на три. Гарантируется, что в исходном массиве есть хотя бы один элемент, значение которого чётно и не кратно трём.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но использовать все описанные переменные не обязательно.

Паскаль

```
const N = 20;
var
a: array [1..N] of integer;
i, j, min: integer;
begin
for i := 1 to N do
readln(a[i]);
...
end.
```

Алгоритмический язык

```
алг
нач
цел N = 20
целтаб a[1:N]
цел i, j, MIN
нц для i от 1 до N
ввод a[i]
кц
...
кон
```

Бейсик

```
N = 20
DIM A(N) AS INTEGER
DIM I, J, MIN AS INTEGER
FOR I = 1 TO N
INPUT A(I)
NEXT I
...
END
```

Си

```
#include <stdio.h>
#define N 20
void main(void){
int a[N];
int i, j, min;
for (i = 0; i < N; i++)
scanf("%d", &a[i]);
...
}
```

Естественный язык

```
Объявляем массив A из 20 элементов.
Объявляем целочисленные переменные I, J, MIN.
В цикле от 1 до 20 вводим элементы массива A с 1-го по 20-й.
...
```

В качестве ответа вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например Borland Pascal 7.0) или в виде блок-схемы. В этом случае вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

Решение

Данная задача по смыслу решения полностью аналогична задаче 8: здесь также нужно искать минимум среди элементов массива, удовлетворяющих заданному условию (в данном случае — чётность и неделимость нацело на 3).

Изменения, которые нужно внести в типовой алгоритм поиска минимума в массиве:

- поскольку в качестве предполагаемого минимума нельзя первоначально использовать первый элемент (он может оказаться не удовлетворяющим заданному условию), нужно в качестве первого значения предполагаемого минимума брать число, заведомо превышающее любое значение в массиве (в данном случае это может быть число 1001);
- в оператор if, проверяющий, не является ли текущий элемент просматриваемого массива меньшим, чем предполагаемый минимум, надо добавить условия «текущий элемент чётный» и «текущий элемент не делится нацело на 3».

Полный текст программы:

```
const N = 20;
var
  a: array [1..N] of integer;
  i, min: integer; // переменная j не используется
begin
  for i := 1 to N do readln(a[i]); // считали массив
min := 1001;
    // в качестве исходного предполагаемого минимума берется значение,
    // превышающее максимально возможное значение элемента в массиве
    for i := 1 to N do // полный просмотр массива
if (a[i] mod 2 = 0) and (a[i] mod 3 <> 0) and (a[i] < min) then min := a[i];
    // если текущий элемент удовлетворяет условиям: он четный И он
    // не делится без остатка на 3 И этот элемент меньше предполагаемого
    // минимума, то этот элемент запоминаем как новый предполагаемый минимум
writeln(min); // вывод найденного минимума на экран
end.
```

Пример для массива (333,80,43,60,44,3,68).

1) первоначально:

1	2	3	4	5	6	7
333	80	43	60	44	3	68

min	1001

2) $i = 1$: $Mas[1] = 333$ — нечётное:

1	2	3	4	5	6	7
333	80	43	60	44	3	68

min	1001

3) $i = 2$: $Mas[2] = 80$ — чётное, не делится на 3, меньше, чем *min*:

1	2	3	4	5	6	7
333	80	43	60	44	3	68

min	80

4) $i = 3$: $Mas[3] = 43$ — нечётное:

1	2	3	4	5	6	7
333	80	43	60	44	3	68

min	80

5) $i = 4$: $Mas[4] = 60$ — чётное, но делится на 3:

1	2	3	4	5	6	7
333	80	43	60	44	3	68

min	80

6) $i = 5$: $Mas[5] = 44$ — чётное, не делится на 3, меньше, чем *min*:

1	2	3	4	5	6	7
333	80	43	60	44	3	68

min	44

7) $i = 6$: $Mas[6] = 3$ — нечётное:

1	2	3	4	5	6	7
333	80	43	60	44	3	68

min	44

8) $i = 7$: $Mas[7] = 68$ — чётное, не делится на 3, но больше, чем *min*:

1	2	3	4	5	6	7
333	80	43	60	44	3	68

min	44

Результат:

min	44

Задачи для самостоятельного решения

1. Опишите на русском языке или одном из языков программирования алгоритм поиска номера третьего по счёту минимального элемента в целочисленном массиве длины 30. (В массиве не обязательно есть не менее трёх элементов, равных минимальному, поэтому предусмотрите соответствующую проверку.)
2. Опишите на русском языке или одном из языков программирования алгоритм вычисления максимального произведения трёх соседних элементов в целочисленном массиве длины 30. (Значение элементов в массиве по модулю не превышают 50.)
3. Дан целочисленный массив из 20 элементов. Элементы массива могут принимать целые значения, по модулю не превышающие 100. Опишите на русском языке или на одном из языков программирования алгоритм, позволяющий найти и вывести номер максимального значения среди неотрицательных элементов массива, не кратных 5. (Гарантируется, что в исходном массиве есть хотя бы один такой элемент.)
4. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения, по модулю не превышающие 50. Опишите на русском языке или на одном из языков программирования алгоритм, позволяющий вычислить среднее квадратичное значение чётных положительных элементов массива.
   Для справки: среднее квадратичное значение равно корню квадратному из среднего арифметического квадратов чисел. В языке Паскаль для возведения в квадрат используется стандартная функция `Sqr()`, а для вычисления квадратного корня — стандартная функция `Sqrt()`.
5. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения, по модулю не превышающие 50. Опишите на русском языке или на одном из языков программирования алгоритм, позволяющий на основе этого массива записать другой массив из 15 элементов, каждый элемент которого равен сумме квадратов пар элементов первого массива (первого и второго, третьего и четвертого и т.д.).
   Для справки: в языке Паскаль для возведения в квадрат используется стандартная функция `Sqr()`.

Ответы для самопроверки

Задача 1.

```
program mas_3_min;
const n = 30;
var mas: array[1..n] of integer;
    i: integer;
    min, nmin, nomin: integer;
    flag: integer;
begin
  for i := 1 to n do read(mas[i]);
    min := mas[1];
    flag := 1;
```

```
    for i := 2 to n do
    begin
      if mas[i] = min then
      begin
        flag := flag + 1;
        if flag = 3 then
        begin
          nmin := mas[i];
          nomin := i;
        end;
      end;
      if mas[i] < min then
      begin
        flag := 1;
        min := mas[i];
      end;
    end;
    if flag >= 3 then writeln('Номер 3-го минимального элемента:
                  ',nomin:2,' ; сам минимальный элемент равен ',nmin:2)
      else writeln('Найдено только ',flag:1,' минимальных элементов');
end.
```

Первоначально в качестве предполагаемого минимума берётся первый элемент массива, а переменная — счётчик *flag* приравнивается единице (предполагается, что первый минимум уже найден).

В ходе просмотра массива в цикле:

- если очередной элемент равен предполагаемому минимуму, то:
- счётчик увеличивается на 1;
- проверяется: если счётчик равен 3, значит, третий по счёту минимальный элемент найден, запоминается его номер и значение;
- иначе, если очередной элемент меньше предполагаемого минимума, то запоминается этот элемент в качестве нового предполагаемого минимума, а счётчик вновь приравнивается единице;
- завершив цикл, проверяется: если значение счётчика не меньше 3, значит, в массиве действительно имелось 3 или более элементов, равных минимальному, причём номер и значение этого третьего по счёту минимального элемента запомнены в переменных *nomin* и *nmin*; иначе выводится сообщение, что в массиве найдено меньшее количество элементов, равных минимальному.

Задача 2.

```
const N = 30;
var Mas: array[1..N] of integer;
MaxSum, Num, i: integer;
Begin
  for i := 1 to N read(Mas[i]);

  MinP := Mas[1] * Mas[2] * Mas[3];
  for i := 2 to N - 2 do begin
    if Mas[i] * Mas[i+1] * Mas[i+2] < MinP then
                          MinP := Mas[i] * Mas[i+1] * Mas[i+2];
  writeln(MinP);
end.
```

Алгоритм поиска минимального значения произведения триад элементов в массиве в целом подобен алгоритму поиска минимального элемента:

- изначально предполагается, что минимальным является произведение первого, второго и третьего элементов;
- просмотр массива ведётся со второго элемента и до третьего с конца включительно, так как при используемом способе выделения пар последний и предпоследний элементы массива триад не имеют;
- если произведение текущего и двух последующих за ним элементов меньше, чем предполагаемое значение минимального произведения, то это произведение запоминается как новое значение предполагаемого минимального произведения;
- по завершении просмотра массива в переменной *MinP* будет содержаться искомое значение минимального произведения триады элементов.

Задача 3.

```
const N = 20;
var
  a: array [1..N] of integer;
  i, max, nomax: integer;
begin
  for i := 1 to N do readln(a[i]);

max := -101;
    // в качестве исходного предполагаемого максимума берется значение,
    // заведомо меньшее минимально возможного значения элементов в массиве
for i := 1 to N do
  if (a[i] >= 0) and (a[i] mod 5 <> 0) and (a[i] > max) then begin
     max := a[i];
     nomax := i;
  end;
writeln(nomax);
end.
```

В данной задаче нужно искать максимум среди элементов массива, удовлетворяющих условию: неотрицательность и некратность 5.

Для этого используется типовой алгоритм поиска максимума в массиве, в который внесены следующие изменения:

- поскольку в качестве предполагаемого максимума нельзя первоначально использовать первый элемент (он может оказаться не удовлетворяющим заданному условию), нужно в качестве первого значения предполагаемого максимума брать число, заведомо меньшее любого значения в массиве (в данном случае это может быть число −101);
- в оператор if, проверяющий, не является ли текущий элемент просматриваемого массива большим, чем предполагаемый максимум, надо добавить условия «текущий элемент больше или равен нулю» и «текущий элемент не делится нацело на 5».

Задача 4.

```
const N = 30;
var a: array [1..N] of integer;
    i, sum, kol: integer;
    sredkv: real;
begin
  for i := 1 to N do readln(a[i]);
```

```
  sum := 0;
  kol := 0;
  for i := 1 to N do
    if (a[i] mod 2 = 0) AND (a[i] > 0) then begin
     sum := sum + Sqr(a[i]);
     kol := kol + 1;
  end;
  sredkv := Sqrt(sum/kol);
  writeln(sredkv);
end.
```

Программа вычисляет корень квадратный из среднего арифметического квадратов элементов массива, удовлетворяющих заданному условию (здесь: положительность и чётность).

Для этого в цикле просмотра массива подсчитывается сумма только возведенных в квадрат элементов, удовлетворяющих этому условию, и одновременно подсчитывается количество таких элементов. По завершении цикла вычисляется квадратный корень из частного от деления подсчитанной суммы на подсчитанное количество просуммированных квадратов элементов.

Задача 5.

```
const N1 = 30;
      N2 = 15;
var Mas1: array[1..N1] of integer;
    Mas2: array[1..N2] of integer;
    i: integer;
begin
  for i := 1 to N2 do
    Mas2[i] := Sqr(Mas1[2*i-1]) + Sqr(Mas1[2*i]);
end.
```

Основная сложность в данной программе — такая реализация цикла и такое вычисление индексов элементов массивов, чтобы перебирать и пары исходного массива, и элементы массива, в который нужно производить запись.

Составляется (фрагментарно) таблица соответствия индексов обрабатываемых элементов обоих массивов:

Индексы элементов первого массива	1 и 2	3 и 4	5 и 6	...	25 и 26	27 и 28	29 и 30
Индекс элемента второго массива	1	2	3	...	13	14	15

Нетрудно заметить, что если переменная i в цикле будет менять своё значение от 1 до 15, то это и будет нужный индекс второго массива. Но удвоенное значение этой переменной совпадает со вторым индексом обрабатываемой пары элементов в исходном массиве; при этом индекс первого элемента этой пары вычисляется вычитанием единицы из индекса второго элемента пары.

Тогда для вычисления индексов первого и второго элементов обрабатываемой пары в исходном массиве можно использовать формулы: $(2 \cdot i - 1)$ и $(2 \cdot i)$.

Программирование

C2 Процедуры и функции

Конспект

Подпрограмма

Подпрограмма – это поименованная или каким-либо иным образом обозначенная часть программы, которая может быть многократно вызвана из разных частей основной программы для выполнения неких «типовых» вычислений, в том числе для различных исходных данных.

Идея здесь состоит в следующем. Предположим, что некоторая часть алгоритма по своей сути повторяется в нашей программе несколько раз, — скажем, в комбинаторике при вычислении количества *сочетаний* из n элементов по k (подмножеств из k элементов, взятых произвольно из множества n элементов и различающихся хотя бы одним элементом без учёта порядка их следования) используется формула:

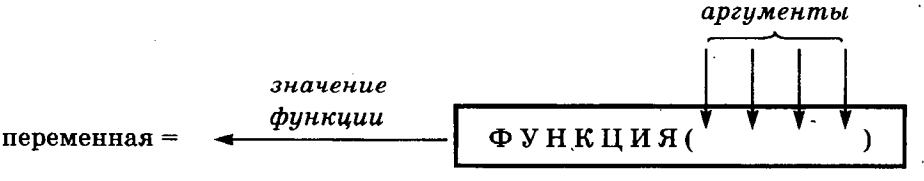
$$C_n^k = \frac{n!}{k!(n-k)!}$$

в которой трижды надо вычислять факториал для трёх разных исходных значений. Тогда вычисление факториала в виде выполняемых в цикле умножений можно оформить в виде подпрограммы, а затем просто обращаться к ней, каждый раз передавая этой подпрограмме требуемое исходное число.

Наконец, деление исходного сложного алгоритма на подпрограммы позволяет обеспечить его структуризацию, облегчить понимание и разработку программного продукта, разделить исходную задачу на более простые подзадачи, реализуя тем самым принцип нисходящего программирования («от общего к частному»).

Функция

Это одна из двух возможных разновидностей подпрограмм. Её отличительная особенность состоит в том, что функция, принимая на вход любое количество исходных данных (аргументов), может возвращать только одно значение, которое передаётся как значение самой этой функции.



Для работы с подпрограммами-функциями в языке Паскаль (и аналогично — в большинстве других современных процедурных языков высокого уровня) требуется знание следующих особенностей:

1. Подпрограмма-функция должна быть *описана* в начале программы — после объявления используемых в ней *глобальных* констант, меток и переменных, но до собственно текста основной программы, заключённого в операторные скобки BEGIN и END (для наглядности эти слова, обрамляющие текст основной программы, записаны прописными буквами).

2. Описание подпрограммы-функции начинается со строки

```
Function <имя функции>(<арг1>, <арг2>, ... : <тип1>; <арг3>, <арг4>, ... :
<тип2>; ...) : <тип результата>;
```

Здесь:

— `Function` — зарезервированное слово, указывающее транслятору, что далее идёт описание подпрограммы-функции;

— `<имя функции>` — идентификатор, выбираемый по тем же правилам, что и для имён переменных;

— сразу после имени функции в скобках записывается перечень передаваемых в эту функцию *формальных параметров* — имён переменных, которые далее будут использоваться при вычислениях, выполняемых в этой функции. При этом формальные параметры группируются (через запятую) в блоки одинаковых типов, а обозначения их типов записываются после списка параметров через двоеточие, как принято в Паскале при определении переменных; блоки параметров различного типа записываются через точку с запятой;

— после закрывающей скобки, также через двоеточие, записывается тип результата, который будет возвращать функция.

Пример (из листинга, приведённого в условии разбираемой нами задачи):

`Function F(x : real) : real;` — объявляется подпрограмма-функция с именем *F*, которая принимает на вход одно действительное (тип `real`) число и возвращает также действительное (`real`, записанное после скобок) значение результата.

3. После строки определения функции записывается её программный код. Правила его записи аналогичны правилам записи основной программы:

— сначала может присутствовать раздел описания *локальных* констант и переменных; эти константы и переменные «действуют» только в пределах данной функции и могут иметь такие же имена, что и локальные переменные, объявленные в других подпрограммах (транслятор их не путает);

— далее между операторными скобками `begin` и `end` записывается текст подпрограммы.

4. В конце текста подпрограммы нужно обязательно присвоить значение, которое функция должна возвращать в качестве результата, специальной переменной, имя которой совпадает с именем самой функции (причём отдельно объявлять эту переменную *не нужно*).

5. В тексте основной программы вызов функции производится следующим образом:

— записывается имя функции, а после него в скобках через запятую — список *фактических параметров* — константы, имена переменных, выражения, а также другие вызовы функций, значения которых нужно передать в функцию для выполнения вычислений. При этом количество и типы таких фактических параметров должны соответствовать указанным в объявлении функции формальным параметрам. Каждый фактический параметр при вызове функции записывается в соответствующий ему по порядку следования в списке формальный параметр и тем самым попадает в выполняемые функцией вычисления. При этом, если для формального параметра указан соответствующий составной тип, объявленный в разделе `type` основной программы, то в подпрограмму может быть передан, например, массив целиком;

— поскольку функция возвращает значение, её запись (вместе с фактическими параметрами в скобках) должна быть либо включена в состав выражения, либо записана после имени переменной и следующего за ней оператора присваивания, либо указана как параметр в процедуре вывода на экран, записи в файл и пр. Иначе возвращенное функцией значение «потеряется».

Разбор типовых задач

Задача 1. Определить, какое число будет напечатано в результате работы следующей программы:

```
Program A14;
Uses crt;
Var d, a, b, t, M, R: real;
Function F(x : real) : real;
 begin
  F := (x + 2) * (4 - x);
 end;

BEGIN
 a := -2; b := 4;
 d := 0.1;
 t := a; M := a; R := F(a);
 while t <= b do
  begin
   if (F(t) > R) then
    begin
     M := t;
     R := F(t);
    end;
   t := t + d;
  end;
 write(M);
END.
```

Решение

Анализируется текст программы. В ней (если не считать подключения модуля `crt` в разделе `uses`) вначале объявлен целый ряд переменных (раздел `var`).

Затем следует описание подпрограммы-функции с именем `F`, которой должно передаваться одно исходное значение — действительное число (формальный параметр `x`, который затем используется в качестве переменной в тексте подпрограммы) и которая должна возвращать действительное значение. Что именно вычисляется в этой подпрограмме, пока можно подробно не рассматривать.

После подпрограммы записан текст основной программы (между `BEGIN` и `END`).

Чтобы решить эту задачу, как говорится, «с нуля», нужно произвести полную трассировку используемых в программе переменных. Для этого составляется таблица трассировки. Важно! Каждый раз, когда встречается в основной программе вызов функции, перейти к тексту её описания и выполнить соответствующие команды до завершающего текст подпрограммы-функции слова `end`. При этом заданные в вызове функции фактические параметры автоматически переписываются в таблице в качестве значений соответствующих им формальных параметров (столбец формального параметра, равно как и команды подпрограммы-функции, в таблице выделены серым фоном ячеек). Кроме того, для наглядности значения переменных, изменяемых в результате действия текущей команды, выделены жирным шрифтом.

Оператор	*d*	*a*	*b*	*t*	*M*	*R*	*x*	F()	Примечание
`a := -2; b := 4; d := 0.1;`	**0.1**	**−2**	**4**						
`t := a; M := a;`	0.1	−2	4	**−2**	**−2**				

Продолжение табл.

Оператор	*d*	*a*	*b*	*t*	*M*	*R*	*x*	F()	Примечание
`R := F(a);`	0.1	−2	4	−2	−2		−2		Вызов функции
`F := (x + 2) * (4 - x);`	0.1	−2	4	−2	−2		−2	0	
`R := F(a);`	0.1	−2	4	−2	−2	0	−2		Значение F (результат выполнения функции) при выходе из функции в основную программу записывается в переменную R
`while t <= b do`	0.1	−2	4	-2	−2	0	−2		Условие цикла истинно — цикл выполняется
			$t \leqslant b$						
`if (F(t) > R) then`	0.1	−2	4	−2	−2	0	−2		Вызов функции
`F := (x + 2) * (4 - x);`	0.1	−2	4	−2	−2	0	−2	0	
`if (F(t) > R) then`	0.1	−2	4	−2	−2	0	−2	0	Условие не выполнено — ветвь then пропускается
						$F() = R$			
`t := t + d;`	0.1	−2	4	−1.9	−2	0	−2		
`while t <= b do`	0.1	−2	4	−1.9	−2	0	−2		Условие цикла истинно — цикл выполняется
			$t \leqslant b$						
`if (F(t) > R) then`	0.1	−2	4	−1.9	-2	0	−1.9		Вызов функции
`F := (x + 2) * (4 - x);`	0.1	−2	4	−1.9	−2	0	−1.9	0.59	
`if (F(t) > R) then`	0.1	−2	4	−1.9	−2	0	−1.9	0.59	Условие выполнено — выполняется ветвь then
						$F() > R$			
`M := t;`	0.1	−2	4	−1.9	−1.9	0	−1.9		
`R := F(t);`	0.1	−2	4	−1.9	−1.9	0	−1.9		Вызов функции
`F := (x + 2) * (4 - x);`	0.1	−2	4	−1.9	−1.9	0	−1.9	0.59	
`R := F(t);`	0.1	−2	4	−1.9	−1.9	0.59	−1.9	0.59	
`t := t + d;`	0.1	−2	4	−1.8	−1.9	0.59	−1.9		
`while t <= b do`	0.1	−2	4	−1.8	−1.9	0.59	−1.9		Условие цикла истинно — цикл выполняется
			$t \leqslant b$						
`if (F(t) > R) then`	0.1	−2	4	−1.8	−1.9	0.59	−1.8		Вызов функции
`F := (x + 2) * (4 - x);`	0.1	−2	4	−1.8	−1.9	0.59	−1.8	1.16	
`if (F(t) > R) then`	0.1	−2	4	−1.8	−1.9	0.59	−1.8	1.16	Условие выполнено — выполняется ветвь then
						$F() > R$			
`M := t;`	0.1	−2	4	−1.8	−1.8	0.59	−1.8		
`R := F(t);`	0.1	−2	4	−1.8	−1.8	0.59	−1.8		Вызов функции
`F := (x + 2) * (4 - x);`	0.1	−2	4	−1.8	−1.8	0.59	−1.8	1.16	
`R := F(t);`	0.1	−2	4	−1.8	−1.8	1.16	−1.8	1.16	
`t := t + d;`	0.1	−2	4	−1.7	−1.8	1.16	−1.8		
`while t <= b do`	0.1	−2	4	−1.7	−1.8	1.16	−1.8		Условие цикла истинно — цикл выполняется
			$t \leqslant b$						
`if (F(t) > R) then`	0.1	−2	4	−1.7	−1.8	1.16	−1.7		Вызов функции
`F := (x + 2) * (4 - x);`	0.1	−2	4	−1.7	−1.8	1.16	−1.7	1.71	
`if (F(t) > R) then`	0.1	−2	4	−1.7	−1.9	1.16	−1.7	1.71	Условие выполнено — выполняется ветвь then
						$F() > R$			

Окончание табл.

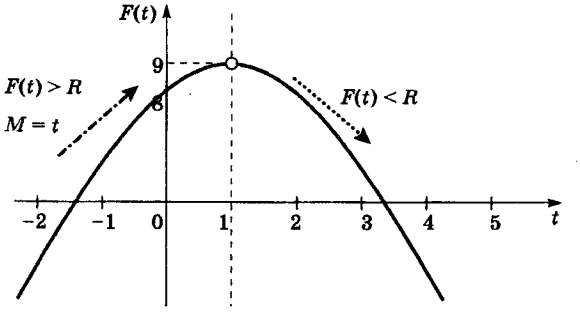
Оператор	*d*	*a*	*b*	*t*	*M*	*R*	*x*	F()	Примечание
`M := t;`	0.1	−2	4	−1.7	−1.7	1.16	−1.7		
`R := F(t);`	0.1	−2	4	−1.7	−1.7	1.16	−1.7		Вызов функции
`F := (x + 2) * (4 - x);`	0.1	−2	4	−1.7	−1.7	1.16	−1.7	1.71	
`R := F(t);`	0.1	−2	4	−1.7	−1.7	1.71	−1.7	1.71	
`t := t + d;`	0.1	−2	4	−1.6	−1.7	1.71	−1.7		

и т. д.

Конечно, выполнение полной трассировки такой программы — дело сложное и долгое. Однако уже сейчас можно увидеть следующую закономерность: пока что с увеличением значения t получаемое значение $F(t)$ также увеличивается, в результате чего это значение каждый раз оказывается больше предыдущего значения $F(t)$, запомненного в переменной R, и тогда текущее значение t заносится в интересующую переменную M.

До каких пор это будет продолжаться? Если выполнить умножение скобок в выражении, записанном в составе функции F, то получится квадратное уравнение: $(x + 2) \cdot (4 - x) = 4x + 8 - x^2 - 2x = -x^2 + 2x + 8$.

Поскольку перед x^2 в нём стоит знак минуса, ветви параболы, соответствующей этому уравнению, будут направлены вниз; мы же в своих вычислениях «движемся» по левой ветви этой параболы снизу вверх. Очевидно, что в процессе такого движения мы рано или поздно доберёмся до вершины параболы, после чего с ростом t начнётся движение по её второй ветви уже сверху вниз. Тогда условие $F(t) > R$ перестанет выполняться (причём это произойдёт, когда будет пройдена вершина параболы и произойдёт переход к следующему же после этого значению t), а значит, прекратится и запись текущих значений t в переменную M. И так будет уже до самого конца цикла изменения t.



Остаётся найти координату вершины параболы t и проследить работу программы вблизи этого значения данной переменной. Для этого можно использовать тот факт, что в вершине параболы значение производной равно нулю: $-2x + 2 = 0$. Тогда $x = 1$. Значит, искомое значение переменной t равно 1. Трассировка продолжается с него.

Оператор	*d*	*a*	*b*	*t*	*M*	*R*	*x*	F()	Примечание
`while t <= b do`	0.1	−2	4 ($t \leqslant b$)	1		8.99			Условие цикла истинно — цикл выполняется; значения переменных M и x мы пока не знаем; значение R равно $F()$ от предыдущего значения t (0,9)
`if (F(t) > R) then`	0.1	−2	4	1		8.99	1		Вызов функции
`F := (x + 2) * (4 - x);`	0.1	−2	4	1		8.99	1	9	
`if (F(t) > R) then`	0.1	−2	4	1		8.99 ($F() > R$)	1	9	Условие выполнено — выполняется ветвь `then`
`M := t;`	0.1	−2	4	1	1	8.99	1		

Окончание табл.

Оператор	*d*	*a*	*b*	*t*	*M*	*R*	*x*	F()	Примечание
F := (x + 2) * (4 - x);	0.1	-2	4	1	1	8.99	1	9	
R := F(t);	0.1	-2	4	1	1	8.99	1		Вызов функции
F := (x + 2) * (4 - x);	0.1	-2	4	1	1	8.99	1	9	
R := F(t);	0.1	-2	4	1	1	9	1	9	
t := t + d;	0.1	-2	4	1.1	1	9	1		
while t <= b do	0.1	-2	4 ($t \leqslant b$)	1.1	1	9	1		Условие цикла истинно — цикл выполняется
if (F(t) > R) then	0.1	-2	4	1.1	1	9	1.1		Вызов функции
F := (x + 2) * (4 - x);	0.1	-2	4	1.1	1	9	1.1	8.99	
if (F(t) > R) then ($F() < R$)	0.1	-2	4	1.1	1	9	1	8.99	Условие **не выполнено** — ветвь then пропускается. **И так будет до конца цикла изменения *t***

Итак, последнее возможное изменение значения переменной *M* уже произошло, и при этом *M* стала равна 1.

Ответ: в результате работы данной программы будет выведено число 1 (вариант ответа №1).

Задача 2*. Определите, какое число будет напечатано в результате работы следующей программы (для Вашего удобства программа представлена на четырёх языках):

Бейсик

```
Module A14
 Sub Main()
 Dim d, a, b, t, M, R As Double
      a = -3: b = 3
      d = 0.1
      t = a: M = a: R = F(a)
      While t < b
       If F(t) < R Then
          M = t
          R = F(t)
          End If
        t = t + d
       End While
       Console.Write(M)
      End Sub

Function F(ByVal x As Double) As Double
      Return (x - 1) * (x - 3)
End Function

End Module
```

Паскаль

```
Program A14;
Uses crt;
Var d, a, b, t, M, R: real;
Function F(x : real): real;
 begin
  F := (x - 1) * (x - 3);
 end;
BEGIN
 a := -3; b := 3;
 d := 0.1;
 t := a; M := a; R := F(a);
 while t < b do
  begin
   if (F(t) < R) then
    begin
     M := t; R := F(t);
    end;
   t: = t + d;
  end;
 write(M);
END.
```

Си

```
#include <stdio.h>
double F(double x)
{return (x - 1) * (x - 3);
}
void main()
{double d, a, b, t, M, R;
 a = -3; b = 3; d = 0.1;
 t = a; M = a; R = F(a);
 while (t < b) {
  if ( F(t) < R ) {
   M = t; R = F(t);
  }
  t = t + d;
 }
printf("%f", M);
}
```

Алгоритмический язык

```
англ 14
нач
  вещ d, a, b, t, M, R
  a := -3; b := 3; d := 0.1
  t := a; M := a; R := F(a)
  нц пока t < b
    если F(t) < R
    то M := t; R := F(t)
    все t := t + d
  кц
  вывод M
кон
англ вещ F(вещ x)
нач
  знач := (x - 1) * (x - 3)
кон
```

1) -1 **2) 2** **3) -3** **4) 24**

Решение

Видя сходство предлагаемого текста программы с разобранным в предыдущей задаче, можно упростить решение.

Из текста программы извлекается «ключевая» информация об обрабатываемом графике функции:

- диапазон изменения аргумента функции [a,b] — от −3 до 3;
- шаг изменения аргумента функции d — 0,1;
- формула, определяющая функцию: $F = (x - 1) \cdot (x - 3)$.

Раскрыв в записи этой функции скобки, получается: $x^2 - 4x + 4$, т. е. опять получается квадратичная функция, график которой представляет собой параболу. При этом «нули» этой функции (точки пересечения её графика с осью абсцисс) нетрудно найти из исходной записи функции: $(x - 1) \cdot (x - 3) = 0 \Rightarrow x_1 = 1, x_2 = 3$.

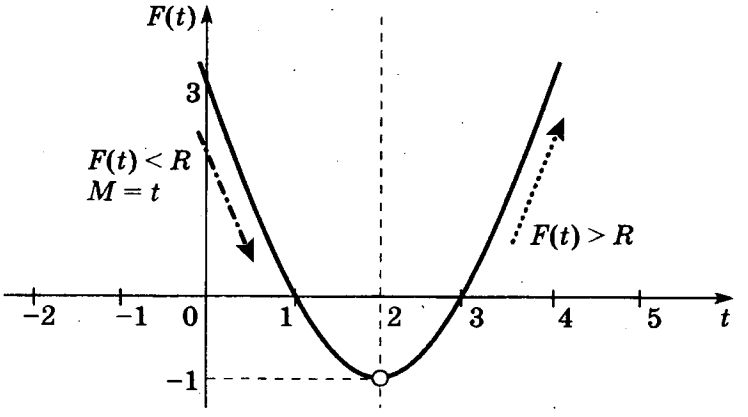
Чтобы определить направление ветвей параболы, определяются знаки значений заданной функции в трёх точках, одна из которых расположена внутри интервала между найденными корнями, а две другие – вне этого интервала:

- $x = 0 \Rightarrow F = (0 - 1) \cdot (0 - 3) = 3$ (знак «+»);
- $x = 2 \Rightarrow F = (2 - 1) \cdot (2 - 3) = -1$ (знак «−»);
- $x = 4 \Rightarrow F = (4 - 1) \cdot (4 - 3) = 3$ (знак «+»).

Таким образом, ветви параболы направлены вверх.

Вершина параболы (точка её минимума) определяется приравниванием нулю производной заданной функции:

$F = (x - 1) \cdot (x - 3) \Rightarrow F' = 2x - 4 \Rightarrow 2x - 4 = 0 \Rightarrow x = 2 \Rightarrow F = (2 - 1) \cdot (2 - 3) = -1$.



Внимание! Фрагмент текста программы:

```
if (F(t) < R) then
 begin
   M := t;
   R := F(t);
 end;
```

До каких пор будет производиться перезапись значения переменной M?

В предыдущей задаче аналогичный фрагмент листинга содержал условие $F(t) > R$, и подобная перезапись значений M происходила, пока последующее значение функции оказывалось больше, чем запомненное в переменной R предыдущее, т. е. происходил подъём по графику до точки максимума. В данном же случае условие обратное: $F(t) < R$. Оно показывает, что изменение значений M будет происходить, пока последующее значение функции будет меньше предыдущего. Это соответствует спуску до минимальной точки параболы.

Тогда по аналогии с предыдущей задачей можно предположить, что последним перезаписанным значением переменной M будет значение t, соответствующее точкам минимума параболы, т. е. 2.

Ответ: в результате работы данной программы будет выведено число 2 (вариант ответа №2).

Итак, в подобных задачах, видя их аналогию, можно упростить решение:

- проанализировать вид графика заданной функции;
- определить, выполняется ли подъём по графику до максимальной точки или спуск до минимальной;
- найти эту максимальную или минимальную точку и записать в качестве ответа соответствующее ей значение аргумента (*x* или *t*).

Однако подобный способ решения — достаточно рискованный, поскольку опирается на предположение, что в условии задачи изменён только вид функции и, возможно, диапазон изменения аргумента. Но если составители заданий ЕГЭ поменяют сам алгоритм, то такое «упрощённое» решение будет неверным. Поэтому на реальном ЕГЭ, если остаётся время, лучше дополнительно провести решение с полной трассировкой программы, показанное в задаче 1.

Задача 3[1]. Определите, какое число будет напечатано в результате работы следующей программы (для Вашего удобства программа представлена на четырёх языках):

Бейсик

```
Module A14
 Sub Main()
 Dim d, a, b, t, M, R As Double
     a = 0; b := 1
     d = 0.1
     t = a; M = a; R = F(a)
     While t < b
      If F(t) < R Then
         M = t
         R = F(t)
      End If
      t = t + d
     End While
     Console.Write(M)
End Sub
Function F(ByVal x As Double) As Double
     Return (x - 1) * (x - 3)
     End Function
End Module
```

Паскаль

```
Program A14;
Uses crt;
Var d, a, b, t, M, R: real;
Function F(x : real): real;
 begin
  F := (x - 1) * (x - 3);
 end;
BEGIN
 a := 0; b := 1; d := 0.1;
 t := a; M := a; R := F(a);
 while t < b do
  begin
   if (F(t) < R) then
    begin
     M := t; R := F(t);
    end;
   t := t + d;
  end;
 write(M);
END.
```

Си

```
#include <stdio.h>
double F(double x)
{
 return (x - 1) * (x - 3);
}
void main()
 {
  double d, a, b, t, M, R;
  a = 0; b = 1;
  d = 0.1;
  t = a; M = a; R = F(a);
  while (t < b) {
   if (F(t) < R ) {
    M = t; R = F(t);
   }
  t = t + d;
 }
printf("%f", M);
}
```

Алгоритмический язык

```
англ 14
нач
   вещ d, a, b, t, M, R
  a := 0; b := 1
  d := 0.1
  t := a; M := a; R := F(a)
  нц пока t < b
   если F(t) < R
    то
     M := t; R := F(t)
    все
    t := t + d
  кц
  вывод M
кон
англ вещ F(x)
нач
 знач := (x - 1) * (x - 3)
кон
```

1) 1 **2) 2** **3) –3** **4) 24**

[1] Незначительная модификация предыдущей задачи, выделенная в листингах жирным шрифтом. Однако, она приводит к значительному изменению в решении.

Решение

Вновь выполняется «упрощённое» решение.

Извлекается из текста программы «ключевая» информация об обрабатываемом графике функции:

- диапазон изменения аргумента функции [a,b] — от 0 до 1;
- шаг изменения аргумента функции d — 0,1;
- формула, определяющая функцию: $F = (x - 1) \cdot (x - 3)$.

Раскрыв в записи этой функции скобки, получится: $x^2 - 4x + 4$, т. е. имеется квадратичная функция, график которой представляет собой параболу. При этом «нули» этой функции (точки пересечения её графика с осью абсцисс) нетрудно найти из исходной записи функции: $(x - 1) \cdot (x - 3) = 0 \Rightarrow x_1 = 1, x_2 = 3$.

Чтобы определить направление ветвей параболы, нужно определить знаки значений заданной функции в трёх точках, одна из которых расположена внутри интервала между найденными корнями, а две другие — вне этого интервала:

- $x = 0 \Rightarrow F = (0 - 1) \cdot (0 - 3) = 3$ (знак «+»);
- $x = 2 \Rightarrow F = (2 - 1) \cdot (2 - 3) = -1$ (знак «−»);
- $x = 4 \Rightarrow F = (4 - 1) \cdot (4 - 3) = 3$ (знак «+»).

Таким образом, ветви параболы направлены вверх.

Вершина параболы (точка её минимума) определяется приравниванием нулю производной заданной функции:

$F = (x - 1) \cdot (x - 3) \Rightarrow F' = 2x - 4 \Rightarrow 2x - 4 = 0 \Rightarrow x = 2 \Rightarrow F = (2 - 1) \cdot (2 - 3) = -1$.

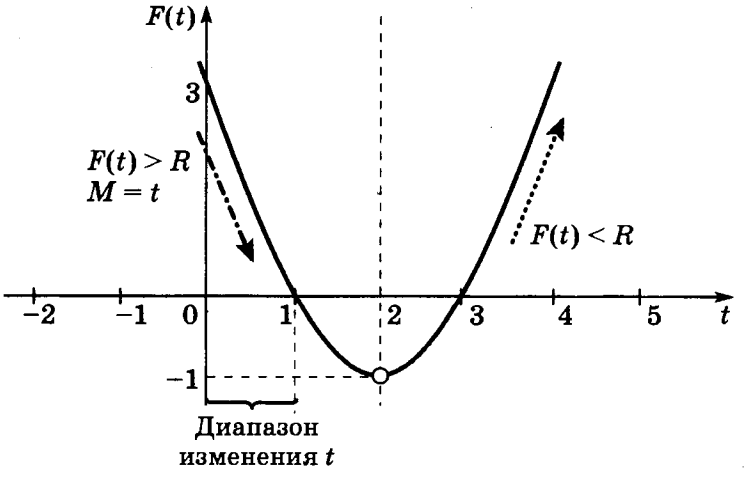
Внимание! Фрагмент текста программы:

```
if (F(t) < R) then
  begin
    M := t;
    R := F(t);
  end;
```

До каких пор будет производиться перезапись значения переменной M?

В предыдущей задаче аналогичный фрагмент листинга содержал условие $F(t) > R$, и подобная перезапись значений M происходила, пока последующее значение функции оказывалось больше, чем запомненное в переменной R предыдущее, т. е. происходил подъем по графику до точки максимума. В данном же случае условие обратное: $F(t) < R$. Оно показывает, что изменение значений M будет происходить, пока последующее значение функции будет меньше предыдущего. Это соответствует спуску до минимальной точки параболы.

До сих пор решение практически полностью повторяло запись решения предыдущей задачи. Теперь следует вспомнить про интервал изменения переменной t и наложить его на вычерченный график.



На данном диапазоне изменения переменной t точка минимума параболы **не достигается!** Это означает, что при работе рассматриваемой программы спуск по графику будет производиться только на указанной части левой ветви параболы и этот спуск прервётся просто по завершению цикла перебора значений t. Поэтому ответом будет являться в данном случае уже не абсцисса точки минимума функции, а просто правая граница заданного интервала изменения абсцисс — точка 1.

Ответ: в результате работы данной программы будет выведено число 1 (вариант ответа №1).

При решении таких задач, анализируя вид графика заданной функции, следует не забывать проверять, находится ли точка её минимума (максимума) в заданном интервале изменения абсцисс!

Задачи для самостоятельного решения

1. Определите, какое число будет напечатано в результате работы следующей программы:

```
Program A14;
Uses crt;
Var d, a, b, t, M, R: real;
Function F(x : real) : real;
  begin
    F := (x - 3) * (x + 2);
  end;
BEGIN
  a := 1; b := 4;
  d := 0.01;
  t := a; M := a; R := F(a);
  while t < b do
    begin
      if (F(t) < R) then
        begin
          M := t;
          R := F(t);
        end;
      t := t + d;
    end;
  write(M);
END.
```

2. Определите, какое число будет напечатано в результате работы следующей программы:

```
Program A14;
Uses crt;
Var d, a, b, t, M, R: real;
Function F(x : real) : real;
  begin
    F := (x - 2) * (x - 5);
  end;
BEGIN
  a := 0; b := 4;
  d := 0.1;
```

```
    t := a; M := a; R := F(a);
    while t < b do
      begin
        if (F(t) < R) then
          begin
            M := t;
            R := F(t);
          end;
        t := t + d;
      end;
    write(M);
  END.
```

3. Определите, какое число будет напечатано в результате работы следующей программы:

```
  Program A14;
  Uses crt;
  Var d, a, b, t, M, R: real;
  Function F(x : real) : real;
    begin
      F := x * x + 9 * x + 18;
    end;
  BEGIN
    a := -8; b := -5;
    d := 0.05;
    t := a; M := a; R := F(a);
    while t < b do
      begin
        if (F(t) < R) then
          begin
            M := t;
            R := F(t);
          end;
        t := t + d;
      end;
    write(M);
  END.
```

4. Определите, какое число будет напечатано в результате работы следующей программы:

```
  Program A14;
  Uses crt;
  Var d, a, b, t, M, R: real;
  Function F(x : real) : real;
    begin
      F := -x * x + 2 * x -2;
    end;
  BEGIN
    a := -3; b := 3;
    d := 0.001;
    t := a; M := a; R := F(a);
    while t < b do
     begin
       if (F(t) > R) then
```

```
        begin
          M := t;
          R := F(t);
        end;
      t := t + d;
    end;
  write(M);
END.
```

5. Определите, какое число будет напечатано в результате работы следующей программы:

```
Program A14;
Uses crt;
Var d, a, b, t, M, R: real;
Function F(x : real) : real;
  begin
    F := -x * x + 4;
  end;
BEGIN
  a := -1; b := 1;
  d := 0.2;
  t := a; M := a; R := F(a);
  while t < b do
   begin
     if (F(t) > R) then
       begin
         M := t;
         R := F(t);
       end;
     t := t + d;
   end;
  write(M);
END.
```

Ответы для самопроверки

Задача	Ответ
1	0,5
2	3,5
3	−5
4	1
5	0

Программирование

B14 Процедуры и функции

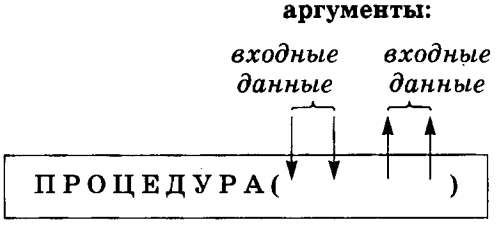
Конспект

Подпрограмма

Подпрограмма — это поименованная или каким-либо иным образом обозначенная часть программы, которая может быть многократно вызвана из разных частей основной программы для выполнения неких «типовых» вычислений, в том числе для различных исходных данных.

Процедура

Отличительная особенность подпрограммы-процедуры (или просто — **процедуры**) состоит в том, что она (в отличие от функции) может как принимать на вход, так и возвращать любое количество данных, причём все они записываются в качестве аргументов процедуры.



Для работы с процедурами в языке Паскаль (и в большинстве других языков высокого уровня) требуется знать следующее:

1. Процедура должна быть *описана* в начале программы — после объявления используемых в ней *глобальных* констант, меток и переменных, но до собственно текста основной программы, заключенного в операторные скобки BEGIN и END (для наглядности эти слова, обрамляющие текст основной программы, записаны прописными буквами).

2. Описание подпрограммы-функции начинается со строки

```
Procedure <имя процедуры>(<арг1>, <арг2>, ... : <тип1>; <арг3>, <арг4>,
... : <тип2>; ...) : <тип результата>;
```

Здесь:

— `Procedure` — зарезервированное слово, указывающее транслятору, что далее идёт описание подпрограммы-процедуры;

— `<имя процедуры>` — идентификатор, выбираемый по тем же правилам, что и для имён переменных;

— сразу после имени функции в скобках записывается перечень передаваемых в эту функцию *формальных параметров* — имён переменных, которые будут использоваться при вы-

числениях, выполняемых в процедуре, и имён переменных, в которые процедура должна записать полученные результаты. Формальные параметры (как и в функциях) группируются через запятую в блоки одинаковых типов, а обозначения их типов записываются после списка параметров через двоеточие; блоки параметров различного типа записываются через точку с запятой;

— в процедуру, кроме обычных значений (числовых или символьных) можно передавать данные составных типов (массивы, записи и пр.). Для этого надо объявить такой составной тип как *пользовательский* в разделе `type` в начале программы, далее объявить в разделе `var` соответствующий *экземпляр* данных как переменную этого составного типа, а далее при определении процедуры указать для соответствующих формальных параметров этот составной (пользовательский) тип.

Пример (из листинга, приведённого в условии разбираемой нами задачи):

`Procedure Pr1(L : integer; var R : atype );` — объявляется процедура с именем `Pr1`, которая работает с целым (тип `integer`) числом `L` и массивом `R`, тип которого (`atype`) был ранее определён в разделе `type`:

```
type
    atype = array [1..L] of integer;
```

При этом параметры процедуры могут передаваться в неё как значения или как ссылки. В чём заключается различие между этими двумя способами передачи данных, будет рассмотрено позже. Пока учитывается, что составные данные (в том числе массивы) обычно передаются как ссылки, а признаком передачи ссылки в процедуру является запись служебного слова `var` перед именем формального параметра (как и сделано в данной программе для передачи в процедуру массива `R`).

3. После строки определения процедуры записывается её программный код. Правила его записи аналогичны правилам записи основной программы:

— сначала может присутствовать раздел описания *локальных* констант и переменных, действующих только в пределах данной процедуры;

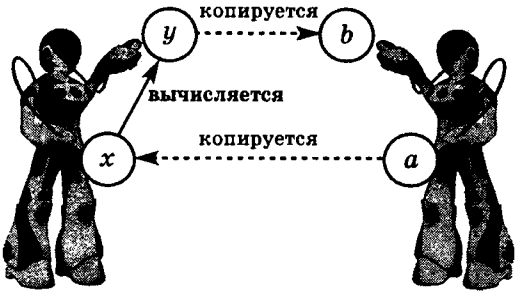
— далее между операторными скобками `begin` и `end` записывается текст подпрограммы.

4. Результаты вычислений, которые процедура должна вернуть в основную программу, должны быть записаны в качестве значения соответствующего формального параметра.

5. В тексте основной программы вызов процедуры производится следующим образом:

— записывается имя процедуры, а после него в скобках через запятую — список *фактических параметров* — константы, имена переменных, выражения, а также другие вызовы процедур или функций, значения которых нужно передать в процедуру для выполнения вычислений. Переменные, в которые процедура должна вернуть результаты вычислений, также записываются как её фактические параметры (и это — переменные, объявленные в основной программе!). Количество и типы всех фактических параметров должны соответствовать указанным в объявлении процедуры формальным параметрам. При этом, если параметр передается как значение, то это значение при вызове подпрограммы будет *скопировано* в соответствующий формальный параметр, так что если процедура в процессе вычислений внутри неё изменит значение этого формального параметра, это никак не повлияет на значение «фактической» переменной в основной программе. А вот если параметр передаётся как ссылка, то в процедуру попадёт информация о расположении в оперативной памяти компьютера самих исходных данных (например, массива), и тогда все изменения, которые процедура совершит с таким формальным параметром, будут отражены в самом исходном массиве. Иными словами, передавая массив (или другой экземпляр составных данных) в процедуру по ссылке, процедуре отдаётся вся «власть» выполнять любые изменения с этим массивом. Соответственно, для возвращения результата изменений параметра, переданного как ссылка, не требуется предусматривать какой-то отдельный параметр: все эти изменения сразу производятся в исходном экземпляре данных.

Передача данных как значения



Значение переменной *a* в основной программе **не изменяется**

Передача данных как ссылки

Значение переменной *M* — экземпляра массива — процедура **изменяет** прямо в основной программе

— поскольку процедура возвращает результаты своей работы через свои параметры, она записывается как отдельная команда программы, а не как составная часть выражения или оператора присваивания.

Разбор типовых задач

Задача 1. Что будет напечатано в результате выполнения этой программы?

```
Program Task;
Uses crt;
const L = 4;
type
   atype = array [1..L] of integer;
Var R: atype;
    N, p: integer;
Procedure Pr1(L: integer; var R: atype );
 var i, n, t: integer;
begin
 for i := 1 to L do
 begin
  t := (R[i] div 2) * 4; R[i] := t mod 5;
 end;
end;
Function F1 (L: integer; R: atype) : integer;
 Var N, i, T: integer;
begin
 N := 1; T := 1;
 for i := 1 to L do
 begin
  N := N * R[i] + T; T := T + 2;
 end;
```

```
  F1 := N;
end;
BEGIN
  R[1] := 6; R[2] := 10; R[3] := 7; R[4] := 3;
  Pr1(L, R);
  N := F1(L,R);
  write(N);
  writeln;
END.
```

Решение

Эта задача (хотя приведённый в её условии программный код более объёмен и содержит, кроме подпрограммы-функции, также подпрограмму-процедуру) все же гораздо проще, чем ранее рассмотренные задачи типа A14. По крайней мере, здесь не требуется трассировка аж по 7 переменным на цикле из 60 шагов. Но решение данной задачи все равно потребует изрядных усилий и знаний.

Зная принципы программирования процедур и отличия процедур от функций, можно проанализировать предлагаемый в задаче программный код:

`Program Task;`	Начало основной программы
`Uses crt;`	
`const L = 4;`	*L* — длина массива
`type`	Определение составного пользовательского типа (целочисленного массива из 4 элементов)
`  atype = array [1..L] of integer;`	
`Var R: atype;`	Определение экземпляра такого массива
`  N, p: integer;`	
`Procedure Pr1(L: integer; var R:  atype);`	Описание процедуры. Ей передаётся значение переменной *L* (как значение) и массив *R* (как ссылка)
`  var i, n, t: integer;`	
`begin`	
`  for i := 1 to L do`	
`  begin`	Процедура непосредственно меняет значения элементов массива *R*, «принадлежащего» основной программе
`    t := (R[i] div 2) * 4;`	
`    R[i] := t mod 5;`	
`  end;`	
`end;`	
`Function F1 (L: integer; R: atype): integer;`	Определение подпрограммы-функции, которой также передаётся переменная *L* и массив *R* (оба параметра — как значения)
`  Var N, i, T: integer;`	
`begin`	
`  N := 1;`	
`  T := 1;`	
`  for i := 1 to L do`	
`  begin`	
`    N := N * R[i] + T;`	
`    T := T + 2;`	
`  end;`	
`  F1 := N;`	Вычисленное значение *N* будет возвращено в основную программу как значение функции
`end;`	
`BEGIN`	Основная программа
`  R[1] := 6; R[2] := 10; R[3] := 7; R[4] := 3;`	
`Pr1(L, R);`	Вызов процедуры (она изменит массив)
`  N := F1(L,R);`	Вызов функции
`  write(N);`	
`  writeln;`	Печатается значение, возвращённое функцией
`END.`	

Начинается «трассировка» переменных в этой программе.

1. *Основная программа:* группа присваиваний `R[1] := 6; R[2] := 10; R[3] := 7; R[4] := 3;` — формируется массив $R = [6, 10, 7, 3]$.

2. *Основная программа:* команда `Pr1(L, R);` — вызов процедуры, ранее описанной как `Procedure Pr1(L: integer; var R: atype);`. Ей в качестве параметров передаётся значение переменной `L` и ссылка на массив `R`, значит, элементы этого массива данная процедура будет менять напрямую.

3. *Процедура Pr1:* обратившись к её программному коду, выполняется трассировка в виде таблицы:

Передано: $L = 4$, $R[] = [6, 10, 7, 3]$

i	t	$R[]$
1	$(R[1] \text{ div } 2) \cdot 4 = (6 \text{ div } 2) \cdot 4 =$ $= 3 \cdot 4 =$ **12**	$t \bmod 5 =$ $12 \bmod 5 =$ **2**
2	$(R[2] \text{ div } 2) \cdot 4 = (10 \text{ div } 2) \cdot 4 =$ $= 5 \cdot 4 =$ **20**	$t \bmod 5 =$ $20 \bmod 5 =$ **0**
3	$(R[3] \text{ div } 2) \cdot 4 = (7 \text{ div } 2) \cdot 4 =$ $= 3 \cdot 4 =$ **12**	$t \bmod 5 =$ $12 \bmod 5 =$ **2**
4	$(R[4] \text{ div } 2) \cdot 4 = (3 \text{ div } 2) \cdot 4 =$ $= 1 \cdot 4 =$ **4**	$t \bmod 5 =$ $4 \bmod 5 =$ **4**

Таким образом, после завершения работы процедуры `Pr1` массив `R` (в основной программе!) будет иметь значения элементов: [2, 0, 2, 4].

4. *Основная программа:* команда `N := F1(L, R);` — вызов функции, ранее описанной как `Function F1 (L: integer; R: atype) : integer;`. Ей в качестве параметров передаются значение переменной `L` и массив `R` (как значения), а вернуть она должна целое число.

5. *Функция F1:* обратившись к её программному коду, выполняется трассировка в виде таблицы:

Передано: $L = 4$, $R[] = [2, 0, 2, 4]$

i	N	T
—	1	1
1	$N \cdot R[1] + T = 1 \cdot 2 + 1 =$ **3**	$T + 2 = 1 + 2 =$ **3**
2	$N \cdot R[2] + T = 3 \cdot 0 + 3 =$ **3**	$T + 2 = 3 + 2 =$ **5**
3	$N \cdot R[3] + T = 3 \cdot 2 + 5 =$ **11**	$T + 2 = 5 + 2 =$ **7**
4	$N \cdot R[4] + T = 11 \cdot 4 + 7 =$ **51**	$T + 2 = 7 + 2 =$ **9**

Функция возвращает значение N (которое присваивается имени этой функции — `F1 := N;`), равное 51. Именно это значение будет выведено на экран последующей командой `write(N);` и является ответом в данной задаче.

Ответ: 51.

Задача 2. Что будет напечатано в результате выполнения этой программы?

```
Program Task;
Uses crt;
const L = 5;
type
   atype = array [1..L] of integer;
Var R: atype;
   N, p: integer;
```

```
Procedure Multiply3_2(L, p: integer; var R: atype );
     var i,n,t : integer;
begin
p := 0;
for i := 1 to L do
 begin
   t := 2 * R[i] + p;
   R[i] := (t)mod(3);
   p := (t)div(3);
 end;
end;

Function Calc3 (L: integer; R: atype) : integer;
var N, i, T: integer;
begin
   N := 0;
   T := 1;
   for i := 1 to L do
    begin
      N := N + T * R[i];
      T := T * 3;
    end;
   Calc3 := N;
end;

BEGIN
   R[1] := 2;
   R[2] := 2;
   R[3] := 0;
   R[4] := 1;
   R[5] := 0;
   Multiply3_2(L, p, R);
   if (p > 0) then
    begin
      write('Переполнение');
      halt;
    end;
   N := Calc3(L,R);
   write(N);
   writeln;
END.
```

Решение

По аналогии с предыдущей задачей, можно выделить три «ключевых» блока: это *основная программа*, в которой задаётся исходный массив и откуда вызываются процедура и функция; *процедура*, которая меняет исходный массив, и *функция*, которая по этому массиву вычисляет некоторое числовое значение.

1) *задание массива:*

```
R[1] := 2; R[2] := 2; R[3] := 0; R[4] := 1; R[5] := 0;
```

Следовательно, массив $R = [2, 2, 0, 1, 0]$.

2) *обработка массива в процедуре:*
Передано: $L = 5$, $R[] = [2, 2, 0, 1, 0]$

i	*t*	*R*[]	*p*
—	—	—	0
1	2 · R[i] + p = 2 · R[1] + p = = 2 · 2 + 0 = **4**	(t)mod(3) = = 4 mod 3 = **1**	(t)div(3) = 4 div 3 = **1**
2	2 · R[i] + p = 2 · R[2] + p = = 2 · 2 + 1 = **5**	(t)mod(3) = = 5 mod 3 = **2**	(t)div(3) = 5 div 3 = **1**
3	2 · R[i] + p = 2 · R[3] + p = = 2 · 0 + 1 = **1**	(t)mod(3) = = 1 mod 3 = **1**	(t)div(3) = 1 div 3 = **0**
4	2 · R[i] + p = 2 · R[4] + p = = 2 · 1 + 0 = **2**	(t)mod(3) = = 2 mod 3 = **2**	(t)div(3) = 2 div 3 = **0**
5	2 · R[i] + p = 2 · R[5] + p = = 2 · 0 + 0 = **0**	(t)mod(3) = = 0 mod 3 = **0**	(t)div(3) = 0 div 3 = **0**

Таким образом, после завершения работы процедуры `Pr1` массив `R` (в основной программе) будет иметь значения элементов: [1, 2, 1, 2, 0].

Кроме того, возвращается значение *p*, равное нулю. Условие в основной программе: p>0 ложно, поэтому сообщение о «переполнении» выдано не будет, а будет вызвана функция.

3) *обработка массива в функции:*
Передано: $L = 5$, $R[] = [1, 2, 1, 2, 0]$

i	*N*	*T*
—	0	1
1	N + T · R[i] = 0 + 1 · *R*[1] = = 0 + 1 · 1 **= 1**	T · 3 = 1 · 3 = **3**
2	N + T · R[i] = 1 + 3 · *R*[2] = = 1 + 3 · 2 **= 7**	T · 3 = 3 · 3 = **9**
3	N + T · R[i] = 7 + 9 · *R*[3] = = 7 + 9 · 1 **= 16**	T · 3 = 9 · 3 = **27**
4	N + T · R[i] = 16 + 27 · *R*[4] = = 16 + 27 · 2 **= 70**	T · 3 = 27 · 3 = **81**
5	N + T · R[i] = 70 + 81 · *R*[5] = = 70 + 81 · 0 **= 70**	T · 3 = 81 · 3 = **243**

Функция возвращает значение *N* (которое присваивается имени этой функции — `Calc3:=N;`), равное 70. Именно это значение будет выведено на экран последующей командой `write(N);` и является ответом в данной задаче.

Ответ: 70.

Задачи для самостоятельного решения

1. Что будет напечатано в результате выполнения этой программы?

```
Program Task;
Uses crt;
const L = 5;
type
    atype = array [1..L] of integer;
Var R: atype;
    N, p: integer;

Procedure Multiply3_2(L, p: integer; var R: atype);
var i, n, t: integer;
begin
p := 2;
for i := 1 to L do
 begin
  t := R[i] * p - i;
  R[i] := t + p;
  p := t + i;
 end;
end;

Function Calc3 (L: integer; R: atype) : integer;
var N, i, T: integer;
begin
  N := 1;
  T := 1;
  for i := 1 to L do
   begin
    N := N * T * R[i];
    T := T + i;
   end;
  Calc3 := N;
end;

BEGIN
  R[1] := 1; R[2] := 1; R[3] := 0;
  R[4] := 1; R[5] := 1;
  Multiply3_2(L, p, R);
  if (p > 0) then
   begin
    write('Переполнение');
    halt;
   end;
  N := Calc3(L,R);
  write(N);
  writeln;
END.
```

Что будет напечатано в результате выполнения этой программы?

```
Program Task;
Uses crt;
const L = 4;
type
    atype = array [1..L] of integer;
Var R: atype;
    N, p: integer;

Procedure Multiply3_2(L, p: integer; var R: atype );
var i, n, t: integer;
begin
p := 5;
for i := 1 to L do
 begin
  t := R[i] * i + p;
  R[i] := t * p;
  p := t * i;
 end;
end;

Function Calc3 (L: integer; R: atype) : integer;
var N, i, T: integer;
begin
  N := 1;
  T := 1;
  for i :=1 to L do
   begin
    N := (N + T) * R[i] * R[i];
    T := T + i;
   end;
  Calc3 := T;
end;

BEGIN
  R[1] := 1;
  R[2] := 0;
  R[3] := 1;
  R[4] := 0;
  Multiply3_2(L, p, R);
  if (p = 0) then
   begin
    write('Некорректные данные');
    halt;
   end;
  N := Calc3(L,R);
  write(N);
  writeln;
END.
```

3. Что будет напечатано в результате выполнения этой программы?

```
Program Task;
Uses crt;
const L = 3;
type
    atype = array [1..L] of integer;
Var R: atype;
    N, p: integer;

Procedure Multiply3_2(L, p: integer; var R: atype );
var i, n, t: integer;
begin
p := 1;
for i := 1 to L do
 begin
  t := (R[i] + i) * p; R[i] := t + p; p:= -(i mod 2);
 end;
end;

Function Calc3 (L: integer; R: atype) : integer;
var N, i, T: integer;
begin
  N := 1; T := 0;
  for i := 1 to L do
   begin
    N := R[i] + T * N * N;
    T := (i mod 2) + 1;
   end;
  Calc3 := N * N;
end;

BEGIN
  R[1] := -1; R[2] := 1; R[3] := -1;
  Multiply3_2(L, p, R);
  if (p >= 0) then
   begin
    write('Некорректный массив');
    halt;
   end;
  N := Calc3(L,R);
  write(N);
  writeln;
END.
```

4. Что будет напечатано в результате выполнения этой программы?

```
Program Task;
Uses crt;
const L = 4;
type
    atype = array [1..L] of integer;
Var R: atype;
    N, p: integer;
```

```
Procedure Multiply3_2(L, p: integer; var R: atype);
var i, n, t: integer;
begin
p := R[1];
for i := 1 to L do
 begin
  t := p mod 10; R[i] := t * i; p := p div 10;
 end;
end;

Function Calc3 (L: integer; R: atype) : integer;
var N, i, T: integer;
begin
  N := 0;
  T := 0;
  for i := 1 to L do
   begin
    N := (R[i] mod 2) - T;
    T := N + T;
   end;
  Calc3 := N;
end;

BEGIN
  R[1] := 3725; R[2] := 0; R[3] := 0; R[4] := 0;
  Multiply3_2(L, p, R);
  if (p <> 0) then
   begin
    write('Неверные данные в R[1]');
    halt;
   end;
  N := Calc3(L,R);
  write(N);
  writeln;
END.
```

Что будет напечатано в результате выполнения этой программы?

```
Program Task;
Uses crt;
const L = 5;
type
    atype = array [1..L] of integer;
Var R: atype;
    N, p: integer;

Procedure Multiply3_2(L, p: integer; var R: atype);
var i, n, t: integer;
begin
p := R[3];
```

```
  for i := 1 to L do
   begin
    t := p mod 10;
    R[i] := t + (i mod 2);
    p := p div 10;
   end;
end;

Function Calc3 (L: integer; R: atype) : integer;
var N, i, T: integer;
begin
   N := 0;
   T := 0;
   for i := L downto 1 do
    begin
     N := N * 10 + R[i] - T;
     T := N mod 2;
    end;
   Calc3 := N * T;
end;

BEGIN
  R[1] := 1234; R[2] := 2341; R[3] := 3412; R[4] := 4123; R[4] := 1234;
   Multiply3_2(L, p, R);
   if (p <> 0) then
    begin
     write('Неверные данные в массиве');
     halt;
    end;
   N := Calc3(L,R);
   if (n = 0) then write('Timeout Error') else write(n);
   writeln;
END.
```

Ответы для самопроверки

Задача	Ответ
1	−73920
2	11. (Внимание! В тексте функции имени функции Calc3 присваивается значение переменной T! Поэтому, хотя в основной программе вызов функции оформлен как N := Calc3(L,R);, а затем выводится это значение *N*, выведено на экран будет именно значение *T*. Следовательно, при трассировке функции значение *N* можно вовсе не вычислять, ограничившись вычислением только переменной *T*).
3	16. (Внимание! Имени функции Calc3 присваивается значение N^2.)
4	−1
5	12503. (Внимание! В функции Calc3 проход по массиву производится в обратном порядке, а имени функции Calc3 присваивается значение N · T.)

Программирование

C1 Задачи на пересечение областей

Конспект

Связь между булевой логикой, теорией множеств и алгеброй функций

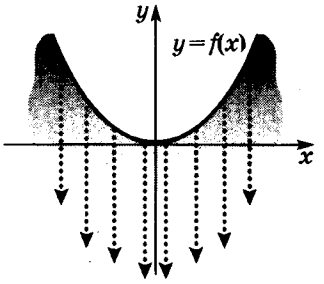
1. **Области**, выделяемые графиками функций на координатной плоскости, могут рассматриваться как **множества точек** этой плоскости (как замкнутые, так и открытые, если они не ограничены графиками со всех сторон).

2. Пересекающиеся множества точек на плоскости могут рассматриваться как аналог кругов Эйлера-Венна: для таких областей аналогичным образом выполняются логические операции **НЕ** (NOT), **И** (AND) и **ИЛИ** (OR), определяющие, соответственно:

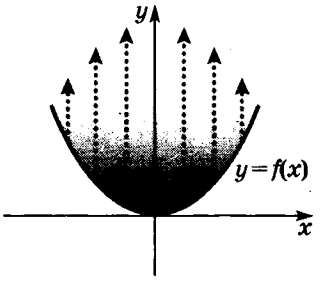
- операция **И** (AND) — пересечение областей (т. е. множество точек плоскости, принадлежащих обеим соединяемым этой операцией областям);
- операция **ИЛИ** (OR) — объединение областей (т. е. множество точек плоскости, принадлежащих любой из соединяемых этой операцией областей, в том числе их пересечению);
- операция **НЕ** (NOT) — множество точек плоскости, не принадлежащих области.

3. Границы областей определяются соответствующими графиками. При этом:

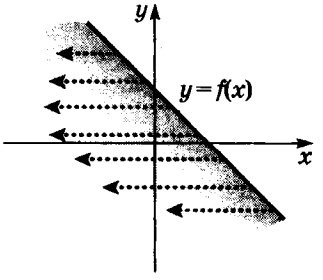
- один график функции обычно определяет соответствующую открытую область (полуплоскость, в том числе криволинейную), при этом условие, определяющее эту область, представляет собой неравенство вида $y \leqslant F(x)$, $y \geqslant F(x)$, $x \leqslant F^{-1}(y)$ или $x \geqslant F^{-1}(y)$, где F^{-1} — обратная функция по отношению к F; указанное неравенство определяет условие принадлежности заданной точки данной области;



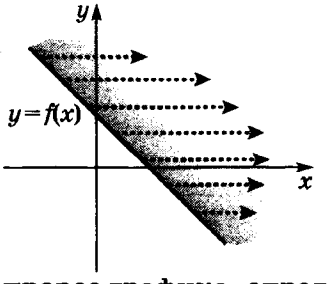
Всё, что **ниже** графика, определяется условием $y \leqslant F(x)$



Всё, что **выше** графика, определяется условием $y \geqslant F(x)$



Всё, что **левее** графика, определяется условием $x \leqslant F^{-1}(y)$



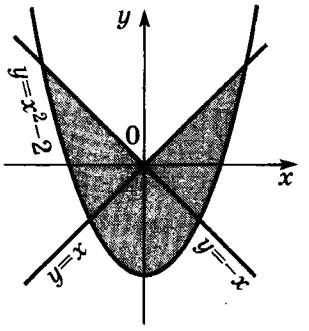
Всё, что **правее** графика, определяется условием $x \geqslant F^{-1}(y)$

- один график функции может определять замкнутую область, если функция имеет вид $F(x,y) = const$ (пример — $x^2 + y^2 = 1$ — окружность единичного радиуса), в этом случае знак $\leqslant$ или $\geqslant$, записанный вместо знака равенства в уравнении функции, определяет соответственно внутреннюю или наружную область относительно замкнутой кривой; указанное неравенство определяет условие принадлежности заданной точки данной области;
- замкнутая область, образуемая пересекающимися графиками функций, определяется как логическое пересечение (**И**, AND) или объединение (**ИЛИ**, OR) отдельных элементарных областей (замкнутых или открытых); условие принадлежности точки такой области определяется как сложное логическое выражение, образованное элементарными условиями сравнения (условиями принадлежности точки каждой из элементарных областей) и указанными логическими операциями: **И** (AND) — для пересечения элементарных областей, **ИЛИ** (OR) — для их объединения.

Разбор типовых задач

Задача 1*. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы).

Программист торопился и написал программу неправильно.



Программа на Паскале	Программа на Бейсике	Программа на Си

var x, y: real;
begin
readln(x, y);
if y <= x then
if y <= -x then
if y >= x * x - 2 then
write('принадлежит')
else
write('не принадлежит')
end.
	INPUT x, y
	IF y <= x THEN
	IF y <= -x THEN
	IF y >= x * x - 2 THEN
	PRINT "принадлежит"
	ELSE
	PRINT "не принадлежит"
	ENDIF
	ENDIF
	ENDIF
	END
		void main(void)
		{ float x, y;
		scanf("%f%f",&x,&y);
		if (y <= x)
		if (y <= -x)
		if (y >= x * x - 2)
		printf("принадлежит");
		else
		printf("не
		принадлежит");
		}

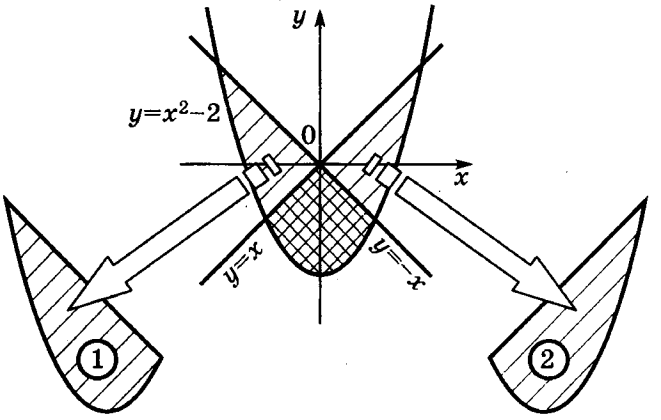
Последовательно выполните следующее:

1) Приведите пример таких чисел x, y, при которых программа неправильно решает поставленную задачу.

2) Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой правильный способ доработки исходной программы.)

Решение

Первое, что надо научиться делать, — это выделять на рисунке «подобласти» (в том числе прилегающие друг к другу или перекрывающиеся), границы которых соответствуют линиям графиков, — причём желательно, чтобы было минимальным как количество таких «подобластей», так и количество графиков, «участвующих» в построении каждой «подобласти». В нашем случае можно выделить две таких «подобласти», одна из которых ограничена параболой и прямой $y = -x$, а вторая — параболой и прямой $y = x$:



Второй важный момент касается построения условий, описывающих каждую выделенную «подобласть».

Выполняется пересечение полуплоскостей (в данном случае пересечение двух полуплоскостей, в том числе с криволинейными границами, определяемыми указанными графиками функций):

- всё, что расположено **ниже** линии графика, соответствует условию «$y \leqslant f(x)$» (где $f(x)$ — функция, по которой построен график);
- всё, что расположено **выше** линии графика, соответствует условию «$y \geqslant f(x)$»;
- всё, что расположено **левее** линии графика, соответствует условию «$x \leqslant f^{-1}(y)$» (где $f^{-1}(y)$ — функция, обратная заданной для графика; для её получения достаточно в уравнении графика выразить x через y);
- всё, что расположено **правее** линии графика, соответствует условию «$x \geqslant f^{-1}(y)$»

(нестрогость операций сравнения следует из того, что согласно условию задачи точки на линиях границ области (графиков) включаются в эту область).

Область (в данном случае — «подобласть») представляет собой **пересечение** указанных полуплоскостей. Очевидно, что каждая точка, принадлежащая этой области, должна одновременно принадлежать **И** первой полуплоскости, **И** второй, — следовательно, условие, определяющее эту область, можно получить из ранее выведенных «элементарных» условий, используя логическую операцию **И**. (Здесь можно провести аналогию с кругами Эйлера.)

Возвращаясь к задаче:

1. «Подобласть» ① ограничена графиком параболы ($y = x^2 - 2$) и прямой $y = -x$. Причём рассматривается всё, что расположено **выше** параболы (следовательно, первое «элементарное» условие: $y \geqslant x^2 - 2$), и всё, что расположено **ниже** прямой (следовательно, второе «элементарное» условие: $y \leqslant -x$). Тогда данная подобласть определяется логическим условием:

$y \geqslant x^2 - 2$ **AND** $y \leqslant -x$.

2. «Подобласть» ② ограничена графиком параболы ($y = x^2 - 2$) и прямой $y = x$. Причём рассматривается всё, что расположено опять-таки **выше** параболы (следовательно, здесь первое «элементарное» условие: $y \geqslant x^2 - 2$), и всё, что расположено **ниже** прямой (следовательно, второе «элементарное» условие: $y \leqslant x$). Тогда данная подобласть определяется логическим условием:

$y \geqslant x^2 - 2$ **AND** $y \leqslant x$.

3. Третье, что нужно знать, касается объединения выделенных «подобластей» в одну требуемую в условии задачи область. Причём, как уже было сказано, эти «подобласти» могут прилегать друг к другу или перекрываться (как в данном случае).

Очевидно, что для записи условия, обозначающего принадлежность точки **любой** из выделенных «подобластей» нужно соединить записи условий, определяющих каждую «подобласть», при помощи операции **ИЛИ**:

($y \geqslant x^2 - 2$ AND $y \leqslant -x$) OR ($y \geqslant x^2 - 2$ AND $y \leqslant x$).

Собственно, это и есть решение задачи, — то самое условие, которое нужно проверять в программе и записать в **единственном** операторе **if**. Однако в задаче ЕГЭ требуется найти ошибку в приведённом листинге, а также указать пример точек, для которых приведённая в условии задачи программа будет работать неправильно. Поэтому далее нужно перейти к анализу указанного в условии листинга (на примере программы на Паскале).

```
var x, y: real;
begin
readln(x, y);
if y <= x then
if y <= -x then
if y >= x * x - 2 then
write('принадлежит')
else
write('не принадлежит')
end.
```

Здесь первое, что нужно всегда помнить, — это то, что если цепочка из нескольких операторов **if ... then** завершается записью **else**, то эта ветвь **else** *всегда* относится к **последнему** из стоящих перед ней операторов **if**.

Второе же важное правило, которое с очевидностью следует из анализа работы программы с такой цепочкой последовательно записанных вложенных друг в друга операторов **if**, — что такая запись эквивалентна соединению условий, записанных в этих операторах **if**, через логическую операцию **И**.

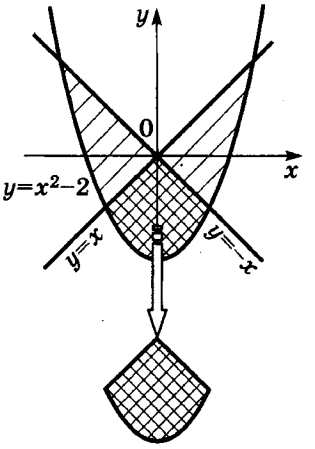
Учитывая всё это, можно записать условие, запрограммированное в приведённом выше листинге, следующим образом:

($y \leqslant x$) AND ($y \leqslant -x$) AND ($y \geqslant x^2 - 2$).

Если сравнить два полученных условия (сформулированное нами из анализа чертежа и «извлечённое» из листинга программы), то сразу бросается в глаза отсутствие во втором условии операции **OR**. Именно в этом и заключается ошибка листинга: зная, что соединение «элементарных» условий через **И** более «строго», чем через **ИЛИ**, нетрудно догадаться, что тем самым в программе «потеряны» какие-то части заданной области. Чтобы понять, какие именно, — расшифровывается второе (ошибочное) условие в виде чертежа, используя рассуждения, обратные тем, которые применялись при составлении условия по чертежу:

- условие «$y \leqslant f(x)$» (где $f(x)$ — функция, по которой построен график) соответствует всему, что расположено **ниже** линии графика;
- условие «$y \geqslant f(x)$» соответствует всему, что расположено **выше** линии графика;
- условие «$x \leqslant f^{-1}(y)$» (где $f^{-1}(y)$ — функция, обратная заданной для графика; для её получения достаточно в уравнении графика выразить x через y) соответствует всему, что расположено **левее** линии графика;
- условие «$x \geqslant f^{-1}(y)$» соответствует всему, что расположено **правее** линии графика;
- соединение этих «элементарных» условий через AND (т. е. через операцию И) соответствует пересечению соответствующих полуплоскостей, а через OR (операцию ИЛИ) — объединению областей (в том числе — с их взаимным наложением), — здесь тоже можно напомнить учащимся про аналогию с кругами Эйлера.

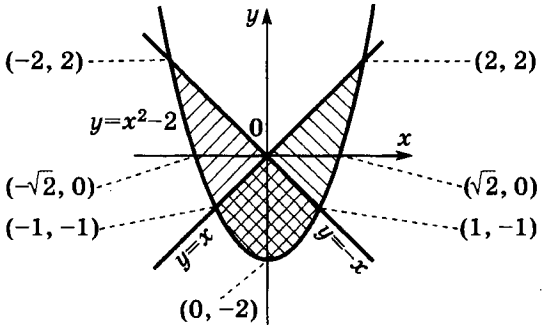
Зная всё это, можно определить, что в программе было заложено условие, обозначающее принадлежность заданной точки области, которая расположена одновременно **выше** графика параболы и **ниже** обеих прямых. То есть речь идёт о «подобласти», которая на рассмотренном ранее чертеже была выделена перекрестной штриховкой:



Отсюда, в качестве примера точек, для которых приведённая в условии задачи программа будет давать неправильный ответ, можно брать любые «удобные» точки из боковых частей области, которые на чертеже выше заштрихованы наклонными линиями. Например, это могут быть точки, расположенные на оси X: (1, 0) или (−1, 0). Или, как указано в ответе к данной задаче в материалах демонстрационного варианта ЕГЭ, точка (2, 2): если решить уравнение $x^2 - 2 = 0$, чтобы определить координаты точек пересечения параболы с осью X, а также решить системы уравнений:

$$\begin{cases} y = x^2 - 2; \\ y = x, \end{cases} \quad \text{и} \quad \begin{cases} y = x^2 - 2; \\ y = -x, \end{cases}$$

чтобы определить координаты точек пересечения заданных прямых с графиком параболы, то можно соответствующим образом «разметить» чертёж и убедиться, что точка (2, 2) находится как раз на пересечении графиков и потому попадает в правую боковую часть областей, которые ошибочная программа «упускает из вида»:



Ответ на первый вопрос задачи получен. Чтобы ответить на её второй вопрос (о необходимой доработке программы), достаточно переписать в виде оператора **if** выведенное ранее условие принадлежности точки заданной области:

($y \geqslant x^2 - 2$ AND $y \leqslant -x$) OR ($y \geqslant x^2 - 2$ AND $y \leqslant x$).

```
var x, y: real;
begin
readln(x,y);
if ((y >= x * x - 2) and (y <= -x)) or ((y >= x * x - 2) and (y <= x)) then
```

```
write('принадлежит')
else
write('не принадлежит')
end.
```

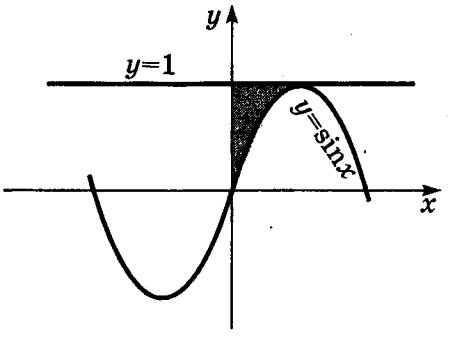
Доработка, приведённая в качестве примера правильного ответа в демонстрационном варианте ЕГЭ:

```
Возможная доработка (Паскаль, разбиение области на две части прямой x = 0):
if (y >= x * x - 2) and (y <= x) and (x >= 0) or (x <= 0) and (y <= -x)
                                                and (y >= x * x - 2) then
write('принадлежит')
else
write('не принадлежит')
```

гораздо менее рациональна, чем сформулированная выше, поскольку разработчики заданий ЕГЭ в своей записи использовали ещё одну границу при разбиении «подобластей» — ось Y, которая записывается уравнением $x = 0$.

Задача 2*. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы).

Программист торопился и написал программу неправильно.



Программа на Паскале	Программа на Бейсике	Программа на Си

var x, y: real;
begin
readln(x,y);
if y <= 1 then
if x >= 0 then
if y >= sin(x) then
write('принадлежит')
else
write('не принадлежит')
end.
	INPUT x, y
	IF y <= 1 THEN
	IF x >= 0 THEN
	IF y >= SIN(x) THEN
	PRINT "принадлежит"
	ELSE
	PRINT "не принадлежит"
	ENDIF
	ENDIF
	ENDIF
	END
		void main(void)
		{ float x, y;
		scanf("%f%f",&x,&y);
		if (y <= 1)
		if (x >= 0)
		if (y >= sin(x))
		printf("принадлежит");
		else
		printf("не принадлежит");
		}

Последовательно выполните следующее:

1. Приведите пример таких чисел x, y, при которых программа неправильно решает поставленную задачу.

2. Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой правильный способ доработки исходной программы.)

Решение

1. Область образована пересечением полуплоскостей:
- ниже прямой $y = 1$;
- правее прямой (оси Y) $x = 0$;
- выше синусоиды $y = \sin(x)$;
- левее вертикали $x = \pi/2 \approx 1{,}57$ (эта прямая на чертеже не показана, поэтому о необходимости добавления этого условия часто забывают; однако если этого не сделать, то решение будет ошибочным, так как в него войдут и все другие такие же области над синусоидой и под горизонтальной прямой $y = 1$, расположенные правее указанной области, — ведь, как нетрудно вспомнить, функция $\sin(x)$ имеет периодический характер).

2. Запись условия, определяющего принадлежность точки этой области:
$y \geqslant \sin(x)$ AND $y \leqslant 1$ AND $x \geqslant 0$ AND $x \leqslant \pi/2$

3. Запись условия, запрограммированного в приведённой в условии задачи программе:
$y \leqslant 1$ AND $x \geqslant 0$ AND $y \geqslant \sin(x)$

Очевидно, что ошибка в программе заключается в неучтённом ограничении «левее вертикали $x = \pi/2 \approx 1{,}57$».

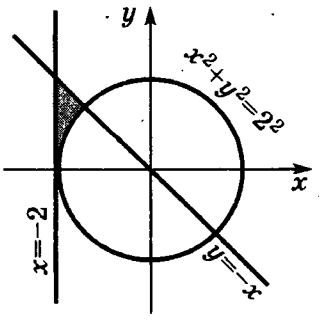
4. Пример точки, для которой программа даёт ошибочный ответ (ошибочно относит эту точку к заданной области): $(\pi, 1)$ — точка, лежащая на горизонтальной прямой $y = 1$. Пригодна в качестве ответа и точка (3, 0.5), приведённая в качестве примера правильного ответа в демо-варианте ЕГЭ.

5. Пример исправления программы:

```
var x, y: real;
begin
readln(x, y);
if (y >= sin(x)) and (y <= 1) and (x >= 0) and (x <= Pi / 2) then
write('принадлежит')
else
write('не принадлежит')
end.
```

Задача 3*. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы).

Программист торопился и написал программу неправильно.



Программа на Паскале	Программа на Бейсике	Программа на Си

var x, y: real;
begin
readln(x,y);
if x * x + y * y >= 4 then
if x >= -2 then
if y <= -x then
write('принадлежит')
else
write('не принадлежит')
end.
	INPUT x, y
	IF x * x + y * y >= 4 THEN
	IF x >= -2 THEN
	IF y <= -x THEN
	PRINT "принадлежит"
	ELSE
	PRINT "не принадлежит"
	ENDIF
	ENDIF
	ENDIF
	END
		void main(void)
		{ float x, y;
		scanf("% f % f",&x,&y);
		if (x * x + y * y >= 4)
		if (x >= -2)
		if (y <= -x)
		printf("принадлежит");
		else
		printf("не принадле-
		жит");
		}

Последовательно выполните следующее:

1) Приведите пример таких чисел x, y, при которых программа неправильно решает поставленную задачу.

2) Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой правильный способ доработки исходной программы.)

Решение

1. Область образована пересечением полуплоскостей:
- ниже прямой $y = -x$;
- правее прямой $x = -2$;
- вне (т. е. «выше») окружности $x^2 + y^2 = 2^2$;
- выше оси X (т. е. прямой $y = 0$).

2. Запись условия, определяющего принадлежность точки этой области:
$y \leqslant -x$ **AND** $x \geqslant -2$ **AND** $x * x + y * y \geqslant 4$ **AND** $y \geqslant 0$.

3. Запись условия, запрограммированного в приведённой в условии задачи программе:
$x * x + y * y \geqslant 4$ **AND** $x \geqslant -2$ **AND** $y \leqslant -x$.

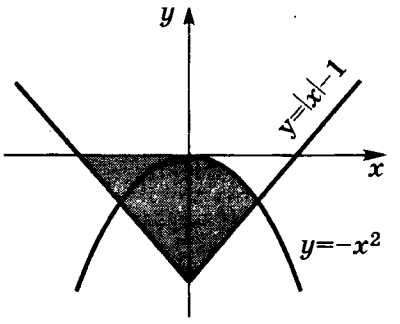
Очевидно, что ошибка в программе заключается в неучтённом ограничении «выше оси X».

4. Пример точки, для которой программа даёт ошибочный ответ (ошибочно относит эту точку к заданной области): (0, −3) — точка, лежащая вне окружности, правее прямой $x = -2$, ниже прямой $y = -x$, но не попадающей в указанную область. Пригодна в качестве ответа и точка (−1, −3), приведённая в качестве примера правильного ответа в демо-варианте ЕГЭ.

5. Пример исправления программы:

```
var x, y: real;
begin
readln(x, y);
if (y <= -x) and (x >= -2) and (x * x + y * y >= 4) and (y >= 0) then
write('принадлежит')
else
write('не принадлежит')
end.
```

Задача 4*. Требовалось написать программу, которая для введённых значений координат x, y точки плоскости (действительные числа) определяет принадлежность этой точки заданной области (включая её границы). Программа содержит ошибку.



Паскаль

```
var x, y: real;
begin
  readln(x, y);
  if y >= abs(x) - 1 then
   if y <= 0 then
    if y <= -x * x then
     write(' принадлежит ')
     else
write(' не принадлежит ')
end.
```

Бейсик

```
INPUT x, y
IF y >= abs(x) - 1 THEN
  IF y <= 0 THEN
    IF y <= -x * x THEN
      PRINT "принадлежит"
        ELSE PRINT "не принадлежит"
      ENDIF
    ENDIF
  ENDIF
END
```

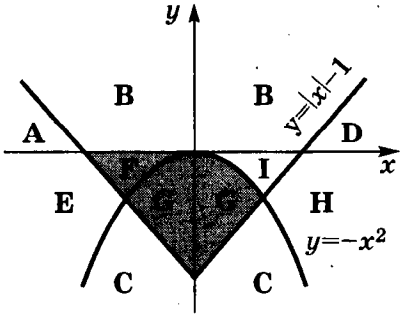
Си

```
void main(void)
{ float x, y;
  scanf ("%f%f ", &x, &y);
  if (y >= fabs(x) - 1)
   if (y <= 0)
    if (y <= -x * x)
     printf("принадлежит");
  else
printf("не принадлежит");
```

Алгоритмический язык

```
алг
нач
  вещ x, y
  ввод x, y
  если y >= abs(x) - 1 то
    если y <= 0 то
      если y <= -x * x то
        вывод ' принадлежит'
      иначе
        вывод 'не принадлежит'
    все
  все
кон
```

1. Заполните таблицу, которая показывает, как программа обрабатывает точки из различных областей (A, B, C, D, E, F, G, H и I).



Область	Условие 1 (y >= abs(x) − 1)	Условие 2 (y<=0)	Условие 3 (y <= −x*x)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					
G					
H					
I					

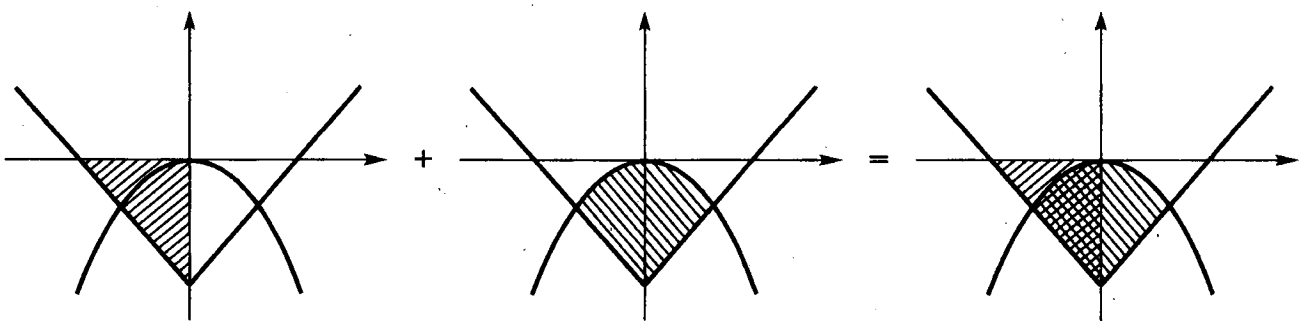
В столбцах условий записывайте «да», если условие выполняется, «нет», если условие не выполняется, «—» (прочерк), если условие не проверяется, «неизв.», если программа ведёт себя по-разному для разных значений, принадлежащих данной области. В столбце «Программа выведет» запишите, что программа выведет на экран. Если ничего не выводится, то запишите «—» (прочерк). Если для разных значений, принадлежащих области, будут выводиться разные ответы, напишите «неизв». В последнем столбце запишите «да» или «нет».

2. Укажите, как доработать программу для её правильного функционирования (достаточно указать любой способ доработки исходной программы).

Решение

Эта новая модификация задач С1, впервые появившаяся в тренажёрных работах в формате ЕГЭ в 2011/12 учебном году, решается практически полностью аналогично предыдущим. Заполнение же таблицы не только не усложняет сколько-нибудь заметно решение, но даже и помогает найти правильное условие для определения принадлежности точки указанной области. Необходим только навык внимательного заполнения таблицы.

1. Область можно определить как объединение двух подобластей:
— пересечение полуплоскостей:
• выше графика $y = |x| - 1$;
• левее оси $x = 0$;
• ниже оси y = 0,
— и пересечение полуплоскостей:
• выше графика $y = |x| - 1$;
• ниже графика $y = -x^2$.



Подобласть FG Подобласть G G Область FGG

2. Запись условия, определяющего принадлежность точки этой области.

1) Условие принадлежности точки подобласти FG:

(y >= abs(x) − 1) AND (y <= 0) AND (x <= 0).

2) Условие принадлежности точки подобласти GG:

(y >= abs(x) − 1) AND (y <= −x*x).

3) Условие принадлежности точки объединенной области:

((y >= abs(x) − 1) AND (y <= 0) AND (x <= 0)) OR ((y >= abs(x) − 1) AND (y <= −x*x)).

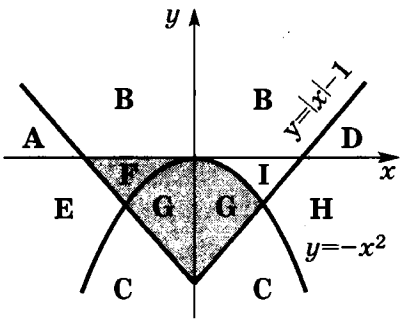
Используя законы алгебры логики, можно упростить это выражение, вынеся за скобки условие **(y >= abs(x) − 1)**:

(y >= abs(x) − 1) AND ((y <= 0) AND (x <= 0) OR (y <= −x*x)).

3. Заполнение таблицы производится построчно, глядя на рисунок с размеченными областями. При этом каждая строка заполняется слева направо, и если до какого-то условия выполнение программы просто не доходит, то в соответствующей ячейке ставится прочерк. В предпоследнем столбце записывается, что программа выведет на экран, — судя по содержимому срабатывающего оператора `write`. В последнем столбце надо записать своё заключение о том, соответствует ли выданное программой сообщение реальному положению дел.

Область	Условие 1 (y >= abs(x) − 1)	Условие 2 (y<=0)	Условие 3 (y <= −x*x)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					
G					
H					
I					

```
if y >= abs(x) - 1 then
   if y <= 0 then
     if y <= -x * x
       then write(' принадлежит ')
       else
write(' не принадлежит ')
```



1) *Область A.*

Условие `y >= abs(x) - 1` не выполняется (так как область A расположена ниже графика). Тогда в соответствующей ячейке пишется «нет»:

Область	Условие 1 (y >= abs(x) − 1)	Условие 2 (y<=0)	Условие 3 (y <= −x*x)	Программа выведет	Область обрабатывается верно
A	Нет				

Поскольку остальные условия размещены в ветви `then` первого оператора `if`, которая не будет срабатывать (ведь условие, записанное в первом операторе `if`, ложно), в остальных графах ставятся прочерки:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
A	Нет	—	—		

По правилам языка Паскаль, ветвь `else` принадлежит последнему оператору `if` в их цепочке. Поскольку он не выполняется, программа не выведет ничего (ставится прочерк в предпоследней ячейке):

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
A	Нет	—	—	—	

Правильно ли обрабатывается эта область? Очевидно, нет, так как программа должна была бы для нее вывести текст «не принадлежит». Поэтому в последней ячейке записывается «Нет»:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
A	Нет	—	—	—	Нет

2) *Область B.*

Условие `y >= abs(x) - 1` выполняется (область B расположена выше графика). Тогда в соответствующей ячейке пишется «да»:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
B	Да				

Срабатывает ветвь `then` первого оператора и проверяется второе условие — `y <= 0`. Оно не выполняется, поэтому во второй ячейке пишется «нет»:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
B	Да	Нет			

Во втором операторе `if` по этой причине (условие ложно) ветвь `then` не выполняется, поэтому проверка третьего условия не производится. Соответственно, в третьей графе ставится прочерк:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
B	Да	Нет	—		

Поскольку ветвь `else` принадлежит последнему оператору `if` в их цепочке, а он не выполняется, программа не выведет ничего (ставится прочерк в предпоследней ячейке):

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
B	Да	Нет	—	—	

Правильно ли обрабатывается эта область? Очевидно, нет, так как программа должна была бы для неё вывести текст «не принадлежит». Поэтому в последней ячейке записывается «Нет»:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
B	Да	Нет	—	—	Нет

3) *Область C.*

Аналогичным образом заполняется строка для области C, учитывая, что для неё не выполняется первое условие:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
C	Нет	—	—	—	Нет

4) *Область D.* Для неё также не выполняется первое условие:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
D	Нет	—	—	—	Нет

5) *Область E.* Первое условие не выполняется:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
E	Нет	—	—	—	Нет

6) *Область F.* Для неё выполняются первое и второе условия (в соответствующих ячейках записывается «Да»), но не выполняется третье (в третьей графе — «Нет»). Соответственно, срабатывает ветвь `else` третьего оператора `if`, и программа выводит «не принадлежит». Но это — неправильное определение, так как данная подобласть входит в заштрихованную область. Поэтому в последней ячейке записывается «Нет»:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
F	Да	Да	Нет	Не принадлежит	Нет

7) *Область G.* Для неё выполняются все три условия, программа выводит текст «принадлежит», и это правильно (в последней ячейке – «Да»):

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
G	Да	Да	Да	Принадлежит	Да

8) *Область H.* Для неё не выполняется первое условие:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
H	Нет	—	—	—	Нет

9) *Область I*. Для неё выполняются первое и второе условия, а третье не выполняется. Программа выводит текст «не принадлежит», и это правильно:

Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
I	Да	Да	Нет	Не принадлежит	Да

В результате заполненная таблица имеет вид:

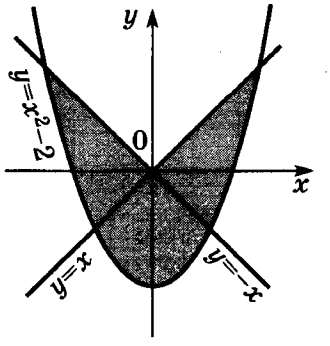
Область	Условие 1 (y >= abs(x) – 1)	Условие 2 (y<=0)	Условие 3 (y <= –x*x)	Программа выведет	Область обрабатывается верно
A	Нет	—	—	—	Нет
B	Да	Нет	—	—	Нет
C	Нет	—	—	—	Нет
D	Нет	—	—	—	Нет
E	Нет	—	—	—	Нет
F	Да	Да	Нет	Не принадлежит	Нет
G	Да	Да	Да	Принадлежит	Да
H	Нет	—	—	—	Нет
I	Да	Да	Нет	Не принадлежит	Да

4. В отличие от предыдущих задач, здесь не требуется указывать конкретную ошибку в записи условий в исходной программе и пример точки, которую программа обрабатывает неправильно. Поэтому можно текст программы не анализировать вовсе.

5. Пример исправления программы, как и раньше, можно получить, записав в одном операторе `if` ранее выведенное условие принадлежности точки заштрихованной области:

```
var x, y: real;
begin
readln(x, y);
if (y >= abs(x) - 1) and ((y <= 0) and (x <= 0) or (y <= -x * x)) then
write('принадлежит')
else
write('не принадлежит')
end.
```

Задача 5*. Требовалось написать программу, которая для введённых значений координат x, y точки плоскости (действительные числа) определяет принадлежность этой точки заданной области (включая её границы). Программа содержит ошибку.



Паскаль

```
var x, y: real;
begin
  readln(x, y);
  if y <= x then
   if y <= -x then
    if y >= x * x - 2 then
     write(' принадлежит ')
     else
write(' не принадлежит ')
end.
```

Бейсик

```
INPUT x, y
IF y <= x THEN
 IF y <= -x THEN
  IF y >= x * x - 2 THEN
   PRINT "принадлежит"
    ELSE
PRINT "не принадлежит"
    ENDIF
   ENDIF
  ENDIF
END
```

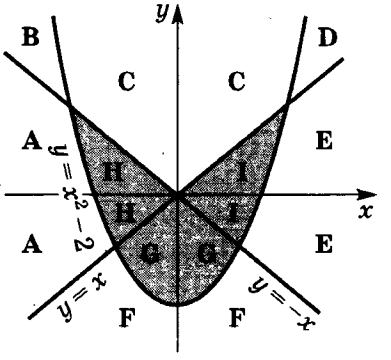
Си

```
void main(void)
{ float x, y;
  scanf ("%f%f ", &x, &y);
  if (y <= x)
   if (y <= -x)
    if (y >= x * x - 2)
     printf("принадлежит");
  else
printf("не принадлежит");
```

Алгоритмический язык

```
алг
нач
  вещ x, y
  ввод x, y
  если y <= x то
   если y <= -x то
    если y >= x * x - 2 то
     вывод ' принадлежит'
    иначе
     вывод 'не принадлежит'
   все
  все
кон
```

1. Заполните таблицу, которая показывает, как программа обрабатывает точки из различных областей (A, B, C, D, E, F, G, H и I).



Область	Условие 1 (y <= x)	Условие 2 (y <= −x)	Условие 3 (y >= x*x − 2)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					
G					
H					
I					

В столбцах условий укажите «да», если условие выполняется, «нет», если условие не выполняется, «—» (прочерк), если условие не проверяется, «неизв.», если программа ведёт себя по-разному для разных значений, принадлежащих данной области. В столбце «Программа выведет» запишите, что программа выведет на экран. Если ничего не выводится, запишите «—» (прочерк). Если для разных значений, принадлежащих области, выводятся разные ответы, запишите «не изв». В последнем столбце запишите «да» или «нет».

2. Укажите, как доработать программу для её правильного функционирования (достаточно указать любой способ доработки исходной программы).

Решение

Эта задача решается достаточно просто ещё и потому, что получение условия для формирования точно такой же закрашенной области было уже рассмотрено в задаче 1.

1. Область можно определить как объединение двух подобластей:

— пересечение полуплоскостей:

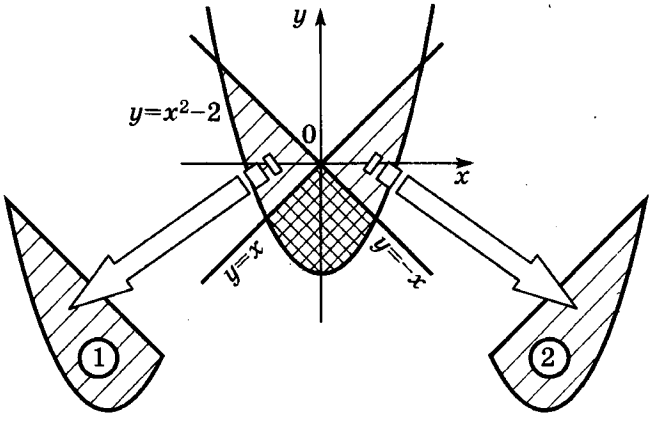
• выше параболы $y = x^2 - 2$;

• ниже прямой $y = -x$,

— и пересечение полуплоскостей:

• выше параболы $y = x^2 - 2$;

• ниже прямой $y \leqslant x$.



2. Запись условия, определяющего принадлежность точки этой области.

1) Условие принадлежности точки подобласти ①:

$(y \geqslant x^2 - 2)$ AND $(y \leqslant -x)$.

2) Условие принадлежности точки подобласти ②:

$(y \geqslant x^2 - 2)$ AND $(y \leqslant x)$.

3) Условие принадлежности точки объединенной области:

$((y \geqslant x^2 - 2)$ AND $(y \leqslant -x))$ OR $((y \geqslant x^2 - 2)$ AND $(y \leqslant x))$.

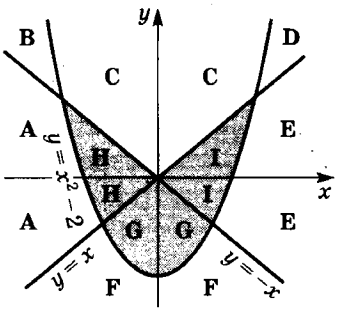
Используя законы алгебры логики, можно упростить это выражение, вынеся за скобки условие $(y \geqslant x^2 - 2)$:

$(y \geqslant x^2 - 2)$ AND $((y \leqslant -x)$ OR $(y \leqslant x))$.

3. Заполнение таблицы производится аналогично предыдущей задаче: для каждой строки поочередно проверяются условия; если какое-то из условий ложно, то последующие условия не срабатывают, а программа ничего не выдаёт на экран (ставятся прочерки); если же выдача сообщения на экран есть (что бывает, если срабатывает третье условие), то в последней графе записывается «вердикт» — правилен ли этот ответ программы.

Область	Условие 1 (y <= x)	Условие 2 (y <= −x)	Условие 3 (y >= x*x − 2)	Программа выведет	Область обрабатывается верно
A	Нет	—	—	—	Нет
B	Нет	—	—	—	Нет
C	Нет	—	—	—	Нет
D	Нет	—	—	—	Нет
E	Да	Нет	—	—	Нет
F	Да	Да	Нет	Не принадлежит	Да
G	Да	Да	Да	Принадлежит	Да
H	Нет	—	—	—	Нет
I	Да	Нет	—	—	Нет

```
if y <= x then
   if y <= -x then
    if y >= x * x - 2 then
     write(' принадлежит ')
     else
write(' не принадлежит ')
```

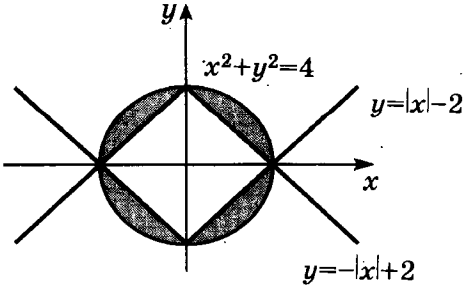


4. Пример исправления программы получается, если записать в одном операторе `if` ранее выведенное условие принадлежности точки заштрихованной области:

```
var x, y: real;
begin
readln(x,y);
if (y >= x * x - 2) and ((y <= -x) or (y <= x)) then
write('принадлежит')
else
write('не принадлежит')
end.
```

Задачи для самостоятельного решения

1. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы). Программист торопился и написал программу неправильно.



Программа на Паскале	Программа на Бейсике	Программа на Си

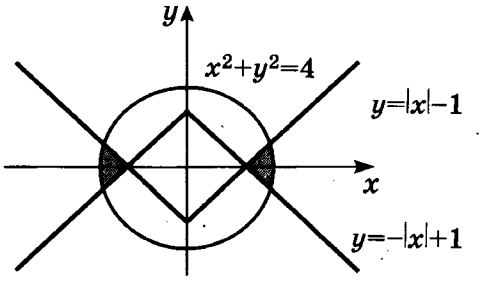
var x, y: real;
begin
readln(x, y);
if x * x + y * y <= 4 then
if y >= -abs(x) + 2 then
if y <= abs(x) - 2 then
write('принадлежит')
else
write('не принадлежит')
end.
	INPUT x, y
	IF x * x + y * y <= 4
	THEN
	IF y >= -abs(x) + 2 THEN
	IF y <= abs(x) - 2 THEN
	PRINT "принадлежит"
	ELSE
	PRINT "не принадлежит"
	ENDIF
	ENDIF
	ENDIF
	END
		void main(void)
		{float x, y;
		scanf("%f%f",&x,&y);
		if (x * x + y * y <= 4)
		if (y >= -abs(x) + 2)
		if (y <= abs(x) - 2)
		printf("принадлежит");
		else
		printf("не
		принадлежит");
		}

Последовательно выполните следующее:

1. Приведите пример таких чисел x, y, при которых программа неправильно решает поставленную задачу.

2. Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой правильный способ доработки исходной программы.)

2. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы). Программист торопился и написал программу неправильно.



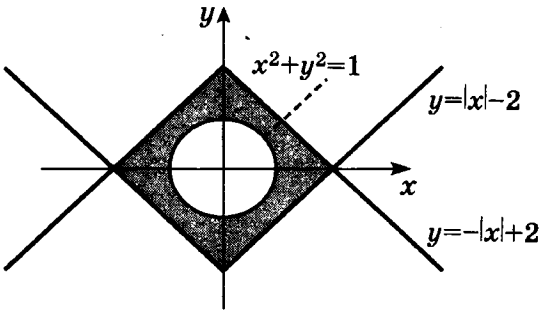
Программа на Паскале	Программа на Бейсике	Программа на Си

var x, y: real;
begin
readln(x, y);
if x * x + y * y <= 4 then
if y >= -abs(x) + 1 then
if y <= abs(x) - 1 then
write('принадлежит')
else
write('не принадлежит')
end.
	INPUT x, y
	IF x * x + y * y <= 4
	THEN
	IF y <= -abs(x) + 1 THEN
	IF y >= abs(x) - 1 THEN
	PRINT "принадлежит"
	ELSE
	PRINT "не принадлежит"
	ENDIF
	ENDIF
	ENDIF
	END
		void main(void)
		{float x, y;
		scanf("%f%f",&x,&y);
		if (x * x + y * y <= 4)
		if (y <= -abs(x) + 1)
		if (y >= abs(x) - 1)
		printf("принадлежит");
		else
		printf("не
		принадлежит");
		}

Последовательно выполните следующее:

1. Приведите пример таких чисел x, y, при которых программа неправильно решает поставленную задачу.
2. Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой правильный способ доработки исходной программы.)

3. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной закрашенной области (включая границы). Программист торопился и написал программу неправильно.



Паскаль

```
var x, y: real;
begin
readln(x, y);
if x * x + y * y >= 1 then
if y <= -abs(x) + 2 then
if y >= abs(x) - 2 then
write('принадлежит')
else
write('не принадлежит')
end.
```

Бейсик

```
INPUT x, y
IF x * x + y * y >= 1 THEN
  IF y <= -abs(x) + 2 THEN
   IF y >= abs(x) - 2 THEN
    PRINT "принадлежит"
      ELSE
PRINT "не принадлежит"
    ENDIF
   ENDIF
  ENDIF
END
```

Си

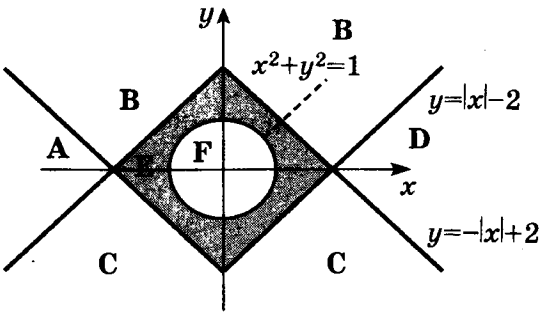
```
void main(void)
{ float x, y;
  scanf ("%f%f ", &x, &y);
  if (x * x + y * y >= 1)
   if (y <= -abs(x) + 2)
    if (y >= abs(x) - 2)
     printf("принадлежит");
  else
printf("не принадлежит");
```

Алгоритмический язык

```
алг
нач
 вещ x, y
 ввод x, y
 если x * x + y * y >= 1 то
  если y <= -abs(x) + 2 то
   если y >= abs(x) - 2 то
    вывод 'принадлежит'
   иначе
    вывод 'не принадлежит'
  все
 все
кон
```

Последовательно выполните следующее:

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F, G, H и I).

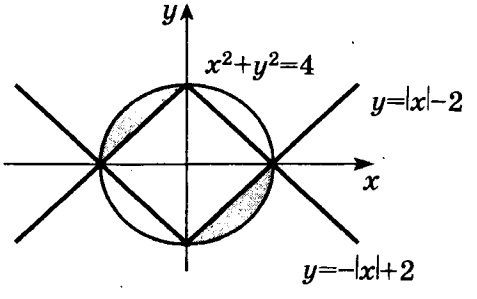


Область	Условие 1 (x*x+y*x>=1)	Условие 2 (y<=-abs(x)+2)	Условие 3 (y>=abs(x)-2)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					

В столбцах условий укажите «да», если условие выполнится, «нет», если условие не выполнится, «—» (прочерк), если условие не будет проверяться, «неизв.», если программа ведёт себя по-разному для разных значений, принадлежащих данной области. В столбце «Программа выведет» укажите, что программа выведет на экран. Если программа ничего не выводит, запишите «—» (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «неизв». В последнем столбце укажите «да» или «нет».

2. Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, достаточно указать любой способ доработки исходной программы.)

4. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной закрашенной области (включая границы). Программист торопился и написал программу неправильно.



Паскаль

```
var x, y: real;
begin
readln(x, y);
if x * x + y * y <= 4 then
if y >= -abs(x) + 2 then
if y <= abs(x) - 2 then
write('принадлежит')
else
write('не принадлежит')
end.
```

Бейсик

```
INPUT x, y
IF x * x + y * y <= 4 THEN
 IF y >= -abs(x) + 2 THEN
  IF y <= abs(x) - 2 THEN
   PRINT "принадлежит"
    ELSE
PRINT "не принадлежит"
   ENDIF
  ENDIF
 ENDIF
END
```

Си

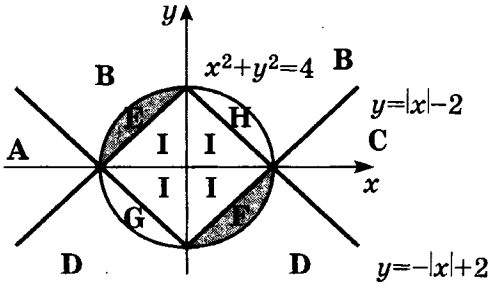
```
void main(void)
{ float x, y;
 scanf ("%f%f ", &x, &y);
 if (x * x + y * y <= 4)
  if (y >= -abs(x) + 2)
   if (y <= abs(x) - 2)
    printf("принадлежит");
 else
printf("не принадлежит");
```

Алгоритмический язык

```
алг
нач
 вещ x, y
 ввод x, y
 если x * x + y * y <= 4 то
  если y >= -abs(x) + 2 то
   если y <= abs(x) - 2 то
   вывод ' принадлежит'
  иначе
   вывод 'не принадлежит'
 все
 все
кон
```

Последовательно выполните следующее:

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F, G, H и I).

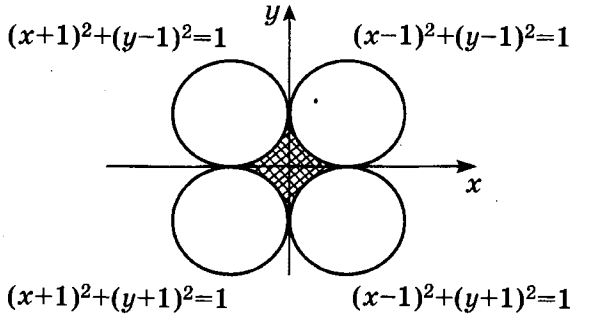


Область	Условие 1 (x*x+y*y<=4)	Условие 2 (y>=−abs(x)+2)	Условие 3 (y<=abs(x)−2)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					
G					
H					
I					

В столбцах условий укажите «да», если условие выполнится, «нет», если условие не выполнится, «—» (прочерк), если условие не будет проверяться, «неизв.», если программа ведёт себя по-разному для разных значений, принадлежащих данной области. В столбце «Программа выведет» укажите, что программа выведет на экран. Если программа ничего не выводит, запишите «—» (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «неизв». В последнем столбце укажите «да» или «нет».

2. Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, достаточно указать любой способ доработки исходной программы.)

. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной закрашенной области (включая границы). Программист торопился и написал программу неправильно.

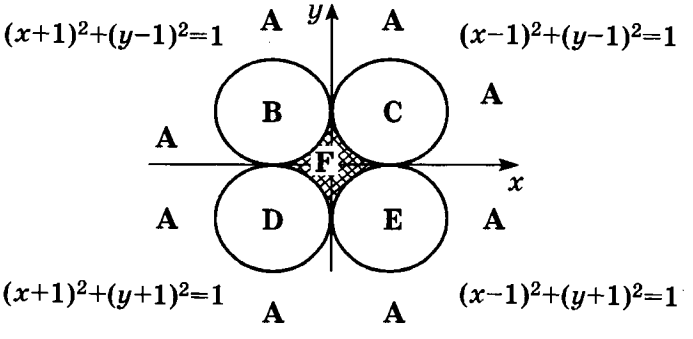


Программа на языке Паскаль:

```
var x, y: real;
begin
readln(x, y);
if (x - 1) * (x - 1) + (y - 1) * (y - 1) >= 1 then
if (x + 1) * (x + 1) + (y - 1) * (y - 1) >= 1 then
if (x + 1) * (x + 1) + (y + 1) * (y + 1) >= 1 then
if (x - 1) * (x - 1) + (y + 1) * (y + 1) >= 1 then
write('принадлежит')
else
write('не принадлежит')
end.
```

Последовательно выполните следующее:

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F, G, H и I).



Область	Условие 1 (x−1)*(x−1)+ +(y−1)*(y−1)>= >=1)	Условие 2 (x+1)*(x+1)+ +(y−1)*(y−1)>= >=1)	Условие 3 (x+1)*(x+1)+ +(y+1)*(y+1)>= >=1)	Условие 4 (x−1)*(x−1)+ +(y+1)*(y+1)>= >=1)	Программа выведет	Область обрабатывается верно
A	Да	Да	Да	Да	Принадлежит	Нет
B	Да	Нет	—	—	—	Нет
C	Нет	—	—	—	—	Нет
D	Да	Да	Нет	—	—	Нет
E	Да	Да	Да	Нет	Не принадлежит	Да
F	Да	Да	Да	Да	Принадлежит	Да

В столбцах условий укажите «да», если условие выполнится, «нет», если условие не выполнится, «—» (прочерк), если условие не будет проверяться, «неизв.», если программа ведёт себя по-разному для разных значений, принадлежащих данной области. В столбце «Программа выведет» укажите, что программа выведет на экран. Если программа ничего не выводит, запишите «—» (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «неизв». В последнем столбце укажите «да» или «нет».

2. Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, достаточно указать любой способ доработки исходной программы.)

Ответы для самопроверки

Задача 1.

1) ошибочная пара (x, y): (0,2), а также любые значения, соответствующие точкам верхней координатной полуплоскости без учёта x = 0;

2) возможная доработка программы:

```
var x, y: real;
begin
readln(x, y);
if ((y >= -abs(x) + 2) or (y <= abs(x) - 2)) and (x * x + y * y <= 4) then
 write('принадлежит')
else
 write('не принадлежит')
end.
```

Задача 2.

1) ошибочная пара (x, y): (0,0), а также любые значения (x, y), соответствующие точка вне окружности или внутри нее и под графиком $y = -|x| + 1$;

2) возможная доработка программы:

```
var x, y: real;
begin
readln(x, y);
if (x * x + y * y <= 4) and (y >= -abs(x) + 1) and (y <= abs(x) - 1) then
   write('принадлежит')
else
   write('не принадлежит')
end.
```

Задача 3.

1)

Область	Условие 1 (x*x+y*y>=1)	Условие 2 (y<=-abs(x)+2)	Условие 3 (y>=abs(x)-2)	Программа выведет	Область обрабатывается верно
A	Да	Нет	—	—	Нет
B	Да	Нет	—	—	Нет
C	Да	Да	Нет	Не принадлежит	Да
D	Да	Нет	—	—	Нет
E	Да	Да	Да	Принадлежит	Да
F	Нет	—	—	—	Нет

2)

```
var x, y: real;
begin
readln(x, y);
if (x * x + y * y >= 1) and (y <= -abs(x) + 2) and (y >= abs(x) - 2) then
write('принадлежит')
else
write('не принадлежит')
end.
```

Задача 4.

1)

Область	Условие 1 (x*x+y*y<=4)	Условие 2 (y>=-abs(x)+2)	Условие 3 (y<=abs(x)-2)	Программа выведет	Область обрабатывается верно
A	Нет	—	—	—	Нет
B	Нет	—	—	—	Нет
C	Нет	—	—	—	Нет
D	Нет	—	—	—	Нет
E	Да	Да	Нет	Не принадлежит	Нет
F	Да	Нет	—	—	Нет
G	Да	Нет	—	—	Нет
H	Да	Да	Нет	Не принадлежит	Да
I	Да	Нет	—	—	Нет

2)

```
var x, y: real;
begin
readln(x, y);
if ((x * x + y *y <= 4) and (y >= -abs(x) + 2) and (x <= 0)) or
   ((x * 2 + y * 2 <= 4) and y <= abs(x) - 2) and (x >= 0)) then
write('принадлежит')
else
write('не принадлежит')
end.
```

или

```
var x, y: real;
begin
readln(x, y);
if (x * x + y * y <= 4) and ((y >= -abs(x) + 2) and (x <= 0) or
   (y <= abs(x) - 2) and (x >= 0)) then
write('принадлежит')
else
write('не принадлежит')
end.
```

Задача 5.

1)

Область	Условие 1 (x−1)*(x−1)+ +(y−1)*(y−1)>= >=1)	Условие 2 (x+1)*(x+1)+ +(y−1)*(y−1)>= >=1)	Условие 3 (x+1)*(x+1)+ +(y+1)*(y+1)>= >=1)	Условие 4 (x−1)*(x−1)+ +(y+1)*(y+1)>= >=1)	Программа выведет	Область обрабатывается верно
A	Да	Да	Да	Да	Принадлежит	Нет
B	Да	Нет	—	—	—	Нет
C	Нет	—	—	—	—	Нет
D	Да	Да	Нет	—	—	Нет
E	Да	Да	Да	Нет	Не принадлежит	Да
F	Да	Да	Да	Да	Принадлежит	Да

2)

```
var x, y: real;
begin
readln(x, y);
if ((x - 1) * (x - 1) + (y - 1) * (y - 1) >= 1) and
   ((x + 1) * (x + 1) + (y - 1) * (y - 1) >= 1) and
   ((x + 1) * (x + 1) + (y + 1) * (y + 1) >= 1) and
   ((x - 1) * (x - 1) + (y + 1) * (y + 1) >= 1) and
   (x >= -1) and (x <= 1) and (y >= -1) and (y <= 1) then
  write('принадлежит')
else
  write('не принадлежит')
end.
```

или

```
var x, y: real;
begin
readln(x, y);
if ((x - 1) * (x - 1) + (y - 1) * (y - 1) >= 1) and
   ((x + 1) * (x + 1) + (y - 1) * (y - 1) >= 1) and
   ((x + 1) * (x + 1) + (y + 1) * (y + 1) >= 1) and
   ((x - 1) * (x - 1) + (y + 1) * (y + 1) >= 1) and
   (abs(x) <= 1) and (abs(y) <=1 ) then
  write('принадлежит')
else
  write('не принадлежит')
end.
```

или

```
var x, y: real;
begin
readln(x, y);
if ((x - 1) * (x - 1) + (y - 1) * (y - 1) >= 1) and
   ((x + 1) * (x + 1) + (y - 1) * (y - 1) >= 1) and
   ((x + 1) * (x + 1) + (y + 1) * (y + 1) >= 1) and
   ((x - 1) * (x - 1) + (y + 1) * (y + 1) >= 1) and
   (x * x + y * y <= 1) then
  write('принадлежит')
else
  write('не принадлежит')
end.
```

Программирование

C4 Задачи на анализ и обработку данных

Конспект

Решение задач типа C4 требует не только умения применять типовые алгоритмы обработки данных (поиск минимума/максимума, вычисление среднего значения, определение количества элементов массива, удовлетворяющих заданному условию, сортировка и пр.), но и умений анализировать условие задачи, выделять «ключевые» данные (и определять, какие данные являются излишними и не требуют запоминания), применять нестандартные приёмы ввода данных, формировать индексные массивы и массивы-счётчики и т. д.

Данное занятие посвящено разбору основных приёмов программирования, используемых при решении подобных задач.

Пропуск текстовых данных из потока символов

Данные на входе программы могут представлять собой строку, содержащую несколько чисел или строковых констант, разделённых пробелом.

При чтении данных из такой входной строки оператором read (readln) чтение данных должно производиться слева направо с начала и до конца.

Если какие-то данные не нужны для работы программы, то для их пропуска можно использовать следующие приемы:

а) для пропуска числовых данных — считать ненужные данные в отдельно объявленную «буферную» переменную соответствующего типа, которая в программе далее не используется (при чтении следующего ненужного данного используется та же переменная).

Пример:

```
...
// входная строка: <день> <месяц> <год>, где день и месяц не нужны
var Year: integer;   // год
    Musor: integer; // переменная для чтения ненужных данных
...
read(Musor);  // считали день
read(Musor);  // считали в ту же переменную месяц
readln(Year); // считали год, переменная Year идет на обработку
```

б) для пропуска текстовых данных — поскольку их длина заранее не известна, а считывание в переменную типа string загружает в неё всю входную строку, нужно в цикле считывать из входной строки отдельные символы, включая пробел, завершающий очередное данное. Для пропуска ещё одного такого данного повторно строится такой же цикл.

Пример:

```
...
// входная строка: <фамилия><имя><возраст>, где фамилия и имя не нужны
var Age: integer; // возраст
  Musor: char ;   // переменная для чтения ненужного символа
...
{Пропуск фамилии}
repeat
read(Musor)
until Musor=' ';
// считывается очередной ненужный символ, пока не окажется,
// что был считан пробел

{Пропуск имени}
repeat
  read(Musor)
until Musor=' ';
// считывается очередной ненужный символ, пока не окажется,
// что был считан пробел

{Чтение возраста}
readln(Age); // считали возраст, переменная Age идет на обработку
```

Массивы-счётчики

При определении количества элементов массива, соответствующих заданному условию (например, количества чётных элементов или количества элементов, равных максимальному), используется переменная-счётчик, которая увеличивает своё значение на 1 при обнаружении подходящего элемента.

При необходимости подсчёта количеств нескольких объектов используется *массив-счетчик* размера, соответствующего количеству объектов. При обнаружении в потоке входных данных объекта, соответствующего условию, нужно увеличить на 1 значение ячейки массива-счётчика, соответствующей данному объекту.

Пример.

Для каждого из 8 филиалов фирмы (пронумерованных от 1 до 8) вразброс вводятся записи из трёх чисел:

<номер филиала> <номер месяца> <прибыль/убыток>,

где прибыль выражена положительной суммой в рублях, а убыток — отрицательной. Завершение входных данных — номер филиала, равный нулю.

Требуется для каждого филиала подсчитать количество месяцев с убытком.

Решение

1. Выделяется массив-счётчик. Количество филиалов равно 8, поэтому массив-счётчик должен состоять из 8 элементов с индексами от 1 до 8:

```
var Kol: array[1..8] of integer;
```

2. Считывается очередная строка данных (три числа в переменные Filial, Mes и Summ).

3. Строится цикл с предусловием:

```
while (Filial <> 0) do begin
```

4. В цикле обрабатывается водные данные: если обнаружен убыток, т. е. Summ < 0, то значение ячейки массива-счётчика, соответствующей номеру филиала, увеличивается на 1.

```
if (Summ < 0) then Kol[Filial] := Kol[Filial] + 1;
```

5. Считывается очередная порция данных (три числа в переменные Filial, Mes и Summ) — если номер филиала не равен нулю, то выполняется очередной проход цикла.


Такое построение цикла позволяет отследить ситуацию, когда входных данных нет (т. е. сразу вводится завершающая строка с нулевым номером филиала).

6. После завершения цикла массив-счётчик, содержащий количества убыточных месяцев для каждого филиала, может быть дополнительно обработан (например, может быть выполнен поиск номера филиала — номера ячейки этого массива, имеющего максимальное количество убыточных месяцев.


При формировании массива-счётчика нужно не забывать, что в качестве индексов его элементов в Паскале могут использоваться не только произвольные диапазоны целых чисел, но и диапазоны символов, например букв алфавита.

Индексные массивы

Возможна ситуация, когда в качестве индексов массива-счётчика невозможно (или не рационально) использовать значения, считываемые из входных данных в качестве идентификатора объектов, например:

- если вводятся словесные названия месяцев, для которых нужно выполнять подсчёт (строковые данные нельзя использовать как индексы элементов массива);
- если номера объектов идут не подряд. Скажем, известно, что вводятся номера школ, для которых обрабатываются какие-то данные: 215, 386, 512, 1167, 3160, тогда при создании массива-счётчика с полным диапазоном индексов от 215 до 3160 подавляющее большинство его элементов, кроме пяти с перечисленными индексами, будут не использованы.

В этих случаях применяется *индексный массив*:

1) массив-счётчик объявляется по количеству обрабатываемых объектов (например, в случае с месяцами — на 12 элементов, в случае со школами — на 5 элементов);

2) отдельно объявляется массив того же размера, который содержит обозначения соответствующих объектов (для месяцев — их словесные названия, для школ — их номера);

3) при считывании входных данных соответствующее обозначение объекта ищется в индексном массиве (путём его просмотра с начала до конца); номер ячейки индексного массива, содержащей искомое обозначение объекта, запоминается;

4) если данный объект соответствует требуемому условию, то массив-счётчик увеличивается на 1 элемент, индекс которого равен запомненному индексу элемента в индексном массиве, содержащем обозначение этого объекта.


Связь индексного массива и массива-счётчика осуществляется через общий индекс элемента. Если при выводе полученных данных требуется выводить обозначения объектов, то они берутся из индексного массива.

Разбор типовых задач

Задача 1. Входные данные представляют собой последовательность строк — записей из базы данных студентов вуза вида:
<идентификационный номер> <фамилия> <имя> <отчество> <номер группы> <возраст>, где данные разделены одним пробелом. Завершение ввода — строка, в которой идентификационный номер равен нулю.

Требуется подсчитать количество студентов старше 25 лет. Если таких нет, то вывести сообщение: «Искомых студентов нет».

Решение

1. В данном случае в качестве счётчика может использоваться отдельная переменная:

```
var Kol: integer;
```

2. При вводе нужно вначале отдельно считывать идентификационный номер и проверять его на равенство нулю (условие завершения ввода данных). Для этого используем цикл с предусловием:

```
read(ID);
while (ID <> 0) do begin
   ...
   read(ID);
end;
```

3. Если идентификационный номер не равен нулю, надо пропускать четыре первых данных (<фамилия> <имя> <отчество> <номер группы>). Для этого можно использовать как буферную переменную (для идентификационного номера и для индекса группы), так и посимвольное чтение до пробела включительно. Последний вариант является более универсальным. Для четырёхкратного повторения этой операции можно использовать цикл с параметром:

```
for i := 1 to 4 do begin
   repeat
    read(c)
   until c = ' ';
end;
```

4. Последнее данное в каждой введенной строке считывается оператором readln (для перехода к следующей строке входных данных):

```
readln(Age);
```

5. Значение переменной Age проверяется и если оно больше 25, то значение переменной-счетчика увеличивается на 1:

```
if Age > 25 then Kol := Kol + 1;
```

Полный текст программы:

```
Program Age_Students;
var Kol: integer;
    ID: integer;
    Age: integer;
    i: integer;
    c: char;
begin
   read(ID);
   while (ID <> 0) do begin
      for i := 1 to 4 do begin
         repeat
      read(c)
     until c = ' ';
  end;
  readln(Age);
  if Age > 25 then Kol := Kol + 1;
  read(ID);
end;
```

```
if Kol = 0 then writeln('Искомых студентов нет')
           else writeln('Студентов старше 25 лет: ', Kol);
end.
```

Задача 2. Входные данные программы — подаётся строка английского текста. Завершение ввода — символ «точка».

Требуется подсчитать количества вхождений в нее каждой буквы от A до Z (строчные и прописные буквы не различаются) и вывести эти данные на экран (для букв, которые не встречаются в строке, не выводить ничего).

Решение

1. Требуется подсчёт количества каждой из букв от A до Z. Поэтому определяем массив-счётчик с индексами элементов, обозначаемыми буквами от A до Z:

```
var Kol: array ['A'..'Z'] of integer;
```

2. Чтение входных данных — посимвольное, в цикле с предусловием, пока очередной прочитанный символ не будет являться точкой:

```
read(c);
while (c <> ".") do begin
  ...
  read(c);
end;
```

3. Обработка данных внутри цикла — проверка: если очередной символ является английской буквой, то:

- преобразовать эту букву в прописной регистр (функция UpperCase());
- увеличить на 1 значение элемента массива-счётчика, индекс которого равен полученному символу.

```
if (c >= 'A') and (c <= 'Z') or (c >= 'a') and (c <= 'z') then
  Kol[UpperCase(c)] := Kol[UpperCase(c)] + 1;
```

Полный текст программы:

```
Program Alfavit;
var Kol: array ['A'..'Z'] of integer;
    c: char;
begin
  // обнуление массива-счетчика
  for c := 'A' to 'Z' do Kol[c] := 0;
  // чтение и обработка данных
  read(c);
  while (c <> '.') do begin
    if (c >= 'A') and (c <= 'Z') or (c >= 'a') and (c <= 'z') then
     Kol[UpperCase(c)] := Kol[UpperCase(c)] + 1;
    read(c);
  end;
  // вывод результата
  for c := 'A' to 'Z' do
    if Kol[c] <> 0 then
     writeln('Буква ', c, ' встречается ', Kol[c], ' раз');
end.
```

Задача 3. Ученики школ города *N* сдавали ЕГЭ. Сведения о результатах сдачи каждого экзамена поступают в методический центр в виде наборов строк вида:

<Номер школы> <Фамилия> <Имя> <Предмет> <Оценка>,

где оценка выражена числом баллов от 1 до 5.

Известно, что в городе 100 школ и они имеют номера от 1100 до 1215. Всего на обработку поступило *X* записей, где *X* не превышает 1 миллион. При вводе данных вначале вводится количество записей, а затем сами записи.

Требуется подсчитать для каждой школы общее количество полученных её учениками неудовлетворительных оценок (единиц и двоек) за все экзамены. При выводе результата — пропускать школы, ученики которых не получили ни одной неудовлетворительной оценки.

Решение

1. Требуется подсчёт количеств неудовлетворительных оценок для каждой из школ с номерами от 1100 до 1215. Всего школ — 100 (т. е. некоторые номера не используются). Количество неиспользуемых номеров школ равно (1215 – 1100) – 100 = 15. Это количество невелико, поэтому использовать индексный массив нецелесообразно. Используется массив-счётчик, индексы элементов которого меняются соответственно номерам школ от 1100 до 1215.

Каким должен быть тип этого массива? Общее число обрабатываемых записей может составлять до 1 000 000. В наихудшем случае, если все ученики получили неудовлетворительные оценки и все эти ученики относятся к одной школе, максимальное количество подсчитанных оценок (т. е. максимальное значение соответствующего элемента массива) будет равно 1 000 000. Для хранения такого числа достаточно типа `integer`.

```
var Kol: array[1100..1215] of integer;
```

2. При вводе данных прежде всего считывается количество записей (переменная типа integer). Поэтому для чтения записей можно использовать цикл с параметром.

При чтении каждой из записей нужно:

- прочитать номер школы;
- пропустить фамилию, имя и название предмета;
- прочитать оценку.

```
read(X); // кол-во записей
for i := 1 to X do begin
  read(NoSch); // номер школы
  for j := 1 to 3 do begin // пропуск трех данных
    repeat
      read(c)
    until c = ' ';
  end;
  read(Ocenka);
  ...
end;
```

3. Проверяется: если оценка неудовлетворительная (равна 1 или 2), то увеличивается на 1 элемент массива-счётчика, индекс которого равен номеру школы.

```
if Ocenka < 3 then
  Kol[NoSch] := Kol[NoSch] + 1;
```

Если не гарантировано, что оценка вводится правильно, то нужно проверять оценку так:

```
if (Ocenka = 1) or (Ocenka = 2) then ...
```

Полный текст программы:

```
Program Ocenki;
var Kol: array[1100..1215] of integer;
    X, i, j, Ocenka, NoSch: integer;
    c: char;
begin
  read(X); // кол-во записей
  for i := 1 to X do begin
    read(NoSch); // номер школы
    for j := 1 to 3 do begin // пропуск трех данных
      repeat
        read(c)
      until c = ' ';
    end;
    read(Ocenka);
    if Ocenka < 3 then
    Kol[NoSch] := Kol[NoSch] + 1;
  end;
   for i := 1100 to 1215 do // просмотр массива-счетчика
   if Kol[i] <> 0 then
     writeln('Школа ', i, ' - ', Kol[i], ' учеников');
end.
```

Задача 4[1]. Ученики школ города *N* сдавали ЕГЭ. Сведения о результатах сдачи каждого экзамена поступают в методический центр в виде наборов строк вида:

<Номер школы> <Фамилия> <Имя> <Предмет> <Оценка>,

где оценка выражена числом баллов от 1 до 5.

Известно, что в городе 100 школ и они имеют номера от 1100 до 1215. Всего на обработку поступило *X* записей, где *X* не превышает 1 миллион. При вводе данных вначале вводится количество записей, а затем сами записи.

Требуется отдельно подсчитать для каждой школы количество полученных её учениками каждой из оценок за все экзамены. При выводе результата — пропускать школы, для которых нет записей, и пропускать записи об оценках, которых нет.

Решение

1. Аналогично предыдущей задаче, нужно использовать массив-счётчик, индексы элементов которого меняются соответственно номерам школ от 1100 до 1215. Однако теперь для каждой школы надо отдельно подсчитывать количества каждой из оценок (от 1 до 5). Поэтому требуется использовать двумерный массив-счётчик, первый индекс которого меняется от 1100 до 1215, а второй индекс — от 1 до 5. Тип массива — `integer`.

```
var Kol: array[1100..1215,1..5] of integer;
```

2. При вводе данных прежде всего считывается количество записей (переменная типа `integer`). Поэтому для чтения записей используется цикл с параметром.

При чтении каждой из записей нужно:

- прочитать номер школы;
- пропустить фамилию, имя и название предмета;
- прочитать оценку.

[1] Модификация предыдущей задачи.

```
read(X); // кол-во записей
for i := 1 to X do begin
  read(NoSch); // номер школы
  for j := 1 to 3 do begin // пропуск трех данных
    repeat
      read(c)
    until c = ' ';
  end;
  read(Ocenka);
  ...
end;
```

3. При обработке оценки увеличивается на 1 элемент массива-счётчика, первый индекс которого равен номеру школы, а второй индекс равен оценке.

```
Kol[NoSch,Ocenka] := Kol[NoSch,Ocenka] + 1;
```

4. Вывод результатов представляет собой достаточно сложную задачу из-за требования пропуска школ, для которых нет оценок. Проще всего использовать следующий критерий:

- **для очередной школы:**
- **подсчёт суммы количеств каждой оценки;**
- **если эта сумма не равна нулю, то:**
- **выводится номер школы;**
- **в цикле по оценкам выводится информация о каждой оценке, если её количество (элемент массива) не равно нулю;**
- **перевод строки (writeln);**

Полный текст программы:

```
Program Ocenki_1_5;
var Kol: array[1100..1215,1..5] of integer;
    X, i, j, Ocenka, NoSch, Sum: integer;
    c: char;
begin
  read(X); // кол-во записей
  for i := 1 to X do begin
    read(NoSch); // номер школы
    for j := 1 to 3 do begin // пропуск трех данных
      repeat
        read(c)
      until c = ' ';
    end;
    read(Ocenka);
    Kol[NoSch,Ocenka] := Kol[NoSch,Ocenka] + 1;
  end;
  for i := 1100 to 1215 do begin // просмотр массива-счетчика
    Sum := 0;
    for j := 1 to 5 do Sum := Sum + Kol[i,j];
    // если сумма оценок для данной школы равна нулю,
    // т.е. оценок для нее не задано, то для этой школы
    // ничего не выводим
    if Sum <> 0 then begin
     write('Школа № ', i, ':');
```

```
      for j := 1 to 5 do
        if Kol[i,j] <> 0 then
          write(' "',j,'" - ',Kol[i,j]);
        writeln;
      end;
    end;
end.
```

Задача 5. Автоматическое устройство в течение года ежедневно производит запись значения среднесуточной температуры, формируя строку вида:

<дата> <название месяца> <температура>

Всего на обработку поступило N таких строк (возможно — не все записанные устройством строки). На вход программы подаётся сначала число N в отдельной строке, а затем сами строки.

Требуется сформировать массив, содержащий для каждого месяца среднюю температуру за этот месяц, и вывести эти значения в формате:

<название месяца> – <средняя температура>

Если для какого-то месяца данных не было, то вывести для него строку вида:

«На <название месяца> данных нет.»

Решение

1. В данном случае требуется вычисление не просто количества, но средней температуры. При этом следует отдельно вычислять для каждого месяца сумму считанных значений температуры и количество таких замеров, а среднее значение можно вычислить только уже после ввода всех имеющихся записей.

Подсчёт ведётся для каждого из 12 месяцев, следовательно, массив-счётчик должен иметь размеры 12×2 (первый индекс — от 1 до 12 по количеству месяцев, второй индекс — от 1 до 2: первая ячейка — для подсчёта суммы температур, вторая — для подсчёта количества замеров). Температура может быть выражена вещественным числом, поэтому массив-счётчик должен иметь тип `real`.

Поскольку вводятся названия месяцев, т. е. строки, которые не могут использоваться в качестве индексов элементов массива, требуется использовать индексный массив из 12 элементов, которые содержат соответствующие их номерам названия месяцев:

```
var IndMas: array [1..12] of string[8] := ('январь', 'февраль', 'март',
    'апрель', 'май', 'июнь', 'июль', 'август', 'сентябрь', 'октябрь',
    'ноябрь', 'декабрь'); KolMas: array [1..12,1..2] of real;
```

2. Чтение строк производится при помощи цикла с параметром (предварительно считывается количество этих строк).

При чтении каждой строки нужно пропустить дату (число), прочитать название месяца (строка) и значение средней температуры за этот день (вещественное число). Для пропуска даты считываем её в «буферную» переменную, позже нигде не используемую.

Особую сложность представляет чтение названия месяца. Оператор `read(Mes)`, где `Mes` — строка длиной в 8 символов (по максимальной длине названия месяца) для этого непригоден, так как считывает из входной строки строку ровно из 8 символов, тогда как названия месяцев имеют разную длину; из-за этого `Mes` после считывания будет кроме названия месяца захватывать пробел и часть цифр из значения температуры. Нам же нужно выполнить чтение в переменную `Mes` символов названия месяца до первого встреченного пробела (не включая этот пробел), поэтому используем цикл с предусловием:

```
Mes := '';
// «обнуление» переменной (ей присваивается пустая строка)
```

```
read(c);
while (c <> ' ') do begin
  Mes := Mes + c;
  read(c);
end;
```

3. Обработка данных:

- **для каждой считанной строки:**
- **просмотром в цикле от 1 до 12 индексного массива ищется индекс ячейки, значение которой совпадает с введённым названием месяца; тем самым определяется и запоминается номер месяца (переменная** `NoMes`**);**
- **в массиве-счётчике к значению элемента** `KolMas[NoMes,1]` **прибавляется считанное значение температуры;**
- **в массиве-счётчике значение элемента** `KolMas[NoMes,2]` **увеличивается на 1;**
- **по завершении просмотра строк для каждого значения от 1 до 12 (т. е. для каждого месяца) в массиве** `KolMas` **выполняется вычисление средней температуры за этот месяц: накопленная в первой ячейке сумма температур делится на накопленное во второй ячейке количество замеров.**

```
KolMas[NoMes,1] := KolMas[NoMes,1]/KolMas[NoMes,2]
```

4. Вывод данных: для каждого номера месяца (от 1 до 12):

- **если количество замеров за данный месяц** `KolMas[NoMes,2]` **равно нулю, то выводится заданная фраза: «На <название месяца> данных нет.»;**
- **иначе выводится строка вида:** `<название месяца> – <средняя температура>`.

В обоих случаях название месяца извлекается по его номеру из индексного массива: `IndMas[<номер месяца>]`.

Полный текст программы:

```
Program Mes_T;
var IndMas: array [1..12] of string[8] := ('январь', 'февраль',
    'март', 'апрель', 'май', 'июнь', 'июль', 'август', 'сентябрь',
    'октябрь', 'ноябрь', 'декабрь');
    KolMas: array [1..12,1..2] of real;
    i, j, N, NoMes, Dat: integer;
    T: real;
    Mes: string[8];
    c: char;
begin
  read(N); // кол-во записей
  for i := 1 to N do begin
    read(Dat); // дата («буферная» переменная)
    Mes := '';
    // «обнуление» переменной (ей присваивается пустая строка)
    read(c);
    while (c <> ' ') do begin
      Mes := Mes + c;
      read(c);
    end;
    read(T);
// обработка данных
    for j := 1 to 12 do
    if IndMas[j] = Mes then begin NoMes := j; break; end;
    // NoMes содержит номер месяца
```

```
    KolMas[NoMes,1] := KolMas[NoMes,1] + T;
    KolMas[NoMes,2] := KolMas[NoMes,2] + 1;
    end;

// входные данные обработаны полностью —
// вычисление средних температур за каждый месяц
  for i := 1 to 12 do
    KolMas[i,1] := KolMas[i,1] / KolMas[i,2];
// вывод результатов
  for i := 1 to 12 do
    if KolMas[i,2] = 0 then
              writeln('На ',IndMas[i],' данных нет.')
    else
              writeln(IndMas[i],' — ',KolMas[i,1]);
end.
```

Задача 6. По школам района (№№ школ: 15, 181, 211, 213, 215, 367, 1115, 1125, 1560 и 1820) ежемесячно собирается информация о количестве участников и победителей олимпиад. При этом входные данные имеют вид:

<№ школы> <название месяца> <Фамилия> <Имя>
<название олимпиадного предмета> <статус>,

где статус обозначается словом «участник» или «победитель».

Завершение ввода — строка, в которой содержится нуль.

Требуется подсчитать для каждой школы количество победителей олимпиад за весь учётный период. Для школ, в которых нет победителей, информацию не выводить.

Решение

1. Имеются 10 школ с разбросом значений их номеров от 15 до 1820. Использовать номера школ в качестве индексов массива-счётчика нерационально, поэтому необходимо использовать индексный массив, содержащий номера школ. Размер индексного массива и массива-счетчика равен 10 элементам (по количеству школ).

```
var IndMas: array [1..10] of integer := (15, 181, 211, 213, 215, 367,
                                         1115, 1125, 1560, 1820);
    KolMas: array [1..10] of integer;
```

2. Чтение данных. Поскольку количество вводимых строк неизвестно, используется цикл с предусловием для проверки считанного значения номера школы на предмет равенства нулю (признак конца ввода).

Данные <название месяца> <Фамилия> <Имя> <название олимпиадного предмета> пропускаются.

Значение статуса ученика («участник»/«победитель») последнее по счёту в строке, поэтому можно выполнить простое чтение строковой переменной.

```
read(NoSch);
while (NoSch <> 0) do begin
  for i := 1 to 4 do // пропуск 4-х ненужных данных
    repeat
      read(c)
    until c = ' ';
  read(Status);
  ...
  read(NoSch);
end;
```

3. Обработка данных:

- просмотром в цикле от 1 до 10 индексного массива ищется индекс ячейки, значение которой совпадает с введённым номером школы; запоминается этот индекс (переменная NoNoSch);
- проверяется считанное значение статуса: если оно равно «победитель», то в массиве-счётчике значение элемента KolMas[NoNoSch] увеличивается на 1.

```
for i := 1 to 10 do
  if IndMas[i] = NoSch then begin NoNoSch := i; break; end;
// NoNoSch содержит номер месяца
if Status = 'победитель' then
  KolMas[NoNoSch] := KolMas[NoNoSch] + 1;
```

4. При выводе данных просматривается весь массив-счётчик, для его ненулевых ячеек выводится количество победителей, номер школы при этом берётся из индексного массива.

Полный текст программы:

```
Program Olimpiads;
var IndMas: array [1..10] of integer := (15, 181, 211, 213, 215, 367,
                                         1115, 1125, 1560, 1820);
    KolMas: array [1..10] of integer;
    NoSch, NoNoSch, i: integer;
    c: char;
    Status: string[10];
begin
  read(NoSch);
  while (NoSch <> 0) do begin
    for i := 1 to 4 do // пропуск 4-х ненужных данных
      repeat
        read(c)
      until c = ' ';
    read(Status);
    for i := 1 to 10 do
      if IndMas[i] = NoSch then begin NoNoSch := i; break; end;
    // NoNoSch содержит номер месяца
    if Status = 'победитель' then
      KolMas[NoNoSch] := KolMas[NoNoSch] + 1;
    read(NoSch);
  end;
  for i := 1 to 10 do
    if KolMas[i] <> 0 then
      writeln('Школа № ',IndMas[i],' — победителей: ', KolMas[i]);
end.
```

Задачи для самостоятельного решения

1. Вводится строка текста, завершаемая точкой. Подсчитать в ней отдельно количества вхождений строчных латинских букв от 'a' до 'z'. Для букв, не входящих в заданную строку, информацию не выводить.
2. Меценаты города *N*: Пётр Ведмедев, Ник Курт, Иван Корзун, Алексей Хитров, Дмитрий Митричёв, Антон Хитров, Гиви Лагидзе, Джон Смирнофф, Анна Колычёва, Илья

Раневский, Анатолий Петренко и Инна Румова в течение года делали пожертвования в детский фонд, в том числе неоднократно. При этом в базе данных этого фонда каждый раз делалась соответствующая запись вида:

<номер записи> <Фамилия> <Имя> <месяц> <сумма взноса, руб.>.

Всего таких записей накопилось M штук (M не превышает 700 000).

На вход программы подаётся вначале количество записей M, затем сами записи.

Подсчитать для каждого мецената общую сумму пожертвований и вывести эту информацию в виде:

<Фамилия> <Имя> – пожертвовал <общая сумма пожертвований>.

Если кто-то из меценатов пожертвовал за учётный период менее 10 000 рублей, то его фамилию и имя в список вывода не включать.

3. В городе N работают 15 автобаз (№№ 3, 18, 23, 25, 26, 27, 38, 52, 55, 380, 387, 500, 818, 1132 и 1508). В ГАИ города N при выписке штрафов в базу данных заносятся записи вида:

<Фамилия> <Имя> <дата> <номер месяца> <год> <номер автобазы> <сумма штрафа, руб.>.

За 5 лет таких записей накопилось X (где X не превышает 10 000).

Автотрест города N обрабатывает этот массив записей. При этом на вход программы вначале подаётся количество записей, затем сами записи.

Требуется подсчитать сумму штрафов, начисленных всем водителям каждой из автобаз в отдельности за последние 3 года (2010 — 2012 включительно). Если для какой-то автобазы нет начисленных её водителям штрафов, то запись для этой автобазы не выводить.

Ответы для самопроверки

Задача 1.

1. Требуется подсчёт количества каждой из строчных букв от *a* до *z*. Поэтому определяется массив-счётчик с индексами элементов, обозначаемыми буквами от *a* до *z*:

```
var Kol: array ['a'..'z'] of integer;
```

2. Чтение входных данных — посимвольное, в цикле с предусловием, пока очередной прочитанный символ не будет являться точкой:

```
read(c);
while (c <> ".") do begin
  ...
  read(c);
end;
```

3. Обработка данных внутри цикла — проверка: если очередной символ является английской буквой, то увеличить на 1 значение элемента массива-счётчика, индекс которого равен полученному символу.

```
if (c >= 'a') and (c <= 'z') then Kol[c] := Kol[c] + 1;
```

Полный текст программы:

```
Program Alfavit2;
var Kol: array ['a'..'z'] of integer;
    c: char;
begin
  // обнуление массива-счетчика
  for c := 'a' to 'z' do Kol[c] := 0;
```

```
    // чтение и обработка данных
  read(c);
  while (c <> '.') do begin
    if (c >= 'a') and (c <= 'z') then Kol[c] := Kol[c] + 1;
    read(c);
  end;
  // вывод результата
  for c := 'a' to 'z' do
    if Kol[c] <> 0 then
      writeln('Буква ', c, ' встречается ', Kol[c], ' раз');
end.
```

Задача 2.

1. Подсчёт ведётся для каждого из 12 меценатов, следовательно, массив-счётчик должен иметь размер 12 элементов. Тип массива-счётчика — integer.

Поскольку меценаты обозначаются их фамилиями и именами, требуется использовать индексный массив из 12 пар элементов, которые содержат соответствующие каждому меценату имя и фамилию:

```
var IndMas: array [1..12,1..2] of string[10] := (('Петр','Ведмедев'),
    ('Ник','Курт'),('Иван','Корзун'), ('Алексей','Хитров'),
    ('Дмитрий','Митричев'), ('Антон','Хитров'), ('Гиви','Лагидзе'),
    ('Джон','Смирнофф'), ('Анна','Колычёва'), ('Илья','Раневский'),
    ('Анатолий','Петренко'), ('Инна','Румова'));
     KolMas: array [1..12] of integer;
```

2. Чтение строк производится при помощи цикла с параметром (предварительно считывается количество этих строк).

При чтении каждой строки нужно пропустить номер записи, прочитать фамилию и имя, пропустить месяц и прочитать сумму взноса.

Для пропуска даты и месяца используем посимвольное считывание до пробела включительно.

Чтение фамилии и имени выполняем путём посимвольного считывания до пробела.

3. Обработка данных считанной строки:

- **поскольку для каждого из меценатов уникальным является сочетание имени и фамилии, при просмотре индексного массива нужно производить поиск по обоим этим параметрам; индекс запоминается в переменной** `NoMecen`;
- **в массиве-счётчике к значению элемента** `KolMas[NoMecen]` **прибавляется считанное значение суммы пожертвования.**

4. Вывод данных: для каждого мецената (индекс от 1 до 12):

- **если сумма пожертвований превышает 10000, то выводится строка вида:**

`<Фамилия> <Имя> – пожертвовал <общая сумма пожертвований>`

При этом фамилия и имя извлекаются из индексного массива.

Полный текст программы:

```
Program Mecenats;
var IndMas: array [1..12,1..2] of string[10] := (('Петр','Ведмедев'),
    ('Ник','Курт'), ('Иван','Корзун'), ('Алексей','Хитров'),
    ('Дмитрий','Митричев'), ('Антон','Хитров'), ('Гиви','Лагидзе'),
    ('Джон','Смирнофф'), ('Анна','Колычёва'), ('Илья','Раневский'),
    ('Анатолий','Петренко'), ('Инна','Румова'));
```

```
    KolMas: array [1..12] of integer;
    i, j, M, NoMecen, Sum : integer;
    Fam, Name: string[10];
    c: char;
begin
   read(M); // кол-во записей
   for i := 1 to M do begin
     repeat // пропуск номера записи
       read(c)
     until c = ' ';
     Fam := ''; // считывание фамилии
     read(c);
     while (c <> ' ') do begin
       Fam := Fam + c;
       read(c);
     end;
     Name := ''; // считывание имени
     read(c);
     while (c <> ' ') do begin
       Name := Name + c;
       read(c);
     end;
     repeat // пропуск месяца
       read(c)
     until c = ' ';
     read(Sum); // считывание суммы пожертвования
// обработка данных
    for j := 1 to 12 do
      if (IndMas[j,1] = Name) and (IndMas[j,2] = Fam) then begin
        NoMecen := j; break; end;
    KolMas[NoMecen] := KolMas[NoMecen] + Sum;
 end;
// вывод результатов
    for i := 1 to 12 do
    if KolMas[i] > 10000 then
    writeln(IndMas[i,2], ' ', IndMas[i,1], ' - пожертвовал ', KolMas[i]);
end.
```

Задача 3.

1. Имеем 15 автобаз с большим разбросом значений их номеров, поэтому необходимо использовать индексный массив, содержащий номера автобаз. Размер индексного массива и массива-счётчика равен 15 элементам (по количеству автобаз).

```
   var IndMas: array [1..15] of integer := (3, 18, 23, 25, 26, 27, 38, 52,
                                            55, 380, 387, 500, 818, 1132, 1508);
       KolMas: array [1..15] of integer;
```

2. Чтение строк производится при помощи цикла с параметром (предварительно считывается количество этих строк).

При чтении каждой строки нужно пропустить данные: <Фамилия> <Имя> <дата> <номер месяца>.

Прочитываем год, номер автобазы и сумму штрафа.

3. Обработка данных:

- **просмотром в цикле от 1 до 15 индексного массива ищется индекс ячейки, значение которой совпадает с введённым номером автобазы; запоминается этот индекс (переменная** `NoNoAvt`**);**
- **проверяется считанное значение года: если он равен 2010, 2011 или 2012, то в массиве-счётчике к значению элемента** `KolMas[NoNoAvt]` **прибавляем значение** `Sum`**.**

4. При выводе данных просматривается весь массив-счётчик, для его ненулевых ячеек выводится накопленная сумма штрафов, номер автобазы при этом берётся из индексного массива.

Полный текст программы:

```
Program Avtobases;
var IndMas: array [1..15] of integer := (3, 18, 23, 25, 26, 27, 38, 52,
                                         55, 380, 387, 500, 818, 1132, 1508);
    KolMas: array [1..15] of integer;
    i, j, X, NoAvt, Year, Sum, NoNoAvt: integer;
    c: char;
begin
  read(X); // кол-во записей
  for i := 1 to X do begin
    for j := 1 to 4 do
      repeat // пропуск 4-х данных
        read(c)
      until c = ' ';
    read(Year);
    read(NoAvt);
    read(Sum);
// обработка данных
    for j := 1 to 15 do
      if (IndMas[j] = NoAvt) then begin NoNoAvt := j; break; end;
    if (Year >= 2010) and (Year <= 2012) then
      KolMas[NoNoAvt] := KolMas[NoNoAvt] + Sum;
end;
// вывод результатов
  for i := 1 to 15 do
    if KolMas[i] <> 0 then
      writeln('Автобаза ', IndMas[i], ' — ', KolMas[i], ' руб.');
end.
```

Программирование

C4 Задачи на анализ и обработку данных: разбор задач

Конспект

Пропуск текстовых данных из потока символов

а) использование «буферной» переменной соответствующего типа, которая далее не используется (при чтении следующего ненужного данного используется та же переменная);

б) для пропуска текстовых данных в цикле считывать из входной строки отдельные символы, включая пробел, завершающий очередное данное. Для пропуска ещё одного такого данного повторно строится такой же цикл.

Массивы-счётчики

Для подсчёта количеств нескольких объектов используется **массив-счётчик** размера, соответствующего количеству объектов. При обнаружении в потоке входных данных объекта, соответствующего условию, нужно увеличить на 1 значение ячейки массива-счётчика, соответствующей данному объекту.

> При формировании массива-счётчика нужно не забывать, что в качестве индексов его элементов в Паскале могут использоваться не только произвольные диапазоны целых чисел, но и диапазоны символов, например букв алфавита.

Индексные массивы

Если в качестве индексов массива-счётчика невозможно (или не рационально) использовать значения, считываемые из входных данных в качестве идентификатора объектов, то применяется **индексный массив**:

1) массив-счётчик объявляется по количеству обрабатываемых объектов;

2) отдельно объявляется массив того же размера, который содержит обозначения соответствующих объектов;

3) при считывании входных данных соответствующее обозначение объекта ищется в индексном массиве (путём его просмотра с начала до конца); номер ячейки индексного массива, содержащей искомое обозначение объекта, запоминается;

4) если данный объект соответствует требуемому условию, то в массиве-счётчике увеличивается на 1 элемент, индекс которого равен запомненному индексу элемента в индексном массиве, содержащего обозначение этого объекта.

> Связь индексного массива и массива-счётчика осуществляется через общий индекс элемента. Если при выводе полученных данных требуется выводить обозначения объектов, то они берутся из индексного массива.

Разбор типовых задач

Задача 1*[1]. На вход программы подаётся текст на английском языке, заканчивающийся точкой (другие символы «.» в этом файле отсутствуют). Требуется написать программу на языке Паскаль или Бейсик, которая будет определять и выводить на экран английскую букву, встречающуюся в этом тексте чаще всего, и количество там таких букв. Строчные и прописные буквы при этом считаются не различимыми. Если искомых букв несколько, то программа должна выводить на экран первую из них по алфавиту. Например, пусть файл содержит следующую запись: It is not a simple task. Yes! Чаще всего здесь встречаются буквы I, S и T (слово Yes в подсчёте не учитывается, так как расположено после точки). Следовательно, в данном случае программа должна вывести два символа, разделённых пробелом: I 3

Решение

1. Требуется произвести подсчёт количеств каждой буквы от A до Z (без различения строчных и прописных букв), следовательно требуется массив-счётчик с индексами от A до Z.

```
var a: array['A'..'Z'] of integer;
```

Необходимость использовать индексный массив отсутствует.

После подсчёта количеств букв в полученном массиве значений требуется произвести поиск первого по счёту максимального значения и вывести на экран индекс этого элемента (английскую букву) и само его значение (количество букв).

2. Входные данные — поток символов текста, завершаемый точкой. При этом текст может содержать посторонние символы (в частности, пробелы), которые нужно исключить из рассмотрения. Чтение можно осуществить посимвольно в цикле с постусловием, проверяющим равенство считанного символа точке.

```
repeat
  read(c);
  <обработка символа>
until (c = '.');
```

3. Обработка считанных данных: преобразовывается считанный символ в прописной регистр (стандартная функция `uppercase()`), проверяем символ на вхождение в заданное множество и если да, то увеличивается на 1 значение элемента массива-счётчика, индекс которого равен текущему считанному символу.

```
c := uppercase(c); // преобразование в прописной регистр
if c in ['A'..'Z'] then a[c] := a[c] + 1;
```

> Проверить, входит ли символ в заданное множество символов, позволяет оператор `in`. Запись вида `<символ> in [<множество>]` истинна, если символ входит в данное множество. При этом требуемое множество в квадратных скобках записывается как перечисление через запятую отдельных символов или интервалов символов. Примеры:
> • множество знаков препинания: `['.', ',', '!', '?', ';', ':']`;
> • множество латинских букв: `['A'..'Z', 'a'..'z']`.

4. Обработка полученного массива-счётчика: стандартный алгоритм поиска максимума в одномерном массиве с условием в виде строгого неравенства (без переприсваивания, если текущий элемент окажется равен предполагаемому максиму) и с запоминанием индекса предполагаемого максимального элемента.

---

[1] Условие задачи здесь несколько изменено: в изначальном варианте задания исходный текст содержался в файле и для его чтения требовалось применение операторов для работы с файлами. Поскольку во все последующие годы на ЕГЭ не предлагались задания, связанные с работой с файлами, в данном пособии файловые операции не рассматриваются.

Достаточно запоминать индекс и непосредственно сравнивать между собой элементы массива, не заводя отдельную переменную для хранения предполагаемого максимума.

```
cmax := 'A';
for c := 'B' to 'Z' do
  if a[c] > a[cmax] then
    cmax := c;
```

5. Вывод данных: индекс найденного максимального элемента массива-счётчика, пробел, значение найденного максимального элемента массива-счётчика.

```
writeln(cmax,' ',a[cmax]);
```

Полный текст программы:

```
program C5;
var a: array['A'..'Z'] of integer;
    c, cmax: char;
begin
  for c := 'A' to 'Z' do a[c] := 0; // обнуление массива-счетчика
  repeat
    read(c);
    c := uppercase(c);
  if c in ['A'..'Z'] then a[c] := a[c] + 1;
  until (c = '.');
  cmax := 'A';
  for c := 'B' to 'Z' do
    if a[c] > a[cmax] then
      cmax := c;
  writeln(cmax,' ',a[cmax])
end.
```

Задача 2*. На вход программе подаются 366 строк, которые содержат информацию о среднесуточной температуре всех дней 2004 года. Формат каждой из строк следующий: сначала записана дата в виде **dd.mm** (на запись номера дня и номера месяца в числовом формате отводится строго два символа, день от месяца отделён точкой), затем через пробел записано значение температуры — число со знаком плюс или минус, с точностью до 1 цифры после десятичной точки. Данная информация отсортирована по значению температуры, то есть хронологический порядок нарушен. Требуется написать программу на языке Паскаль или Бейсик, которая будет выводить на экран информацию о месяце (месяцах), среднемесячная температура у которого (которых) наименее отклоняется от среднегодовой. В первой строке вывести среднегодовую температуру. Найденные значения для каждого из месяцев следует выводить в отдельной строке в виде: номер месяца, значение среднемесячной температуры, отклонение от среднегодовой температуры.

Решение

1. Требуется подсчёт средней температуры для каждого из 12 месяцев. Для этого необходимо для каждого месяца определить сумму всех значений температуры в каждый из дней этого месяца, следовательно, требуется массив-счётчик на 12 элементов (по числу месяцев). Тип массива-счётчика — real, так как значения температуры — вещественные числа с точностью до 1 знака после запятой.

```
tm: array[1..12] of real;
```

Поскольку входными данными являются номера месяцев, индексный массив не требуется.

По завершении обработки входных данных для каждого месяца разделить подсчитанную сумму на количество дней в месяце. Так как удобно реализовать эту операцию в цикле, нуж-

но заранее заготовить массив, содержащий для каждого месяца количество дней в нём (принципы работы с таким массивом аналогичны принципам работы с индексным массивом).

```
Const d: array[1..12] of integer = (31,29,31,30,31,30,31,31,30,31,30,31);
```

Для подсчёта среднегодовой температуры объявляется отдельная переменная, в которой суммируются *все* считанные значения температуры независимо от месяца. По завершении обработки входных данных среднегодовая температура вычисляется делением подсчитанной суммы температур на общее число записей (366).

2. Входные данные — набор строк, количество которых заранее известно (366), следовательно, используется цикл с параметром, конечное значение которого задано константой.

Строки имеют вид: <dd.mm> <температура>, т. е. каждая строка содержит вначале запись даты и месяца (по 2 цифры, разделённые точкой), а затем через пробел — значение температуры (вещественное число).

Как удобнее всего выполнить чтение данных, чтобы получить номер месяца?

Запись <dd.mm> можно считать как строку текста из 5 символов, из которой после её считывания извлекается подстрока (последние 2 символа) и преобразуется в собственно число.

Другой вариант — пропуск лишних символов до точки включительно и последующее чтение номера месяца как целого числа.

Получить число из его символьной записи (строки цифр) можно:

- при помощи стандартных функций `copy(<строка>, <начальная позиция>, <длина>)` (извлекает из строки подстроку заданной длины, начинающуюся с символа с заданной начальной позицией), либо `LeftStr(<строка>, <длина>)` (извлекает из строки подстроку заданной длины слева), или `RightStr(<строка>, <длина>)` (извлекает из строки подстроку заданной длины справа) и `StrToInt(<строковая запись числа>)` (преобразует запись целого числа в виде строки цифр в само число);
- путём обработки кодов символов цифр: код символа позволяет получить стандартная функция `ord(<символ>)`, при этом разность кода символа заданной цифры и кода символа цифры «нуль» равна числовому значению заданной цифры (см. таблицу кодировки ASCII). Тогда можно получить число, отдельно обрабатывая каждую его цифру указанным способом и суммируя полученные числовые значения цифр с учётом их разрядов (умножением на 10 в степени, равной номеру разряда).

3. Обработка считанных данных:

1) из считанной строки длиной 5 символов извлекается номер месяца (как сумму числового значения 4-й цифры, умноженного на 10, и числового значения 5-й цифры);

2) в массиве-счётчике прибавляется считанное значение температуры к значению элемента, индекс которого равен полученному номеру месяца;

3) считанное значение температуры также прибавляется к значению переменной, в которой накапливается сумма температур за год.

По завершении обработки входных данных вычисляются среднемесячные температуры (цикл по номерам месяцев от 1 до 12) делением запомненных сумм температур н количества дней в соответствующем месяце, а также среднегодовую температуру.

4. Обработка полученного массива-счётчика: алгоритм поиска минимума модуля разности значений среднемесячной и среднегодовой температуры. При этом в качестве первого значения предполагаемого минимума берётся число, заведомо большее любого возможного значения такой разности (например, 100).

5. Вывод данных:

1) выводится значение среднегодовой температуры;

2) просматриваются все месяцы (цикл по номерам месяцев от 1 до 12), и если модуль разности среднемесячной и среднегодовой температуры равен найденному минимуму, то выводится номер месяца, значение среднемесячной температуры и модуль разности среднемесячной и среднегодовой температуры для этого месяца.

Проверка равенства вещественных значений
Как известно, вещественные значения в памяти компьютера представлены приближённо. Поэтому, например, вычисленное значение `2.0 + 2.0` в общем случае может оказаться не равно вычисленному значению `2.0 * 2.0` (из-за погрешностей вычислений). Из-за этого использовать при сравнении вещественных значений оператор равенства не совсем корректно. Вместо этого производится сравнение модуля разности сравниваемых величин с некоторым малым значением, определяемым точностью вычислений. В данном случае в качестве такого малого значения можно использовать число `0,0001`.

Полный текст программы:

```
program C5;
const d: array[1..12] of integer = (31,29,31,30,31,30,31,31,30,31,30,31);
// количества дней в месяцах
var tm: array[1..12] of real; // массив-счетчик
    m: 1..12; // номер месяца
    data: string[5]; // строка для считывания записи «dd.mm»
    min, ty, t: real;
    i: integer;
begin
  for i := 1 to 12 do tm[i] := 0; // обнуление массива-счетчика
  ty := 0; // обнуление счетчика среднегодовой температуры

  for i := 1 to 366 do // чтение входных данных
  begin
    readln(data,t);
    m := (ord(data[4]) - ord('0')) * 10 + ord(data[5]) - ord('0');
    // вычисление номера месяца
    tm[m] := tm[m] + t; // сумма температур для данного месяца
    ty := ty + t; // сумма температур за год
  end;

  for i := 1 to 12 do
    tm[i] := tm[i] / d[i]; // вычисление среднемесячных температур
  ty := ty / 366; // вычисление среднегодовой температуры
  min := 100; // предполагаемое значение минимума
  for i := 1 to 12 do
    if abs(tm[i] - ty) < min then
    min := abs(tm[i] - ty);

// вывод данных
  writeln('Среднегодовая температура = ',ty:0:2);
  for i := 1 to 12 do
    if abs(abs(tm[i]-ty)-min) < 0.0001 then
    writeln(i,' ',tm[i]:0:2,' ',abs(tm[i]-ty):0:2);
  readln
end.
```

Задача 3*. Вступительные испытания в некоторый вуз состоят из трёх экзаменов: математика (максимальный балл — 9), информатика (максимальный балл — 9), литература (максимальный балл — 5). На вход программе подаются сведения о сдаче этих экзаменов абитуриентами. В первой строке вводится количество абитуриентов N, во второй — количество мест K (K < N) на которые эти абитуриенты претендуют. Каждая из следующих N строк имеет сле-

дующий формат: <Фамилия> <оценка1> <оценка2> <оценка3>, где <Фамилия> — строка, состоящая не более, чем из 20 символов, оценки — числа от 0 до максимальной оценки по предмету соответственно. (Ноль ставится в случае, если экзамен не сдавался, например, после полученной на предыдущем экзамене двойки. Все баллы, большие 2, считаются удовлетворительными).

Пример входных строк:

Иванов 8 9 3

Петров 2 0 0

Требуется написать программу на языке Паскаль или Бейсик, которая определяла бы по имеющимся данным количество абитуриентов, набравших полупроходной балл в данный вуз или сообщала, что такой балл отсутствует. (Полупроходным называется такой балл, что лишь часть абитуриентов, набравших такой балл и не получивших ни одной неудовлетворительной оценки, попадает в K лучших, которые должны быть зачислены на 1 курс) Считается, что абитуриенты, получившие только удовлетворительные оценки, обязательно присутствуют.

Решение

1. Необходимо вычислять количества студентов, получивших на вступительных экзаменах тот или иной суммарный балл (как сумму баллов по трём экзаменам). Если максимальные баллы за экзамены составляют 9, 9 и 5 баллов, то максимальное значение суммарного балла равно 23, поэтому объявляется массив-счётчик с индексами элементов от 0 до 23 и для каждого значения баллов подсчитывается количество абитуриентов, набравших такое число баллов:

```
var m: array[0..23] of integer;
```

При этом, поскольку учитываются только удовлетворительные оценки (больше 3), минимальное число учитываемых баллов равно 9 (3 + 3 + 3). Если на каком-то экзамене получена хотя бы одна неудовлетворительная оценка, то абитуриент выбывает из учёта, т. е. для него сумму баллов можно считать равной нулю. Итого из массива-счётчика используются ячейки с индексами 0 и 9..23, неиспользуемых — 8 ячеек из 24. Это достаточно приемлемо, индексный массив не используется (его использование привело бы к большему расходу памяти).

2. Входные данные: вначале считываются значения N и K, затем в цикле с параметром считываются N строк данных.

Для каждой строки вначале пропускается ненужное данное — фамилия абитуриента (посимвольно до пробела включительно), а затем считываются три целых числа — баллы за три экзамена соответственно.

3. Обработка считанных данных:

1) если хотя бы одно количество баллов меньше 3, то вся сумма приравнивается нулю, иначе вычисляется сумма баллов;

2) в массиве-счётчике на 1 увеличивается значение элемента с индексом, равным вычисленной сумме баллов.

4. Обработка полученного массива-счётчика. Полупроходной балл, по условию, — это такой балл, что при зачислении абитуриентов, начиная с наилучших (с наибольших значений баллов — с конца массива-счётчика), наступит момент, что абитуриентов, набравших некоторый балл, больше, чем можно зачислить. Именно этот балл и будет «полупроходным». Но если абитуриентов так мало, что их общее количество меньше требуемой численности группы K, либо «граница зачисления» точно проходит по границе между двумя соседними значениями баллов (с каким-то баллом *все* абитуриенты зачислены, а с баллом на 1 меньше — ни один не зачислен), то «полупроходного» балла не существует.

Реализация такой обработки может быть выполнена в виде цикла, в котором с конца массива-счётчика (от максимальных баллов) подсчитывается сумма количеств абитуриентов, набравших баллы, и сравнивается с значением K:

- если на очередном шаге уже накопленная сумма (меньшая K) плюс количество абитуриентов с баллом на 1 меньше окажется больше I, то именно этот балл (на 1 меньше текущего) и является «полупроходным»;
- если же такого не случилось вплоть до достижения минимального количества баллов, равного 9, то «полпроходной» балл отсутствует.

Программный код[1]:

```
s := m[23]; i := 23;
// начинаем с конца массива-счетчика
while (s < K) and (i => 9) do begin
// если текущее суммарное количество лучших абитуриентов меньше
// количества зачисляемых (K) и при этом еще не все абитуриенты
// рассмотрены, то
  i := i - 1;
  // переходим к рассмотрению следующей группы абитуриентов с баллами на
  // 1 меньше
  s := s + m[i];
  // и включаем их в сумму
end;
// завершение цикла либо по найденному полупроходному баллу i,
// либо по окончанию просмотра всего количества абитуриентов (i<9)

if (s > K)and(i => 9) then
    // если полученная сумма больше K, то
    writeln('полупроходной балл набрали', m[i], ' человек')
    // иначе (если сумма равна или меньше K)
else writeln('полупроходной балл отсутствует');
```

Полный текст программы:

```
program C4;
var m: array[0..23] of integer;
    c: char;
    i, K, N, S, m1, m2, m3: integer;
begin
  for i := 0 to 23 do m[i] := 0; // обнуление массива-счетчика
  readln(N); // ввод количества абитуриентов
  readln(K); // ввод количества зачисляемых

  for i := 1 to N do // чтение строк входных данных
  begin
    repeat
      read(c)
    until c = ' '; // пропуск фамилии абитуриента
    readln(m1, m2, m3); // чтение баллов за экзамены
```

[1] Этот вариант решения отличается от предложенного в качестве ответа в демонстрационной версии ЕГЭ 2006 года: устранена ошибка (не отслеживается случай, когда уже количество студентов с баллом 23 превышает значение K) и упрощено понимание работы алгоритма. — Прим. авт.

```
    if (m1 < 3)or(m2 < 3)or(m3 < 3) then s := 0
   // если хотя бы один балл неудовлетворительный, то сумма приравнивается
    // к нулю
    else s := m1 + m2 + m3; // иначе вычисляется суммарный балл
    m[s] := m[s] + 1 // подсчет количества абитуриентов с таким баллом
  end;

  s := m[23]; i := 23;
  // начинаем с конца массива-счетчика
  while (s < K) and (i => 9) do begin
  // если текущее суммарное количество лучших абитуриентов меньше
  // количества зачисляемых (K) и при этом еще не все абитуриенты
  // рассмотрены, то
  i := i - 1;
  // переходим к рассмотрению следующей группы абитуриентов с баллами
  // на 1 меньше
  s := s + m[i];
  // и включаем их в сумму
 end;
  // завершение цикла либо по найденному полупроходному баллу i,
  // либо по окончанию просмотра всего количества абитуриентов (i<9)

 if (s > K)and(i => 9) then
     // если полученная сумма больше K, то
     writeln('полупроходной балл набрали', m[i], ' человек')
 // иначе (если сумма равна или меньше K)
 else writeln('полупроходной балл отсутствует');

end.
```

Задача 4*. На вход программе подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке сообщается количество учеников N, которое не меньше 10, но не превосходит 100, каждая из следующих N строк имеет следующий формат: <Фамилия> <Имя> <оценки>, где <Фамилия> — строка, состоящая не более чем из 20 символов, <Имя> — строка, состоящая не более чем из 15 символов, <оценки> — через пробел три целых числа, соответствующие оценкам по пятибалльной системе. <Фамилия> и <Имя>, а также <Имя> и <оценки> разделены одним пробелом.

Пример входной строки:

Иванов Петр 4 5 4

Требуется написать программу, которая будет выводить на экран фамилии и имена трёх лучших по среднему баллу учеников. Если среди остальных есть ученики, набравшие тот же средний балл, что и один из трёх лучших, то следует вывести и их фамилии и имена. Требуемые имена и фамилии можно выводить в произвольном порядке.

Решение

1. Требуется подсчёт среднего балла для каждого ученика, при этом нужно запоминать их фамилии и имена. Следовательно, необходим массив-счётчик по количеству учеников (до 100), способный хранить для каждого ученика текстовую строку (фамилия + имя) и числовое значение. Наиболее эффективная реализация — массив записей:

```
var p: array[1..100] of record
       name: string;
       sum: integer;
   end;
```

2. Входные данные: вначале отдельно считывается количество строк, а затем в цикле с параметром вводятся данные об N учениках.

При этом в первый элемент очередной записи надо считать (посимвольно, так как длина строки заранее не известна) сначала фамилию — до пробела после неё включительно, а затем имя — также до пробела включительно. Затем считываются и суммируются во втором элементе записи значения оценок. Эти операции можно объединить в цикле от 1 до 3 прямым прибавлением к заранее обнуленному значению второго элемента записи считанного очередного числа (по завершении цикла нужен оператор readln для перехода к следующей строке).

Средний балл для каждого ученика может быть вычислен делением полученной суммы на 3.

3. Обработка входных данных выполняется во время их чтения (суммирование оценок во втором элементе записи).

4. Обработка массива-счётчика: необходимо определить «третий максимум» (третье по величине значение среднего балла), а затем при повторном просмотре массива-счётчика (массива записей) выводить на экран информацию об учениках, у которых средний балл больше или равен найденному третьему максимуму.

Нужен поиск максимума среди средних баллов для выявления лучших учеников. Но для получения того же самого результата, очевидно, можно сравнивать не средние баллы, а просто суммарные баллы! Это позволит не только упростить программу (не требуется выполнять деление на 3), но и избежать работы с вещественными значениями (тогда как суммы баллов — это целые числа).

Учитывая это замечание, предыдущую часть решения можно изменить:

1'. Требуется подсчёт суммы баллов для каждого ученика, при этом нужно запоминать их фамилии и имена.

2'. При чтении входных данных во втором элементе очередной записи вычисляется сумма баллов для данного ученика, но её деление на 3 для вычисления среднего балла не производится.

3'. Обработка входных данных выполняется во время их чтения (суммирование оценок во втором элементе записи).

4'. Обработка массива-счётчика: необходимо определить «третий максимум» (третье по величине значение суммы баллов), а затем при повторном просмотре массива-счётчика (массива записей) выводить на экран информацию об учениках, у которых сумма баллов больше или равна найденному третьему максимуму.

Заметим, что само значение среднего балла при этом выводить не требуется. Иначе можно было бы вычислять его непосредственно при выводе на экран, деля запомненную сумму баллов для очередного ученика на 3.

Как осуществить поиск «третьего максимума»?

1. Объявляются три переменные, содержащие предположительные значения первого максимума (s1), «второго максимума» (s2) и «третьего максимума» (s3).

2. Этим переменным изначально присваиваются значения, заведомо меньшие любого возможного значения суммы баллов, например нули.

3. Просматриваются все значения сумм баллов для каждого ученика (вторые элементы записей для элементов массива от 1 до N).

4. Если текущая сумма баллов больше первого максимума s1, то:

- это значение суммы станет новым значением s1;
- перед этим прежнее значение s1 станет новым значением «второго максимума» s2;
- перед этим прежнее значение s2 станет новым значением «третьего максимума» s3.

5. Иначе, если текущая сумма баллов больше «второго максимума» s2, то:

• это значение суммы станет новым значением s2;

• перед этим прежнее значение s2 станет новым значением «третьего максимума» s3.

6. Иначе, если текущая сумма баллов больше «третьего максимума» s3, то это значение суммы станет новым значением s3.

Результат: «третий максимум» будет храниться в переменной s3.

Фрагмент программного кода (в комментариях обозначены вышеуказанные шаги алгоритма):

```
s1 := 0; s2 := 0; s3 := 0;                    // шаги 1 и 2
for i := 1 to N do                            // шаг 3
begin
  if p[i].sum > s1 then                       // шаг 4
    begin
      s3 := s2; s2 := s1;
      s1 := p[i].sum
    end else
  if p[i].sum > s2 then                       // шаг 5
    begin
      s3 := s2; s2 := p[i].sum
    end else
  if p[i].sum > s3 then s3 := p[i].sum;       // шаг 6
end;
```

5. Вывод данных: просмотр массива записей (учеников от 1 до N). Если у очередного ученика значение суммы баллов (второй элемент записи) больше или равен ранее вычисленному «третьему максимуму» (s3), то выводится фамилия и имя ученика (значение первого элемента записи).

Полный текст программы:

```
var p: array[1..100] of record // массив записей
      name: string;
      sum: integer;
    end;
    c: char;
    i, j, N, s1, s2, s3, m: integer;
begin
  readln(N); // считано количество учащихся
  for i := 1 to N do // чтение строк об учениках
  begin
    p[i].name := ''; // «обнуление» первого элемента записи
    repeat
      read(c);
      p[i].name := p[i].name + c
    until c = ' ';  // считана фамилия (включая пробел после нее)
  repeat
    read(c);
    p[i].name := p[i].name + c
  until c = ' ';      // считано имя
  p[i].sum := 0;      // обнуление второго элемента записи
  for j := 1 to 3 do // чтение трех значений оценок
```

```
  begin
    read(m);       // считана очередная оценка
    p[i].sum := p[i].sum + m    // оценка прибавляется к сумме
  end;   // сумма баллов вычислена
  readln; // переход к следующей строке входных данных
end;

// поиск «третьего максимума»:
s1 := 0; s2 := 0; s3 := 0;
for i := 1 to N do
begin
  if p[i].sum > s1 then
    begin
      s3 := s2; s2 := s1;
      s1 := p[i].sum
    end else
  if p[i].sum > s2 then
    begin
      s3 := s2; s2 := p[i].sum
    end else
  if p[i].sum > s3 then s3 := p[i].sum;
end;

// просмотр учеников: если сумма баллов больше или равна
// «третьему максимуму», то выводится фамилия и имя
  for i := 1 to N do
    if p[i].sum >= s3 then writeln(p[i].name);
end.
```

Задача 5*. По итогам городской олимпиады школам разослан список её участников с указанием количества баллов, набранных каждым из них.

Требуется написать эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая для заданного номера школы Z определяет двух самых лучших по числу набранных баллов олимпиадников, учащихся данной школы, и выводит на экран их фамилии и имена.

Если наибольший балл получили более двух учащихся, то вывести на экран количество таких учащихся.

Если наибольший балл получил один учащийся, а следующий балл получили несколько учеников, то нужно вывести только фамилию и имя одного лучшего.

Известно, что в олимпиаде участвовало больше 5 учеников каждой школы.

На вход программе сначала подаётся номер школы Z, затем в следующей строке вводится общее количество учеников олимпиады N. В каждой из следующих N строк записана информация об учениках в формате:

<Фамилия> <Имя> <Номер школы> <Количество баллов>,

где <Фамилия> — строка, состоящая не более чем из 30 символов без пробелов, <Имя> — строка, состоящая не более чем из 20 символов без пробелов, <Номер школы> — целое число в диапазоне от 1 до 99, количество баллов> — целое число в диапазоне от 1 до 100. Эти данные записаны через пробел, причём ровно один между каждой парой (то есть всего по три пробела в каждой строке).

Пример входной строки:

Иванов Иван 50 87

Пример выходных данных:
Круглое Василий
Тарасова Дарья
Другой вариант выходных данных:
7
Третий вариант выходных данных:
Гусарский Илья

Решение

В отличие от предыдущих задач, здесь количество записей о школьниках может быть очень большим (максимально возможное значение N не оговорено). Вместе с тем, в задаче не требуется выполнять операции, связанные с подсчётами количеств объектов или суммарных величин для них. Поэтому можно выполнять требуемый поиск двух лучших учеников сразу при вводе данных, не запоминая их и не используя массив-счётчик. (Фамилии и имена лучших учеников, как и их баллы, запоминаются в простых переменных.)

1. Требуется запоминать фамилии, имена и количества баллов для двух лучших школьников. Поэтому вместо массива-счётчика используются соответствующие простые переменные.

2. При чтении входных данных сначала считываются номер школы Z и количество строк, а затем сами строки. При этом фамилия + имя считываются в переменную S, номер школы — в переменную sh, баллы — в переменную ball. Номер школы служит только для проверки: если он не равен заданному, то данные из этой строки в обработку не включаются.

3. Обработка входных данных — поиск первого и «второго» максимума с одновременным подсчётом количеств значений, равных им:

1). Используются шесть переменных, содержащих: предположительные значения первого и второго максимумов (max и max2 соответственно), соответствующие им фамилии-имена учеников (Smax и Smax2 соответственно) и счётчики количеств значений баллов, равных первому и второму максимуму (nmax и nmax2 соответственно).

2). Изначально переменным max и max2 присваиваются значения, заведомо меньшие любого возможного значения баллов, например −1. Переменной Smax присваивается значение «пустая строка». Переменной nmax присваивается нулевое значение. (Переменные Smax2 и nmax2 можно не «обнулять».)

> Поскольку поиск максимумов производится по мере ввода данных, т. е. в том же цикле, что и ввод данных, то все указанные инициализационные присваивания значений переменных должны быть выполнены **до** начала этого цикла чтения входных данных!

3). Только что считанное значение количества баллов (ball) сравнивается с предполагаемыми первым и вторым максимумом.

4). Если текущее значение баллов больше предполагаемого первого максимума max, то:

- это значение баллов станет новым значением max, считанное значение S (фамилия и имя текущего ученика) запоминаются в Smax, а счётчик количества значений, равных первому максимуму, «очищается» (становится равным 1, т. е. пока найдено одно такое значение);
- перед этим прежнее значение max становится новым значением «второго максимума» max2, прежнее значение Smax перезапоминается как Smax2 (бывший «первый ученик» стал вторым), а ранее найденное количество значений, равных max, перезапоминается как значение счетчика количества элементов, равных второму максимуму.

5). Иначе, если текущий балл равен текущему максимальному, то:

- счётчик количества элементов, равных максимальному, увеличивается на 1;
- раз найден второй лучший ученик, то его значение max копируется в качестве второго максимума max2, а в Smax2 запоминается фамилия и имя текущего ученика (как второго лучшего).

6). Иначе, если текущее значение баллов больше предполагаемого второго максимума max2, то это значение баллов станет новым значением max2, считанное значение S (фамилия и имя текущего ученика) запоминаются в Smax2, а счётчик количества значений, равных второму максимуму, «очищается» (становится равным 1, т. е. пока найдено одно такое значение).

7). Иначе, если текущий балл равен текущему второму максимуму, то счётчик количества элементов, равных второму максимуму, увеличивается на 1.

Фрагмент программного кода (в комментариях обозначены вышеуказанные шаги алгоритма):

```
max := -1; // шаги 1 и 2
Smax := '';
nmax := 0;
max2 := -1;
<цикл чтения данных>
   <чтение данных из очередной строки>
   if ball > max then begin // шаг 4
     max2 := max; Smax2 := Smax; nmax2 := nmax;
     max := ball; Smax := S; nmax :=1
    end else
   if ball = max then begin // шаг 5
     nmax := nmax + 1; max2 := max; Smax2 := S
    end else
   if ball > max2 then begin // шаг 6
     max2 := ball; Smax2 := S; nmax2 := 1
    end else
   if ball = max2 then // шаг 7
     nmax2 := nmax2 + 1
end;
```

5. Вывод данных: необходимо проанализировать три случая:

1) имеется ровно два лучших ученика — или оба имеющие один и тот же максимальный балл, или один с максимальным баллом и один с баллом, равным второму максимуму, тогда надо вывести фамилии и имена их обоих;

```
if (nmax = 2) or (nmax = 1) and (nmax2 = 1) then begin
  writeln (Smax);
  writeln (Smax2)
```

2) иначе если имеется только один лучший ученик (с баллами, равными максимуму), а количество вторых учеников (у которых балл равен второму максимуму) больше одного, тогда надо вывести только фамилию и имя лучшего ученика;

```
if (nmax = 1) and (nmax2 > 1) then
  writeln(Smax)
```

3) иначе если лучших учеников (с баллами, равными максимуму) больше двух, тогда надо вывести только их количество.

```
  writeln(nmax)
```

Полный текст программы:

```
var S, Smax, Smax2: string [52];
    ch: char;
    i, Z, N, sh, ball, max, nmax, max2, nmax2: integer;
begin
  readln(Z); // считан номер школы
  readln(N); // считано количество строк
```

```
  max := -1; // перед началом цикла чтения строк данных
  Smax := ''; // инициализируются переменные
  nmax := 0; // для поиска первого и второго максимумов
  max2 := -1;

  for i := 1 to N do // чтение строк входных данных
   begin
    s := ''; // «обнуление» переменной для фамилии и имени
    repeat
      read(ch);
      s := s + ch
    until ch = ' '; // считана фамилия (включая пробел за ней)
    repeat
      read(ch);
      s := s + ch
    until ch = ' '; // считано имя
    readln(sh,ball); // считаны номер школы и балл ученика
    if sh = Z then // обрабатываются только ученики заданной школы

    // поиск первого и второго максимума
    if ball > max then // текущий балл больше максимального
     begin
      max2 := max; Smax2 := Smax; nmax2 := nmax;
      max := ball; Smax := S; nmax := 1
     end else
    if ball = max then // текущий балл равен максимальному
     begin
      nmax := nmax + 1; max2 := max; Smax2 := S
     end else
    if ball > max2 then // текущий балл больше второго максимума
     begin
      max2 := ball; Smax2 := S; nmax2 := 1
     end else
    if ball = max2 then // текущий балл равен второму максимуму
      max2 := nmax2 + 1
  end;

  // анализ полученных результатов и вывод на экран
  if (nmax = 2) or (nmax = 1) and (nmax2 = 1) then
  // два лучших ученика
   begin
    writeln (Smax);
    writeln (Smax2)
   end else
  if (nmax = 1) and (nmax2 > 1) then
  // один лучший ученик
   writeln(Smax)
  else
  // лучших учеников больше двух
   writeln(nmax)
end.
```

Задачи для самостоятельного решения

1. Радиотелескоп, ведущий наблюдение за звёздным небом в рамках программы поиска внеземных цивилизаций (SETI), принимает с одной из звёзд сигналы, которые можно воспринимать как закодированные цифры от 1 до 9. Исследовательский центр обрабатывает строки таких цифр, соответствующие последовательным сериям сигналов. Строка, начинающаяся нулём (или состоящая только из одного нуля), — конец серии сигналов. В строке последовательности цифр могут быть разбиты на группы пробелами. Общая длина каждой из строк не превышает 255 знаков.
   Можно предположить, что эти послания отправлены разумными существами, если суммарные количества каждой из цифр, переданных в послании, пропорциональны значениям этих цифр с одним и тем же коэффициентом, например:

Цифра	1	2	3	4	5	6	7	8	9	**Коэфф.**
Количество таких цифр	1	2	3	4	5	6	7	8	9	1

   или

Цифра	1	2	3	4	5	6	7	8	9	**Коэфф.**
Количество таких цифр	2	4	6	8	10	12	14	16	18	2

   или

Цифра	1	2	3	4	5	6	7	8	9	**Коэфф.**
Количество таких цифр	10	20	30	40	50	60	70	80	90	10

   Напишите программу, которая, анализирует входные строки (их количество не превышает 500 000), делает предположение о разумности отправителей сообщения и выводит на экран, соответственно, слово «РАЗУМ» или слово «СЛУЧАЙ».

2*. На вход программе подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке сообщается количество учеников N, которое не меньше 10, но не превосходит 100, каждая из следующих N строк имеет следующий формат:

   <Фамилия> <Имя> <оценки>,

   где <Фамилия> — строка, состоящая не более чем из 20 символов, <Имя> — строка, состоящая не более чем из 15 символов, <оценки> — через пробел три целых числа, соответствующие оценкам по пятибалльной системе. <Фамилия> и <Имя>, а также <Имя> и <оценки> разделены одним пробелом. Пример входной строки:
   Иванов Петр 4 5 3
   Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет выводить на экран фамилии и имена трех худших по среднему баллу учеников. Если среди остальных есть ученики, набравшие тот же средний балл, что и один из трех худших, то следует вывести и их фамилии и имена.

Ответы для самопроверки

Задача 1.

Решение

1. Производится подсчёт количества цифр от 1 до 9, причём работа производится с символами этих цифр, следовательно, используется массив-счётчик с индексами от '1' до '9'.

```
var a: array ['1'..'9'] of integer;
```

Внимание! Индексы элементов массива — это символы '1' .. '9', а не числа 1 .. 9 (либо при использовании в качестве индексов именно чисел нужно считанные из входной строки символы цифр преобразовывать в их числовые значения).

Поскольку в качестве индексов массива-счётчика используются сами считываемые обозначения объектов, использовать индексный массив не требуется.

2. Входные данные: считываются строки данных, пока не окажется, что очередная считанная строка первым символом имеет символ цифры «нуль» (цикл с предусловием).

3. Обработка считываемых данных: каждая очередная строка читается целиком в строковую переменную *S* максимальной длины 255 символов. Затем определяется реальное количество символов в считанной строке (стандартная функция `Length(S)`) и эта строка рассматривается как массив символов соответствующей длины.

Далее в цикле производится обращение к очередному символу, и если это цифра от 1 до 9, то значение элемента массива-счётчика с соответствующим индексом увеличивается на 1.

4. Обработка массива-счётчика: нужно проверить, действительно ли подсчитанные количества каждой из цифр пропорциональны значениям этих цифр с одним и тем же коэффициентом, т. е. действительно ли равны друг другу результаты деления хранящегося в массиве-счётчике количества каждой цифры на числовое значение этой цифры. При этом очевидно, что коэффициент равен количеству встреченных в сообщении цифр 1.

5. Вывод результатов: слово **«РАЗУМ»**, если указанное выше равенство соблюдается для всех цифр, или слово **«СЛУЧАЙ»**, если есть хотя бы одна цифра, для которой это правило не соблюдается.

Полный текст программы:

```
program SETI;
var a: array ['1'..'9'] of integer;
    S: string[255];
    N, L, i: integer;
    j: char;
    X: boolean;
begin
  readln(S); // первая строка считана как единое целое
  while (S[1] <> '0') do begin
  // чтение и обработка строк данных, пока они не начинаются с нуля
    L := Length(S); // реальная длина считанной строки
    for i := 1 to L do // посимвольный разбор строки
      if S[i] in ['0'..'9'] then // если это цифра, то
        a[S[i]] := a[S[i]] + 1;
        // символ — значение элемента массива с индексом i
        // рассматривается как индекс элемента в массиве a
    readln(S); // считана следующая строка
  end; // чтение входных строк закончено
//   обработка массива-счетчика
  X := (a['2']/2 = a['1']); // проверка первого такого соответствия
  for j := '3' to '9' do
   X := X and (a[j] / (ord(j) - ord('0')) = a['1']); // проверка очередного
                                                     // соответствия
  // если после всех проверок X имеет значение TRUE, то условие «количества
  // всех цифр пропорциональны значениям этих цифр с одним и тем же
  // коэффициентом» выполнено
  if X then writeln('РАЗУМ') else writeln('СЛУЧАЙ');
end.
```

Задача 2.

Решение

Эта задача решается аналогично задаче 4, разобранной ранее, при этом вместо средних баллов обрабатываются суммы баллов.

1. Требуется подсчёт суммы баллов для каждого ученика, при этом нужно запоминать их фамилии и имена.

2. При чтении входных данных во втором элементе очередной записи вычисляется сумма баллов для данного ученика.

3. Обработка входных данных — выполняется во время их чтения (суммирование оценок во втором элементе записи).

4. Обработка массива-счётчика: необходимо определить «третий минимум» (третье по малости значение суммы баллов), а затем при повторном просмотре массива-счётчика (массива записей) выводить на экран информацию об учениках, у которых сумма баллов меньше или равна найденному третьему минимуму.

Как осуществить поиск «третьего минимума»?

1) Объявляются три переменные, содержащие предположительные значения первого минимума (s1), «второго минимума» (s2) и «третьего минимума» (s3).

2) Этим переменным изначально присваиваются значения, заведомо большие любого возможного значения суммы баллов, например 20 (максимальные оценки по трём экзаменам — три пятёрки, их максимальная сумма — 15) .

3) Просматриваются все значения сумм баллов для каждого ученика (вторые элементы записей для элементов массива от 1 до N).

4) Если текущая сумма баллов меньше первого минимума s1, то:

- это значение суммы станет новым значением s1;
- перед этим прежнее значение s1 станет новым значением «второго минимума» s2;
- перед этим прежнее значение s2 станет новым значением «третьего минимума» s3.

5) Иначе, если текущая сумма баллов больше «второго минимума» s2, то:

- это значение суммы станет новым значением s2;
- перед этим прежнее значение s2 станет новым значением «третьего минимума» s3.

6) Иначе, если текущая сумма баллов больше «третьего минимума» s3, то это значение суммы станет новым значением s3.

Результат: «третий минимум» будет храниться в переменной s3.

5. Вывод данных: просмотр массива записей (учеников от 1 до N). Если у очередного ученика значение суммы баллов (второй элемент записи) меньше или равен ранее вычисленному «третьему минимуму» (s3), то выводится фамилия и имя ученика (значение первого элемента записи).

Полный текст программы:

```
var p: array[1..100] of record // массив записей
    name: string;
    sum: integer;
   end;
    c: char;
    i, j, N, s1, s2, s3, m: integer;
begin
  readln(N); // считано количество учеников
  for i := 1 to N do // чтение строк об учениках
begin
 p[i].name := ''; // «обнуление» первого элемента записи
 repeat
   read(c);
```

```
    p[i].name := p[i].name + c
  until c = ' '; // считана фамилия (включая пробел за ней)
  repeat
   read(c);
   p[i].name := p[i].name + c
  until c = ' '; // считано имя
  p[i].sum := 0; // обнуление второго элемента записи
  for j := 1 to 3 do // чтение оценок за три экзамена
   begin
    read(m); // считана очередная оценка
    p[i].sum := p[i].sum + m // вычисление суммы
   end; // подсчитана сумма баллов
   readln; // переход на следующую входную строку
end;
// вычисление «третьего минимума»:
s1 := 20; s2 := 20; s3 := 20;
for i := 1 to N do
 begin
  if p[i].sum < s1 then
   begin
    s3 := s2; s2 := s1;
    s1 := p[i].sum
   end else
  if p[i].sum < s2 then
   begin
    s3 := s2; s2 := p[i].sum
   end else
  if p[i].sum < s3 then s3 := p[i].sum;
end;
 // вывод результата: если сумма баллов у очередного ученика меньше
 // или равна найденному «третьему минимуму», то выводится его фамилия
 // и имя
 for i := 1 to N do
   if p[i].sum <= s3 then writeln(p[i].name);
end.
```

Программирование

C4 Задачи на анализ и обработку данных: разбор задач (окончание)

Конспект

Пропуск текстовых данных из потока символов

а) использование «буферной» переменной соответствующего типа, которая далее не используется (при чтении следующего ненужного данного используется та же переменная);

б) для пропуска текстовых данных в цикле считывать из входной строки отдельные символы, включая пробел, завершающий очередное данное. Для пропуска ещё одного такого данного повторно строится такой же цикл.

Массивы-счётчики

Для подсчёта количеств нескольких объектов используется *массив-счётчик* размера, соответствующего количеству объектов. При обнаружении в потоке входных данных объекта, соответствующего условию, нужно увеличить на 1 значение ячейки массива-счётчика, соответствующей данному объекту.

Индексные массивы

Если в качестве индексов массива-счётчика невозможно (или не рационально) использовать значения, считываемые из входных данных в качестве идентификатора объектов, то применяется *индексный массив*:

1) массив-счётчик объявляется по количеству обрабатываемых объектов;

2) отдельно объявляется массив того же размера, который содержит обозначения соответствующих объектов;

3) при считывании входных данных соответствующее обозначение объекта ищется в индексном массиве (путём его просмотра с начала до конца); номер ячейки индексного массива, содержащей искомое обозначение объекта, запоминается;

4) если данный объект соответствует требуемому условию, то в массиве-счётчике увеличивается на 1 элемент, индекс которого равен запомненному индексу элемента в индексном массиве, содержащего обозначение этого объекта.

Разбор типовых задач

Задача 1*. На вход программе подаются сведения о номерах школ учащихся, участвовавших в олимпиаде. В первой строке сообщается количество учащихся N, каждая из следующих

N строк имеет формат: <Фамилия> <Инициалы> <номер школы>, где <Фамилия> — строка, состоящая не более чем из 20 символов, <Инициалы> — строка, состоящая из 4-х символов (буква, точка, буква, точка), <номер школы> — не более чем двузначный номер. <Фамилия> и <Инициалы>, а также <Инициалы> и <номер школы> разделены одним пробелом.

Пример входной строки:

Иванов П.С. 57

Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет выводить на экран информацию, из какой школы было меньше всего участников (таких школ может быть несколько). При этом необходимо вывести информацию только по школам, пославшим хотя бы одного участника.

Следует учитывать, что $N \geqslant 1000$.

Решение

1. По условию номер школы — не более чем двузначный, тогда максимальное значение номера школы равно 99. Требуется произвести подсчёт количества участников от каждой школы, следовательно требуется массив-счётчик с индексами от 1 до 99.

```
var nc: array[1..99] of integer;
```

Необходимость использовать индексный массив отсутствует.

2. Входные данные: вначале считывается количество строк N, затем в цикле с параметром считываются сами эти строки.

Фамилия и инициалы участника — лишние данные, их нужно пропустить путём посимвольного чтения до завершающего пробела включительно.

Номер школы считывается в переменную целого типа.

3. Обработка считанных данных: увеличивается на 1 значение элемента массива-счётчика, индекс которого равен считанному номеру школы.

4. Обработка полученного массива-счётчика: стандартный алгоритм поиска минимума в одномерном массиве для ненулевых элементов.

5. Вывод данных: просмотр всего массива-счётчика и вывод индексов его элементов, значения которых равны найденному минимуму.

Полный текст программы:

```
var nc: array[1..99] of integer;
    p: 1..99;
    c: char;
    i, k, N, min: integer;
begin
  for i := 0 to 99 do nc[i] := 0; // обнуление массива-счетчика
  readln(N); // считано количество строк
  for i := 1 to N do // чтение строк данных
   begin
    repeat
     read(c)
   until c = ' '; // пропущена фамилия
   repeat
     read(c)
   until c = ' '; // пропущены инициалы
   readln(p); // считан номер школы
   nc[p] := nc[p] + 1; // увеличение счетчика для данной школы
end;
// поиск минимума
  min := N;
```

```
  // максимально возможное значение количества участников — N, если все они
  // от одной и той же школы
  for i := 1 to 99 do
    if (nc[i] > 0) and (nc[i] < min) then min := nc[i];
   // если текущее значение больше нуля и меньше предполагаемого минимума,
    // то назначить его как новый минимум
    // вывод результатов
  for i := 1 to 99 do
    if nc[i] = min then
      writeln(i);
end.
```

Задача 2*. На автозаправочных станциях (АЗС) продаётся бензин с маркировкой 92, 95 и 98. В городе N был проведён мониторинг цены бензина на различных АЗС.

Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет определять для каждого вида бензина, сколько АЗС продают его дешевле всего. На вход программе в первой строке подаётся число данных о стоимости бензина. В каждой из последующих N строк находится информация в следующем формате:

<Компания> <Улица> <Марка> <Цена>,

где <Компания> — строка, состоящая не более, чем из 20 символов без пробелов, <Улица> — строка, состоящая не более, чем из 20 символов без пробелов, <Марка> — одно из чисел — 92, 95 или 98, <Цена> — целое число в диапазоне от 1000 до 3000, обозначающее стоимость одного литра бензина в копейках. <Компания> и <Улица>, <Улица> и <Марка>, а также <Марка> и <цена> разделены ровно одним пробелом.

Пример входной строки:

Синойл Цветочная 95 2250

Программа должна выводить через пробел 3 числа — количество АЗС, продающих дешевле всего 92-й, 95-й и 98-й бензин соответственно. Если бензин какой-то марки нигде не продавался, то следует вывести 0.

Пример выходных данных:

12 1 0

Решение

1. Необходимо определять минимальную цену на бензин каждого вида и количество таких значений. Это можно делать сразу при обработке введенных данных без организации массива-счётчика.

Вместо этого удобно определить два массива (взаимосвязанных через индексы элементов), в элементах которых будут храниться соответственно текущее минимальное значение цены бензина соответствующей марки (массив min) и количество элементов, равных этому минимуму (массив ans).

```
var min, ans: array[92..98] of integer;
```

При этом принципы работы с этими массивами аналогичны работе с массивом-счётчиком: считанный номер марки бензина используется как индекс соответствующего элемента в массиве. (Массивы объявлены на 7 элементов каждый, при этом в каждом используется только по три элемента.)

2. Входные данные: сначала отдельно считывается количество строк N, а затем в цикле с параметром считываются сами строки.

При этом строковые данные <компания> и <улица> — лишние, их нужно пропустить путем посимвольного чтения до завершающего пробела включительно.

Номер бензина и цена (целые числа) считываются в соответствующие переменные.

3. Обработка считанных данных: считанное значение номера марки бензина используется как индекс и определяет, с какой парой переменных (предполагаемый минимум и счётчик значений, равных ему) работать в данном случае. Считанное значение цены обрабатывается в типовом алгоритме поиска минимума с подсчётом количества значений, равных минимуму, аналогично элементу массива:

1) цена, по условию, не превышает 3000, поэтому при начальной инициализации ячеек массива min в качестве предполагаемых минимумов задаётся константа, заведомо большая максимально возможного значения цены (число 3001 или большее); счётчики количества значений, равных минимальному (массив ans), обнуляются.

Все эти инициализационные действия производятся в цикле (индексы массивов меняются от 92 до 98) **до** начала цикла чтения входных строк.

2) после чтения номера марки бензина k и цены b выполняется проверка:
если $b <$ min[k], то:

- значение b запоминается как новое значение предполагаемого минимума min[k];
- значение счётчика ans[k] устанавливается в 1 (т.е. одно такое значение уже найдено);

В зависимости от номера марки бензина, работа ведётся с соответствующей парой ячеек «минимум — счётчик».

3) иначе если значение b равно текущему значению минимума для данной марки бензина min[k], то значение счётчика ans[k] увеличивается на 1.

Если бензина какой-то марки не было вообще, то значение соответствующего счётчика не изменится (останется равным нулю).

По завершении обработки входных данных в ячейках ans[92], ans[95] и ans[98] определились искомые количества встреченных значений цен, равных минимальным (для каждой из трёх этих марок), либо нули.

4. Вывод данных: на экран выводятся полученные значения ans[92], ans[95] и ans[98].

Полный текст программы:

```
program C4;
var min, ans: array[92..98] of integer;
    c: char;
    i, k, N, b: integer;
begin
  for i := 92 to 98 do // инициализация начальных значений
   begin // предполагаемых минимумов и счетчиков
    min[i] := 3001;
    ans[i] := 0;
   end;
  readln(N); // считано количество строк
  for i := 1 to N do // чтение строк входных данных
   begin
    repeat
      read(c);
    until c = ' '; // пропуск названия компании
    repeat
      read(c);
```

```
    until c = ' '; // пропуск названия улицы
    readln(k, b); // чтение номера марки бензина и его цены
  // поиск минимума с подсчетом количества элементов, равных этому минимуму
  if b < min[k] then
    begin
     min[k] := b;
     ans[k] := 1
    end else
   if min[k] = b then ans[k] := ans[k]+1;
  end;
  // вывод результатов
  writeln(ans[92],' ', ans[95],' ', ans[98])
end.
```

Задача 3*. На вход программе подаётся набор символов, заканчивающийся точкой (в программе на языке Бейсик символы можно вводить по одному в строке, пока не будет введена точка, или считывать данные из файла). Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая сначала будет определять, есть ли в этом наборе символы, соответствующие десятичным цифрам. Если такие символы есть, то можно ли переставить их так, чтобы полученное число было симметричным (читалось одинаково как слева направо, так и справа налево). Ведущих нулей в числе быть не должно, исключение — число 0, запись которого содержит ровно один ноль.

Если требуемое число составить невозможно, то программа должна вывести на экран слово «NO». А если возможно, то в первой строке следует вывести слово «YES», а во второй — искомое симметрическое число. Если таких чисел несколько, то программа должна выводить максимальное из них. Например, пусть на вход подаются следующие символы:

Do not 911 to 09 do.

В данном случае программа должна вывести

YES

91019

Решение

1. Необходимо вычислять количества каждой из цифр от 0 до 9, поэтому объявляется массив-счётчик с индексами элементов от '0' до '9'.

```
var a: array['0'..'9'] of integer;
```

Внимание! Индексы элементов массива — это символы '1' .. '9', а не числа 1 .. 9 (либо при использовании в качестве индексов именно чисел нужно считанные из входной строки символы цифр преобразовывать в их числовые значения).

Поскольку в качестве индексов массива-счётчика используются сами считываемые обозначения объектов, использовать индексный массив не требуется.

2. Входные данные: считываются посимвольно до завершающей точки (цикл с предусловием).

3. Обработка считанных данных: если считанный символ представляет собой цифру, то в массиве-счётчике значение элемента с индексом увеличивается на 1, равным этому символу цифры.

4. Обработка полученного массива-счётчика.

В каких случаях можно построить максимальное по величине симметричное число? Очевидно, что это возможно в трёх случаях:

- если количества всех цифр чётны;

- если есть только одна цифра, количество которой нечётно (цепочка этих цифр будет в полученном симметричном числе стоять в середине);
- если не имеется никаких цифр, кроме нулей, то симметричное число можно построить только если этот нуль — единственный.

Исходя из анализа этих условий и решается задача[1]:

1) просматривая массив-счётчик, подсчитывается в нём (в переменной k) количество цифр, встречающихся нечётное число раз. При этом в переменной c_odd запоминается символ цифры, которая последней встретилась нечётное число раз, а в переменной s независимо от чётности количества каждой цифры подсчитывается суммарное количество всех цифр (оно понадобится позже для проверки другого условия).

```
k := 0;
for c := '0' to '9' do
 begin
  if a[c] mod 2 = 1 then
   begin
    k := k + 1;
    c_odd := c
   end;
  s := s + a[c];
 end;
```

2) проверяется условие: количество нулей равно 1, а количества других цифр равны нулю (что понятно из равенства общей суммы количеств всех цифр всё той же единице). Если это истинно, то выводится слово «YES» и число 0.

```
if (a['0'] = 1) and (s = 1) then
 begin
  writeln('YES');
  writeln('0');
 end else
```

3) иначе проверяется условие: если нули — не единственные цифры и при этом $k = 0$ или $k = 1$, то выводится слово «YES» и конструируется число; иначе выводится слово «NO»:

```
if (s <> a['0']) and ((k = 0) or (k = 1)) then
 begin
  writeln('YES');
  <вывод симметричного числа>
 end else writeln('NO');
```

4) симметричное число конструируется так:

- перебираются цифры с 9 до 0 (цикл с параметром, изменяемым в обратном порядке — от '9' до '0'):

```
for c := '9' downto '0' do
```

- выводится цепочка таких цифр (цифра — текущее значение параметра цикла), длина которой равна половине всего количества таких цифр (если это количество нечётно, то дробная часть отбрасывается, т. е. используется целочисленное деление):

Способ 1: с использованием стандартной функции	Способ 2: без использования стандартной функции
`write(StringOfChar(c, a[c] div 2));`	`for i := 1 to a[c] div 2 do` `  write(c);`

[1] В демонстрационном варианте ЕГЭ за 2011 год приведено другое решение, но оно достаточно сложно для понимания. Предлагаемый здесь вариант решения более «прозрачен» для понимания смысла алгоритма.

- **по завершении цикла перебора цифр от 9 до 0, если $k = 1$, то выводится одна цифра, количество которых было нечётно (она запомнена в переменной c_odd):**

```
if k = 1 then write(c_odd);
```

перебираем цифры с 0 до 9:

```
for c := '0' to '9' do
```

- **выводится цепочка таких цифр (цифра — текущее значение параметра цикла), длина которой равна половине всего количества таких цифр (если это количество нечётно, то дробная часть отбрасывается, т. е. используется целочисленное деление):**

Способ 1: с использованием стандартной функции	Способ 2: без использования стандартной функции
`write(StringOfChar(c, a[c] div 2));`	`for i := 1 to a[c] div 2 do` `  write(c);`

Полный текст программы:

```
program C4;
var a: array['0'..'9'] of integer;
    c, c_odd: char;
    i, k, s: integer;
begin
   for c := '0' to '9' do a[c] := 0; // обнуляется массив-счетчик
   read(c); // чтение символов из строки до точки
   while c <> '.' do
    begin
     if c in ['0' .. '9'] then a[c] := a[c] + 1;
    // если считанный символ – цифра, то соответствующий элемент
    // массива-счетчика увеличивается на 1
     read(c);
   end;
// подсчет количества цифр, встречающихся нечетное число раз
 k := 0; s := 0; // начальные обнуления
 for c := '0' to '9' do // просмотр каждой цифры
  begin
   if a[c] mod 2 = 1 then // если ее количество нечетно, то
    begin
      k := k + 1; // увеличить счетчик
      c_odd := c // и запомнить эту цифру
    end;
   s := s + a[c];
   // в любом случае подсчет суммарного количества всех цифр
end;
// проверка первого условия (единственный нуль)
if (a['0'] = 1) and (s = 1) then
 begin
  writeln('YES');
  writeln('0');
  end else
  // иначе проверка второго условия (не более одной цифры встречается
  // нечетное число раз, притом нули – не единственные цифры)
  if (s <> a['0']) and ((k = 0) or (k = 1)) then
```

```
    begin
      writeln('YES');
      // если условие соблюдается, то конструируется число
      for c := '9' downto '0' do
        write(StringOfChar(c,a[c] div 2)); // первая половина
      if k = 1 then write(c_odd); // средний символ
      for c := '0' to '9' do
        write(StringOfChar(c,a[c] div 2)); // вторая половина
    // иначе ответ – «NO»
    end else writeln('NO');
end.
```

Задача 4*. В командных олимпиадах по программированию для решения предлагается не больше 11 задач. Команда может решать предложенные задачи в любом порядке. Подготовленные решения команда посылает в единую проверяющую систему соревнований.

Вам предлагается написать эффективную, в том числе по используемой памяти, программу, которая будет статистически обрабатывать пришедшие запросы, чтобы определить наиболее популярные задачи. Следует учитывать, что количество запросов в списке может быть очень велико, так как многие соревнования проходят с использованием Интернет.

Перед текстом программы кратко опишите используемый вами алгоритм решения задачи.

На вход программе в первой строке подаётся количество пришедших запросов N. В каждой из последующих N строк записано название задачи в виде текстовой строки. Длина строки не превосходит 100 символов, название может содержать буквы, цифры, пробелы и знаки препинания.

Пример входных данных:

6
A+B
Крестики-Нолики
Прямоугольник
Простой делитель
A+B
Простой делитель

Программа должна вывести список из трёх наиболее популярных задач с указанием количества запросов по ним. Если в запросах упоминается менее трёх задач, то выведите информацию об имеющихся задачах. Если несколько задач имеют ту же частоту встречаемости, что и третья по частоте встречаемости задача, их тоже нужно вывести.

Пример выходных данных для приведённого выше примера входных данных:

A+B 2
Простой делитель 2
Крестики-Нолики 1
Прямоугольник 1

Решение

1. Требуется подсчёт количества каждой из задач. Всего задач может быть 11 видов. Следовательно, необходим массив-счётчик на 11 ячеек. Поскольку названия задач не могут являться индексами массива-счётчика, требуется также индексный массив из 11 ячеек, который должен хранить названия этих задач.

```
Var Count: array[1..11] of integer; // массив-счетчик
    Names: array[1..11] of string; // индексный массив
```

Однако есть дополнительная сложность: названия задач, которые нужно запоминать, и даже их реальное количество неизвестно. Поэтому требуется *динамически формировать индексный массив.*

2. Входные данные: вначале отдельно считывается количество строк, а затем в цикле с параметром вводятся строки — названия задач.

3. Обработка входных данных:

1) в индексном массиве путём просмотра с его начала количества заполненных его элементов (это количество изначально равно нулю) ищется элемент, равный считанному названию задачи;

```
Num := 0;
// количество заполненных элементов индексного массива первоначально
// равно нулю
...
 j := 1; // просмотр индексного массива с начала
 while (j <= Num) and (s <> Names[j]) do j := j + 1;
 // переходим к следующему элементу индексного массива, пока или
 // не будут просмотрены все его заполненные элементы, или в нем
 // не будет найдено считанное название задачи
```

2) если такой элемент найден, то элемент массива-счетчика с соответствующим индексом (j) увеличивается на 1;

```
if j <= Num then // значит, просмотр завершен раньше, чем просмотрены
// все заполненные элементы, т.е. в ячейке с индексом j найдено
// требуемое название задачи
   Count[j] := Count[j] + 1
```

3) иначе (если такой элемент не найден) индексный массив наращивается на один элемент:

```
else begin // иначе
```

- в текущую (незанятую) ячейку индексного массива с индексом *j* записывается считанное новое название задачи;

```
Names[j] := s;
```

- в соответствующую ячейку (с индексом *j*) массива-счётчика записывается единица (одна такая задача уже встречена);

```
Count[j] := 1;
```

- количество заполненных элементов индексного массива увеличивается на 1.

```
Num := Num + 1
end
```

Результат: заполненные по всем встреченным задачам индексный массив с названиями задач и массив-счётчик с их количествами.

4. Обработка массива-счётчика: требуется определить три вида задач, для которых количества будут наибольшими. Для этого проще всего отсортировать по убыванию массив-счетчик (и синхронно с ним — индексный массив). Используется метод «пузырька» (попарное сравнение элементов и при необходимости их обмен местами); для упрощения решения задачи — с полным просмотром (без контроля досрочного прерывания сортировки, если массив уже отсортирован).

```
for i := Num downto 2 do
  for j := 2 to i do
    if Count[j-1] < Count[j] then
     begin
       t := Count[j]; Count[j] := Count[j-1]; Count[j-1] := t;
       s := Names[j]; Names[j] := Names[j-1]; Names[j-1] := s;
     end;
```

Нужно не забывать, что при перестановке элементов в массиве-счётчике нужно одновременно переставлять соответствующие им элементы индексного массива, чтобы сохранить соответствие этих массивов.

5. Вывод данных:

1) если найдено только три вида задач или меньше, то надо выводить информацию обо всех них (тогда в переменной *j* запоминается номер последней запомненной задачи), иначе (если видов задач больше) — только о трёх первых видах (тогда в переменной *j* запоминается номер 3);

2) массив-счётчик просматривается с начала и до тех пор, пока не исчерпано количество заполненных его элементов (Num) и пока количество задач текущего вида не меньше количества задач того вида, что в отсортированных массивах (счётчике и индексном) расположен месте под номером j. Очевидно, что при этом если различных задач меньше трёх, то будет выведена информация обо всех них, а если их больше трёх, то будет выведена информация о трёх первых по количеству видах задач, а также о следующих задачах (четвёртой и т.д.), если их количество совпадает с количеством задач третьего вида.

```
if Num >= 3 then j := 3 else j := Num;
i := 1;
while (i <= Num) and (Count[i] >= Count[j]) do
 begin
  WriteLn(Names[i], ' ', Count[i]);
  i := i + 1;
end
```

Полный текст программы:

```
Program C4;
Var Count: array[1..11] of integer; // массив-счетчик
    Names: array[1..11] of string; // индексный массив
    n, Num, i, j, t: integer;
    s: string;
Begin
  Num := 0;
  // изначально в индексном массиве не заполнено ни одного элемента
  ReadLn(N); // считано количество строк
  for i := 1 to N do // чтение строк входных данных
   begin
    ReadLn(S); // считано очередное название задачи
    // считанное название задачи ищется в индексном массиве:
    j := 1;
    while (j <= Num) and (s <> Names[j]) do j := j + 1;
    if j <= Num then // если задача найдена в j-й ячейке,
      Count[j] := Count[j] + 1 // то увеличиваем соответствующий счетчик
    else begin // иначе название задачи добавляется в конце индексного
               // массива (в ячейку с индексом j)
      Names[j] := s;
      Count[j] := 1;
      Num := Num + 1
    end
  end;
// сортировка массива-счетчика по убыванию и синхронно с ним — индексного
// массива (методом «пузырька»)
```

```
 for i := N um downto 2 do
   for j := 2 to i do
    if Count[j-1] < Count[j] then
     begin
      t := Count[j]; Count[j] := Count[j-1]; Count[j-1] := t;
      s := Names[j]; Names[j] := Names[j-1]; Names[j-1] := s;
     end;
 // вывод результата
 if Num >= 3 then j := 3 else j := Num;
 i := 1;
 while (i <= Num) and (Count[i] >= Count[j]) do
  begin
   WriteLn(Names[i], ' ', Count[i]);
   i := i + 1;
  end
end.
```

Задачи для самостоятельного решения

1. По итогам ЕГЭ по информатике подготовлена сводная информация с указанием набранного каждым учеником той или иной школы количества баллов. Известно, что ЕГЭ сдавало более 5-ти учеников и что нумерация школ в городе не всегда идёт подряд. Требуется определить школу, учащиеся которой получили наибольший средний балл (с учётом округления до целых), и вывести на экран номер этой школы и средний балл её учащихся.
   Если школ, учащиеся которой набрали наибольший средний балл, больше одной, то надо вывести количество таких школ.
   Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая выводит требуемую информацию.
   На вход программе сначала подаётся число учеников, сдававших ЕГЭ. В каждой из следующих N строк находится информация об учениках в формате:
   <Фамилия> <Имя> <Номер школы> <Количество баллов>,
   где <Фамилия> — строка, состоящая не более чем из 30 символов без пробелов, <Имя> — строка, состоящая не более чем из 20 символов без пробелов, <Номер школы> — целое число в диапазоне от 1 до 99, количество баллов> — целое число в диапазоне от 1 до 100. Эти данные записаны через пробел, причём ровно один между каждой парой (то есть всего по три пробела в каждой строке).
   Пример входной строки:
   Иванов Иван 50 87
   Пример выходных данных:
   50 74
   Другой вариант выходных данных:
   7
2. По итогам ЕГЭ по математике подготовлена сводная информация с указанием набранного каждым учеником той или иной школы количества баллов. Известно, что ЕГЭ сдавало более 5-ти учеников и что нумерация школ в городе не всегда идёт подряд. Требуется определить для каждой школы фамилии учеников, получивших наибольший балл, при условии, что из школы математику сдавали не менее 5 человек (иначе информацию по этой школе выводить на экран не нужно).

Информацию нужно выводить на экран в виде: <Номер школы> <Фамилия ученика> в отдельной строке для каждой школы.

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая решает описанную задачу.

В программу сначала вводится количество учеников, сдававших ЕГЭ, а далее в каждой из N строк содержится информация об учениках в формате:

<Фамилия> <Имя> <Номер школы> <Количество баллов>,

где <Фамилия> — строка, состоящая не более чем из 30 символов без пробелов, <Имя> — строка, состоящая не более чем из 20 символов без пробелов, <Номер школы> — целое число в диапазоне от 1 до 99, <Количество баллов> — целое число в диапазоне от 0 до 100. Эти данные записаны через пробел, причём ровно один между каждой парой (то есть всего по три пробела в каждой строке).

Пример входной строки:

Иванов Иван 50 87

Пример выходных данных:

5 Иванов

50 Петров

74 Сидоров

Ответы для самопроверки

Задача 1.

Решение

1. Нужно определять средние баллы учеников каждой школы. Для этого требуется отдельно подсчитывать суммы баллов учеников каждой школы и количества учеников этих школ. Номер школы — двузначный, т. е. может быть равен числу от 1 до 99. Следовательно, нужно использовать два массива-счётчика с индексами от 1 до 99 для подсчёта сумм и количеств (либо можно использовать один двумерный массив-счётчик).

```
var s, k: array[1..99] of integer;
```

В качестве индексов массива-счётчика используются сами считываемые номера школ. Кроме того, хотя в массивах-счётчиках могут использоваться не все ячейки, список школ заранее не известен, поэтому использовать индексный массив нерационально.

2. Входные данные: вначале считывается количество строк данных N, затем в цикле с параметром считываются сами строки данных.

При этом фамилия и имя ученика — это лишние данные, их нужно пропускать путём посимвольного чтения до завершающего пробела включительно.

Далее номер школы и количество баллов (целые числа) считываются и обрабатываются.

3. Обработка считываемых данных:

1) в первом массиве-счётчике значение элемента с индексом, равным считанному номеру школы, увеличивается на считанное значение баллов (подсчёт суммы баллов всех учеников данной школы);

2) во втором массиве-счётчике значение элемента с индексом, равным считанному номеру школы, увеличивается на 1 (подсчёт количества учеников данной школы).

4. Обработка массива-счётчика:

1) вычисление среднего балла по каждой школе путём целочисленного деления суммы баллов на количество учеников (деление целочисленное, так как по условию задачи требуется определять средние значения с точностью до целых; по этой же причине можно объявить массив-счётчик целого типа, а не вещественного);

2) поиск максимума с одновременным подсчётом количества элементов массива-счётчика, равных максимальному (типовой алгоритм, но переприсваиваются индексы, а сравниваются соответствующие элементы массивов).

5. Вывод результатов: если количество найденных элементов, равных максимальному, равно 1, то вывести номер школы (он равен найденному индексу максимального элемента — max) и средний балл для этой школы (s[max]). Иначе вывести количество элементов, равных максимальному (nmax).

Полный текст программы:

```
program C4;
var s, k: array[1..99] of integer; // массивы-счетчики
    ch: char;
    i, N, sh, ball, max, nmax: integer;
begin
   for i := 1 to 99 do // обнуление массивов-счетчиков
    begin
     s[i] := 0; k[i] := 0
    end;
   readln(N); // считали количество строк
   for i := 1 to N do // считывание строк входных данных
    begin
     repeat
       read(ch)
     until ch = ' '; // пропуск фамилии
     repeat
      read(ch)
     until ch = ' '; // пропуск имени
     readln(sh,ball); // чтение номера школы и балла
     s[sh] := s[sh] + ball; // подсчет суммы баллов для данной школы
     k[sh] := k[sh] + 1 // подсчет количества ее учеников
    end;
   // вычисление среднего балла для школ, для которых есть данные
   for i := 1 to 99 do
     if k[i] > 0 then s[i] := s[i] div k[i];
   // поиск максимума и количества элементов, равных максимуму
   max := 1; nmax := 1;
   // первоначально приняли за максимальный первый элемент,
   // счетчик количества таких элементов равен 1
   for i := 2 to 99 do // просмотр остальных элементов
    if s[i] > s[max] then
    // если текущий элемент больше предполагаемого максимального, то
    begin
     max := i; // этот элемент будет новым максимумом
     nmax := 1 // а счетчик сбрасывается в 1
    end else
   if s[i] = s[max] then
   // иначе если текущий элемент равен максимуму, то
    nmax := nmax + 1; // счетчик таких элементов увеличиваем на 1
end;
// вывод результатов
   if nmax = 1 then writeln(max,' ', s[max]) // для одной школы
   else writeln (nmax); // для нескольких школ
end.
```

Задача 2.

Решение

1. Нужно запоминать для каждой школы:

• максимальный балл учеников;

• имя ученика с максимальным баллом;

• общее количество учеников данной школы.

Для этого требуется три отдельных массива: два массива-счётчика и индексный массив, заполняемый по мере обработки данных. Номер школы — двузначный, т. е. может быть равен от 1 до 99. Следовательно, диапазон индексов этих массивов-счётчиков — от 1 до 99.

```
var num, bal: array[1..99] of integer; // массивы-счетчики
    name: array[1..99] of string[52]; // индексный массив
```

В качестве индексов массива-счётчика используются сами считываемые номера школ.

2. Входные данные: вначале считывается количество строк данных N, затем в цикле с параметром считываются сами строки данных.

При этом фамилия ученика важна и посимвольно считывается в отдельную переменную — «буфер» до завершающего пробела включительно.

Имя ученика — это лишнее данное, оно пропускается путём посимвольного чтения до завершающего пробела включительно.

Далее номер школы и количество баллов (целые числа) считываются и обрабатываются.

3. Обработка считываемых данных:

1) проверяется: если считанное значение баллов больше, чем запомненное ранее для данной школы в элементе массива-счётчика *bal* с индексом, равным считанному номеру школы, то:

• в этот элемент массива-счётчика *bal* записывается новое максимальное значение;

• в элемент массива *name* с индексом, равным номеру школы, записывается фамилия текущего ученика как предположительно набравшего максимальный балл в данной школе;

2) в массиве-счётчике *num* значение элемента с индексом, равным считанному номеру школы, увеличивается на 1 (подсчёт количества учеников данной школы).

4. Обработка массива-счётчика после завершения обработки входных данных не требуется.

5. Вывод результатов:

1) просматриваются все номера школ от 1 до 99;

2) если для данной школы количество учеников больше или равно пяти, то выводится номер школы (параметр цикла), а затем запомненная фамилия ученика для данной школы (из индексного массива).

Полный текст программы:

```
program C4;
var num, bal: array[1..99] of integer; // массивы-счетчики
    name: array[1..99] of string[52]; // индексный массив
    s: string[52]; // «буферная» переменная для фамилии
    ch: char;
    i, N, sh, ball: integer;
begin
  for i := 1 to 99 do // инициализация массивов-счетчиков
  begin
    num[i] := 0; // количества учеников — обнуляются
  bal[i] := -1 // в качестве первого предполагаемого максимума берется
  // значение, заведомо меньшее минимально возможного числа баллов
  end;
 readln(N); // считано количество строк
```

```
for i := 1 to N do // чтение строк входных данных
 begin
  s := ''; // «обнуление» строки для чтения фамилии
  repeat
    read(ch);
    s := s + ch
  until ch = ' '; // считана фамилия
  repeat
    read(ch)
  until ch = ' '; // пропущено имя
  readln(sh, ball); // считаны номер школы и балл
  if ball > bal[sh] then
  // если считанный балл больше, чем ранее запомненный максимум
  // для данной школы, то
  begin
  bal[sh]:= ball; // запоминаем этот балл как новый максимум
                  // для данной школы
   name[sh] := s // запоминаем фамилию ученика с этим баллом
  end;

  num[sh] := num[sh] + 1 // подсчет количества учеников данной школы
 end;

 // вывод результатов
 for i := 1 to 99 do
  if num[i] >= 5 then // если в данной школе сдавало экзамен не менее
                      // пяти учеников, то
   writeln(i,' ',name[i]); // вывести номер школы и фамилию ученика,
                           // запомненную для нее
end.
```

Итоговое тестирование

Выполните все предложенные задания теста в формате ЕГЭ, а затем сравните свои результаты с приведёнными после теста правильными ответами. В таблице правильных ответов указаны номера занятий, которые необходимо повторить в случае неправильного решения той или иной задачи.

Тест

А1. Сколько нулей содержится в 16-разрядной двоичной записи числа 11512_{10}?
1) 6 2) 7 3) 8 4) 9

А2. Населённые пункты Ачинск, Беломорск, Ванино, Гурово и Дмитровск связывает железная дорога. Длительность поездок между городами (в часах) приведена в таблице. (Отсутствие числа в таблице означает, что железнодорожного сообщения между соответствующими пунктами нет.)

	Ачинск	Беломорск	Ванино	Гурово	Дмитровск
Ачинск		3	18		
Беломорск	3		12		5
Ванино	18	12		3	6
Гурово			3		5
Дмитровск		5	6	5	

Определите наименьшее возможное время поездки (без учёта времени на пересадки и ожидание поездов) между Ачинском и Ваниным (передвигаться можно только по указанным железнодорожным маршрутам).
1) 9 2) 14 3) 15 4) 18

А3. Дан фрагмент таблицы истинности выражения F:

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

Какое выражение может соответствовать F?
1) $(A \to B) \vee (B \to C)$
2) $A \wedge (B \to C)$
3) $(A \to B) \wedge (B \to C)$
4) $A \to B \wedge C$

А4. Для групповых операций с файлами используются *маски имён файлов*. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которой также могут встречаться следующие символы:

- символ «?» (вопросительный знак) — означает ровно один произвольный символ;
- символ «*» (звёздочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность.

В каталоге находятся шесть файлов:
regata.docx
stakkato.docx
baton.docx
ats.doc
kati.rocx
matt.acx

Какая из масок позволяет выбрать из них указанную группу файлов:
regata.docx
stakkato.docx
kati.rocx

1) ?*at?.??cx 2) *at?.??cx 3) *at?.?oc? 4) *at*.??cx

А5. Автомат получает на вход два трёхзначных восьмеричных числа. По этим числам строится новое восьмеричное число по следующим правилам.

1. Вычисляются три числа — абсолютное значение (модуль) разности старших разрядов заданных трёхзначных чисел, абсолютное значение разности средних разрядов этих чисел и абсолютное значение разности младших разрядов.

2. Полученные три числа записываются друг за другом в порядке убывания (без разделителей).

Пример. Исходные трёхзначные числа: 575, 232. Вычисленные значения: 3, 4, 3. Результат: 433

Определите, какое из следующих чисел может быть результатом работы подобного автомата.

1) 357 2) 675 3) 874 4) 631

А6. В фрагменте базы данных представлены сведения о родственных отношениях. Определите на основании приведённых данных фамилию и инициалы внука Пановой Л.А.

1) Кузнецов Л.Б.
2) Рыкунов Н.С.
3) Кузнецов И.А.
4) Кузнецов Б.И.

Таблица 1

ID	Фамилия_И.О.	Пол
71	Рыкунов Н.С.	М
85	Кузнецов И.А.	М
13	Кузнецова А.И.	Ж
42	Панова Л.А.	Ж
23	Рыкунова Е.Н.	Ж
96	Кузнецова И.Б.	Ж
82	Кузнецов Б.И.	М
95	Кузнецова О.Б.	Ж
10	Кузнецов Л.Б.	М
...		

Таблица 2

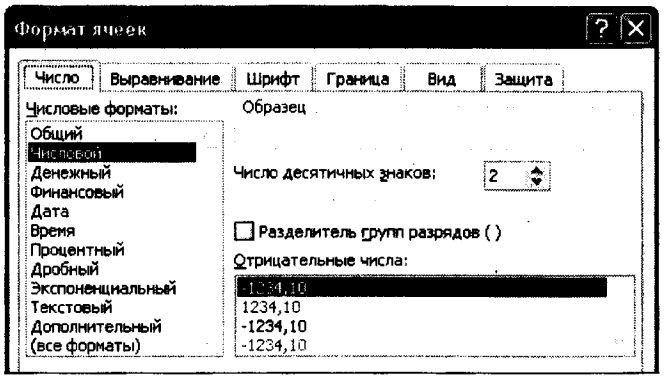
ID_Родителя	ID_Ребёнка
71	23
85	13
85	82
42	13
42	82
23	96
23	10
82	10
82	95
...	...

А7. В стране коротышек три города: Цветочный город, Солнечный город и Ягодный город организовали соревнование — кто больше соберёт арбузного сока. Для учёта результатов соревнования Знайка подготовил электронную таблицу с информацией о том, сколько арбузного сока было собрано жителями каждого города и за какое время. В отдельном столбце под-

считывалась средняя производительность труда одного жителя каждого города. Однако Незнайка по ошибке удалил в таблице некоторые ячейки.

	A	B	C	D	E
1	Город	Кол-во жителей (чел.)	Добыто арбузного сока (л)	Кол-во рабочих дней	Средняя производительность труда
2	Цветочный	45	16,5	12	
3	Солнечный	58	16	8	
4	Ягодный	53	18	10	
5	ВСЕГО				

Определите, какое значение должно стоять в ячейке **E5**, если для столбца **E** был установлен следующий числовой формат:



1) 1,2 2) 1 3) 0,97 4) 0,975

A8. Фотовидеоцентр оказывает услуги населению по оцифровке старых кинопленок. Симпсоны хотели бы оцифровать и записать на двухслойном DVD ёмкостью 8 Гб кино, отснятое своим дедушкой, при этом озвучив его. Длительность фильма — 10 минут. Кинокамера снимала 25 кадров в секунду. При оцифровке будет выбрано разрешение 800×600 с 256 оттенками серого с сохранением частоты кадров по оригиналу. При озвучивании предполагается записать с микрофона комментарии папы и мамы Симпсонов в течение всего фильма со стереозвуком с частотой дискретизации 16 кГц и 24-битным разрешением. При записи в файл выбраны видео- и аудиокодек, которые не осуществляют никакого сжатия информации, а просто записывают видео с добавлением к нему звука.

Уместится ли на том же самом DVD оцифрованная коллекция семейных фотографий Симпсонов, если эта коллекция включает 2000 цифровых фотографий в формате bmp с разрешением 640×480 и глубиной цвета 16 бит (формат bmp сохраняет информацию без сжатия, но включает в каждый файл служебную информацию объёмом 512 байт)? Если да, то сколько на DVD останется (приблизительно) свободного места?

1) Да; 1,2 Гб 2) Да; 90 Мб 3) Да; 30 Мб 4) Нет

A9. Шпион Штампф передаёт разведданные кодом, в котором используется двойной шифр: сообщение вначале преобразуется в последовательность букв A, B, C, D, E, а затем эти буквы передаются по радио в виде «морзянки» из коротких и длинных сигналов («точек» и «тире»). При этом для букв A, B, C и D коды определены разведцентром:

A	B	C	D
—	···	··—	·——

Для передачи буквы E Штампф должен придумать и сообщить в разведцентр свой код, причём длина этого кода должна быть наименьшей из возможных, а код должен обеспечивать однозначное декодирование любого сообщения.

Какой из представленных ниже кодов может выбрать Штампф?

1)	2)	3)	4)
··	·—	— —	·—·

А10. Ниже приведены несколько кличек собак и кошек. Известно, что только собачьи клички удовлетворяют следующему логическому условию:
(Первая буква гласная → последняя буква согласная) ∧ (предпоследняя буква гласная → вторая буква согласная).

Кто из перечисленных животных — собака?

1) БАРИН 2) АНГАРА 3) АЯКС 4) ЕФРОСИНИЯ

А11. При регистрации электронного почтового ящика в некоторой организации приняты следующие правила: пароль должен иметь длину не менее 6 и не более 8 символов и может состоять из заглавных латинских букв A, B, C, E, H, K, M, O, P, T, X, десятичных цифр и знака подчеркивания (никакие другие символы не допускаются).

В базе данных, которую формирует системный администратор, под хранение каждого такого пароля отводится минимально достаточное и всегда одинаковое целое количество байтов, при этом используется посимвольное кодирование, а все символы кодируются одинаковым и минимально возможным количеством битов.

Определите объём памяти, который потребуется для хранения 128 таких паролей.

1) 2 Кб 2) 3000 байт 3) 4000 байт 4) 4 Кб

А12. В программе используется одномерный целочисленный массив A с индексами от 0 до 9. Ниже представлен фрагмент программы, записанный на языках Паскаль и Бейсик, в котором значения элементов сначала задаются, а затем меняются.

Паскаль

```
for j := 0 to 9 do
  A[j] := j
for k := 0 to 4 do
begin
  t := A[k];
  A[k] := A[4-k];
  A[4-k] := t;
end;
```

Бейсик

```
For j = 0 To 9
  A.SetValue(j, j)
Next
For k = 0 To 4
  T = A.GetValue(k)
  A.SetValue(A.GetValue(4-k), k)
  A.SetValue(T, 4-k)
Next
```

Чему будут равны элементы этого массива после выполнения фрагмента программы?

1) 0 1 2 3 4 5 6 7 8 9 3) 4 3 2 1 0 5 6 7 8 9
2) 9 8 7 6 5 4 3 2 1 0 4) 0 1 2 3 4 9 8 7 6 5

А13. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	вниз	влево	вправо

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

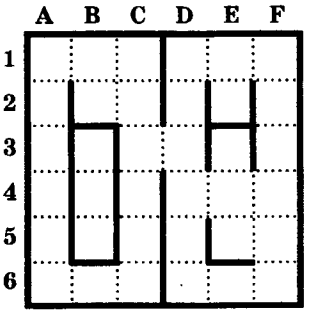
Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно	снизу свободно	слева свободно	справа свободно

Цикл ПОКА <условие> команда выполняется, пока условие истинно, иначе происходит переход на следующую строку. Если РОБОТ начнёт движение в сторону стены, то он разрушится и программа прервётся.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ уцелеет и остановится в той же клетке, с которой он начал движение?

```
НАЧАЛО
ПОКА <сверху свободно> вправо
ПОКА <справа свободно> вниз
ПОКА <снизу свободно> влево
ПОКА <слева свободно> вверх
КОНЕЦ
```



1) Ни одной 2) 1 3) 2 4) 4

A14. Определите, какое число будет напечатано в результате работы следующей программы (для Вашего удобства программа представлена на языках Паскаль и Бейсик):

Бейсик

```
Module A14
   Sub Main()
   Dim d, a, b, t, M, R As Double
            a = -5; b := 5
            d = 0.5
            t = a; M = a; R = F(a)
            While t < b
                 If F(t) < R Then
                    M = t
                    R = F(t)
                 End If
                 t = t + d
            End While
            Console.Write(M)
         End Sub

Function F(ByVal x As Double) As
                              Double
      Return x * (x - 4)
      End Function

End Module
```

Паскаль

```
Program A14;
Uses crt;
Var d, a, b, t, M, R: real;
Function F(x: real) : real;
  begin
     F := x * (x - 4);
  end;
BEGIN
  a := -5; b := 5;
  d := 0.5;
  t := a; M := a; R := F(a);
  while t < b do
    begin
       if (F(t) < R) then
        begin
           M := t;
           R := F(t);
        end;
       t := t + d;
    end;
  write(M);
END.
```

1) 0 2) 2 3) 4 4) −4

B1. Автоматическое устройство осуществило перекодировку информационного сообщения на русском языке длиной в 40 символов, первоначально записанного в 8-битной кодировке КОИ-8R, в 4-байтный код Unicode-32. На сколько бит увеличилась длина сообщения? В ответе запишите только число.

Ответ: _____.

B2. У исполнителя Пятёрочка две команды, которым присвоены номера:

1. прибавить 1,

2. умножить на 5.

Первая из них увеличивает число на экране на 1, вторая — увеличивает его в пять раз.

Запишите порядок команд в программе преобразования числа 2 в число 230, содержащей не более 5 команд, указывая лишь номера команд. (Например, **21121** — это программа

умножить на 5
прибавить 1
прибавить 1
умножить на 5
прибавить 1

которая преобразует число 3 в 86.)

(Если таких программ более одной, то запишите любую из них.)

Ответ: _____.

B3. Определите, что будет напечатано в результате работы следующего фрагмента программы:

Паскаль	Бейсик

`Var x, z: integer;`
`BEGIN`
`  z := 1; x := 0;`
`  while z < 100 do`
`  begin`
`  z := z + 5;`
`  x := 2 * x + 1;`
`  end;`
`  write(x);`
`END.`
	`Dim x, z As Integer`
	`z = 1`
	`x = 0`
	`While z < 100`
	`  z = z + 5`
	`  x = 2 * x + 1`
	`End While`
	`Console.Write(x)`

Ответ: _____.

B4. Все 5-буквенные слова, составленные из букв B, E, C, записаны в алфавитном порядке. Вот начало списка:

0. BBBBB
1. BBBBE
2. BBBBC
3. BBBEB

.....

На каком месте списка стоит слово **EECEB**? Запишите только номер места (по списку).

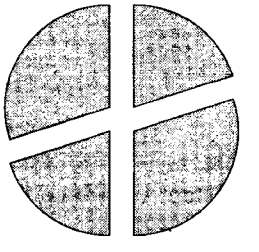
Ответ: _____.

B5. Дан фрагмент электронной таблицы:

	A	B	C	D
1	10	10		10
2	=2*A1+B1	=2*A2	=D1+C1+B1	=7*C1-D1

Какое число должно быть записана в ячейке **C1**, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек **A2:D2** соответствовала рисунку:

Ответ: _____.



B6. Определите значение переменной *z* после выполнения следующего фрагмента программы (записанного ниже на разных языках программирования):

Паскаль	Бейсик

x := 55;
y := 7;
x := (x div y) + (x mod y);
if x < y then
  z := y - x
else
  z := x + y;
	x = 55
	y = 7
	x = (x \ y) + (x MOD y)
	If x < y Then
	  z = y - x
	Else
	  z = x + y
	End If

Ответ: _____.

B7. Ниже приведён текст программы, записанный на языке программирования Паскаль. Что будет напечатано в результате выполнения этой программы?

```
Program Task;
Uses crt;
const L = 5;
type
    atype = array [1..L] of integer;
Var R: atype;
    N, p: integer;

Procedure Modifikator(L, p: integer; var R: atype);
var i, n, t: integer;
begin
p := 1;
for i := 1 to L do
 begin
  t := 2 * R[i] - p;
  R[i] := (R[i] + t * p) mod i;
  p := (R[i] + t) div 2;
 end;
end;

Function Schetchik (L: integer; R: atype) : integer;
var
    N, i, T: integer;
begin
  N := 100;
  T := 3;
  for i := 1 to L do
   begin
    N := N - T * R[i];
    T := (N mod 3) + 2 * i;
   end;
  N := N mod T;
  Schetchik := N;
end;

BEGIN
  R[1] := 1; R[2] := 1; R[3] := 1; R[4] := 1; R[5] := 1;
  Modifikator(L, p, R);
```

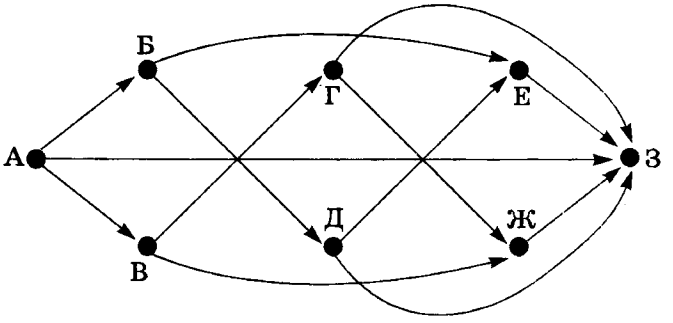
```
  if (p = 0) then
   begin
    write('Некорректные данные');
    halt;
   end;
  N := Calc3(L,R);
  write(N);
  writeln;
END.
```

Ответ: _____.

B8. Запись числа 10100010_2 в системе счисления с основанием N оканчивается на 1 и содержит 3 цифры. Чему равно основание этой системы счисления N?

Ответ: _____.

B9. На рисунке показана схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж и З. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Сколько существует различных путей из города **А** в город **З**?



Ответ: _____.

B10. У Насти есть доступ в Интернет по высокоскоростному одностороннему спутниковому радиоканалу, обеспечивающему скорость получения информации 2^{20} бит в секунду, при этом информация передаётся порциями по 4 Мб. Для получения очередной порции компьютер Насти отправляет запрос объёмом 64 байта по низкоскоростному каналу GPRS, обеспечивающему скорость 2 Кбит/с.

У Лены нет скоростного доступа в Интернет, но есть возможность получать информацию от Насти по телефонному каналу со средней скоростью 2^{10} бит в секунду, при этом информация передаётся единым массивом (кроме пауз, если компьютер Насти ещё не готов передавать информацию).

Лена договорилась с Настей, что она скачает для неё файл объёмом 120 Мбайт по высокоскоростному каналу и ретранслирует их Лене по низкоскоростному каналу.

Компьютер Насти может начать ретрансляцию данных только после того, как им будут получены первые 1 Мбайт этих данных.

Каков минимально возможный промежуток времени (в секундах) с момента начала скачивания данных Настей и до полного их получения Леной?

В ответе укажите только число, слово «секунд» или букву «с» добавлять не нужно.

Ответ: _____.

B11. В терминологии сетей TCP/IP (версия IP v.4) маской подсети называется 32-разрядное двоичное число, определяющее, какие именно разряды IP-адреса компьютера являются общими для всей подсети, — в этих разрядах маски стоит 1. Обычно маски записываются в виде четвёрки десятичных чисел — по тем же правилам, что и IP-адреса.

Некоторая подсеть объединяет 2000 компьютеров и иных сетевых устройств. Определите, какова должна быть маска этой подсети, чтобы каждый компьютер или сетевое устройство имел уникальный IP-адрес в сети Интернет, и определите адрес данной подсети, если один из её компьютеров получил в результате IP-адрес 118.234.115.248.

Примечание. В каждой подсети существует два специальных IP-адреса, которые не могут быть использованы для компьютеров или иных сетевых устройств: это адреса, у которых все биты, кроме относящиеся к адресу всей подсети в целом, равны нулю или все такие биты равны единице.

В качестве ответа запишите полученный адрес подсети в десятичном формате.

Ответ: _____.

B12. В языке запросов поискового сервера для обозначения логической операции «ИЛИ» используется символ «|», а для логической операции «И» — символ «&». Операция логического отрицания («НЕ») обозначается символом «~».

В таблице приведены запросы и количество найденных по ним страниц для некоторого сегмента сети Интернет.

Запрос	Найдено страниц (в тысячах)
Пушкин	750
Пушкин & Лермонтов	220
Пушкин | Лермонтов	1080

Какое количество страниц (в тысячах) будет найдено по запросу:

Пушкин & ~(Лермонтов) ?

Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Ответ: _____.

B13. У исполнителя Кузнечик две команды:

1. Умножь на 2,

2. Вычти 5.

Первая из них увеличивает число вдвое, вторая — уменьшает его на 5 (отрицательные числа допускаются).

Программа для Кузнечика — это последовательность команд. Сколько различных чисел можно получить из числа 3 с помощью программы, которая содержит не более 4 команд?

Ответ: _____.

B14. Сколько различных решений имеет система уравнений:

$(x_1 \equiv x_2) \to (x_3 \equiv x_4) = 1$

$(x_3 \equiv x_4) \to (x_5 \equiv x_6) = 1$

$(x_5 \equiv x_6) \to (x_7 \equiv x_8) = 1$

$(x_7 \equiv x_8) \to (x_9 \equiv x_{10}) = 1$

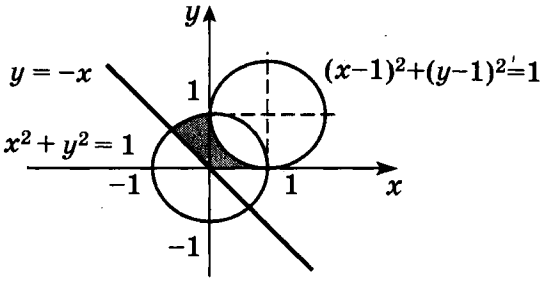
$x_1 \wedge x_{10} = 1$

где $x_1, x_2, \ldots, x_{10}$ — логические переменные?

В ответе **не нужно** перечислять все различные наборы значений $x_1, x_2, \ldots, x_{10}$, при которых выполнена данная система равенств. В качестве ответа нужно указать только количество таких наборов.

Ответ: _____.

C1. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y — действительные числа) и определяется принадлежность этой точки заданной закрашенной области (включая границы).



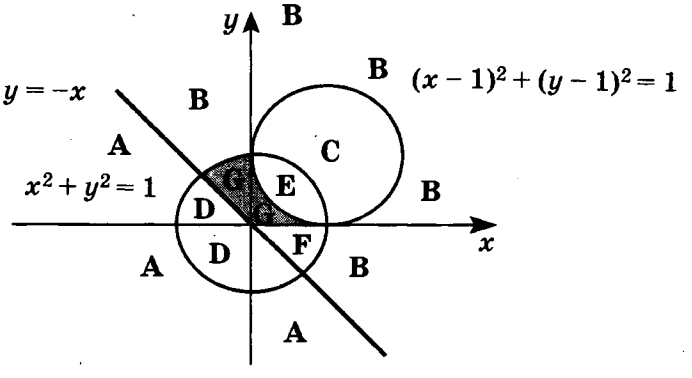
Программист торопился и написал программу неправильно.

Паскаль	Бейсик

var x, y: real;
begin
readln(x, y);
if y >= -x then
if (x-1)*(x-1)+(y-1)*(y-1)>=1
then
if (x * x + y * y <= 1) then
write('принадлежит')
else
write('не принадлежит')
end.
	Module Program
	Sub Main()
	Dim x, y As Double
	x = Val(Console.ReadLine())
	y = Val(Console.ReadLine())
	If y >= -x Then
	If (x-1)*(x-1)+(y-1)*(y-1)>=1
	Then
	If x * x + y * y <= 1 Then
	Console.Write("принадлежит")
	Else
	Console.Write("не принадлежит")
	End If
	End If
	End If
	End Sub
	End Module

Последовательно выполните следующее.

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F и G).



Область	Условие 1 (y >= −x)	Условие 2 ((x−1)²+(y−1)²>=1)	Условие 3 (x²+y²<=1)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					
G					

В столбцах условий укажите «да», если условие выполнится, «нет», если условие не выполнится, «—» (прочерк), если условие не будет проверяться, «неизв.», если программа ведёт себя по-разному для разных значений, принадлежащих данной области. В столбце «Программа выведет» укажите, что программа выведет на экран. Если программа ничего не выводит, запишите «—» (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «неизв.». В последнем столбце укажите «да» или «нет».

2. Приведите пример таких чисел x, y, при которых программа неверно решает поставленную задачу.

3. Укажите, как нужно доработать программу, чтобы не было случаев её неправильной работы. (Это можно сделать несколькими способами, достаточно указать любой способ доработки исходной программы.)

С2. Дан целочисленный массив из 30 элементов, которые могут принимать целые значения от −100 до 100. Опишите на русском языке или на одном из языков программирования алгоритм, позволяющий найти и вывести максимальное значение среди элементов массива, которые имеют чётное значение индекса, но сами нечётны и не кратны трём, а также общее количество элементов массива, удовлетворяющих этим условиям. (Гарантируется, что в исходном массиве есть хотя бы один такой элемент.)

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из них.

Паскаль

```
const N = 30;
var a: array [1..N] of integer;
    i, j, k, max: integer;
begin
   for i := 1 to N do
     readln(a[i]);
       ...
end.
```

Бейсик

```
Module program
Sub Main()
Dim N As Integer = 30
Dim A(N) As Integer
Dim I, J, K, MIN As Integer
For I = 1 To N
A.SetValue(Val(Console.ReadLine()),I)
Next
...
End Sub
End Module
```

В качестве ответа необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия.

С3. У исполнителя Удвоитель-Вычитатель две команды, которым присвоены номера:

1. Прибавь 1,

2. Умножь на 2.

Первая из них увеличивает число на 1, вторая — удваивает его.

Программа для этого исполнителя — это последовательность команд.

Сколько есть программ, которые число 1 преобразуют в число 21?

Ответ обоснуйте.

С4. В связи с совершенствованием страховой медицины при министерстве здравоохранения создан центр статистической обработки данных. В него присылаются сведения из всех поликлиник города о суммах, которые страховой фонд должен перечислить каждой поликлинике за обслуживание каждого пациента.

Вам предлагается написать эффективную, в том числе по используемой памяти и по количеству команд, программу, которая будет статистически обрабатывать пришедшие запросы, чтобы определить поликлиники с минимальным и максимальным отклонением среднего значения страховой суммы на одного человека от среднего значения такой суммы по городу в целом.

Перед текстом программы кратко опишите используемый вами алгоритм решения задачи.

На вход программе подаются типовые записи, каждая из которых содержит разделённые строго одним пробелом данные:

<Фамилия пациента> <Имя пациента> <код обращения> <номер поликлиники> <сумма, руб>,

где <код обращения> — 8-значный код, определяющий диагноз и повторность обращения с ним в поликлинику; <номер поликлиники> — целое число от 1 до 200 (для некоторых поликлиник записи могут отсутствовать, если, например, поликлиника находилась на ремонте). Страховые суммы могут составлять от 100 до 10 000 рублей. Фамилии и имена пациентов всегда записаны русскими буквами.

Количество строк с исходными данными может быть очень велико и заранее не известно. Завершение ввода данных определяется вводом строки «End of text». Длина каждой такой строки не превышает 255 символов (включая разделяющие пробелы).

Пример входных данных:

Иванков Александр 01728435 115 5500

Сидоров Иван 11638436 122 3800

Никитенко Владислав 00127845 3 580

...

End of text

Программа должна вывести:

- среднее значение страховой суммы на одного человека по городу (вычисленное по сведениям, содержащимся в исходных данных) с округлением до целого числа рублей;
- номер поликлиники, в которой наблюдается максимальное отклонение среднего значения страховой суммы на одного человека по данной поликлинике от среднегородского значения, и само это значение с округлением до целого числа рублей;
- номер поликлиники, в которой наблюдается минимальное отклонение среднего значения страховой суммы на одного человека по данной поликлинике от среднегородского значения, и само это значение с округлением до целого числа рублей.

Пример выходных данных:

Средняя страховая сумма по городу: 3878 руб.

Минимальное отклонение — поликлиника № 118: 300 руб.

Максимальное отклонение — поликлиника № 178: 2015 руб.

Ответы и решения

№ задания	Ответ/решение	№ занятия при неправильном решении
A1	8 (вариант ответа №3) ⌛ Не забудьте дописать незначащие нули слева, дополняющие двоичное число до 16 разрядов!	5
A2	14 (вариант ответа №2)	4
A3	$(A \to B) \wedge (B \to C)$ (вариант ответа №3)	7
A4	*at?.??cx (вариант ответа №2)	13
A5	631 (вариант ответа №4)	9
A6	Кузнецов Л.Б. (вариант ответа №1)	18
A7	0,97 (вариант ответа №3)	16
A8	Да; 90 Мб (вариант ответа №2)	15
A9	· — · (вариант ответа №4) ⌛ Проведите аналогию между знаками предлагаемого кода «морзянки» и двоичной системой счисления!	2
A10	АЯКС (вариант ответа №3)	7
A11	4 Кб (вариант ответа №4) ⌛ Количество байтов, отводимых на каждый пароль, должно быть таким, чтобы его хватило на хранение любого пароля — и короткого, и длинного (при этом для короткого пароля часть битов может не использоваться)!	1
A12	0 1 2 3 4 5 6 7 8 9 (вариант ответа №1)	23
A13	2 (вариант ответа №3) ⌛ Хотя способ задания условия движения и остановки РОБОТа в данной задаче изменён, общие принципы решения остаются теми же, что и раньше.	10
A14	2 (вариант ответа №2)	25
B1	960	1
B2	12212	11
B3	1048575	21
B4	129	6
B5	10	17
B6	20	20
B7	5	26

Окончание табл.

№ задания	Ответ/решение	№ занятия при неправильном решении
B8	7 ⌛ Задачу удобнее решать, преобразовав исходное двоичное число в десятичное	5
B9	7	4
B10	983048,25 ⌛ Нужно учесть, что время получения Настей первого мегабайта данных включает в себя и время на отправку запроса на первую порцию данных. Далее, так как скачивание данных Настей происходит быстрее их ретрансляции, а ретрансляция производится без разбиения на порции, это условие разбиения файла на порции при его получении Настей больше ни на что не влияет.	3
B11	118.234.112.0 ⌛ Нужно, чтобы маска выделяла такое пространство адресов внутри сети, чтобы его было минимально достаточно для подключения заданного числа компьютеров (и иных устройств), учитывая, что два адреса внутри сети являются служебными. После того, как маска определена, адрес подсети ищется по ней обычным способом.	14
B12	530	19
B13	25	11
B14	48 ⌛ 1) Последнее уравнение жёстко определяет $x_{10} = 1$. Это влияет на количество вариантов при анализе предпоследнего уравнения. 2) Анализ предпоследнего уравнения показывает: при $(x_7 \equiv x_8) = 0$ допустимы любые комбинации $(x_9 \equiv x_{10})$, т.е. имеем для каждого из двух наборов различных (x_7, x_8) по 4 набора (x_9, x_{10}), из которых в силу последнего уравнения годятся только по 2 набора; при $(x_7 \equiv x_8) = 1$ допустимо только равенство x_9 и x_{10}, т.е. для каждого из двух наборов равных (x_7, x_8) по 2 набора (x_9, x_{10}), из которых в силу последнего уравнения годятся только по 1 набору. 3) Анализ остальных уравнений — аналогично предпоследнему, но без исключения половины наборов, которое для предпоследнего уравнения делалось в силу необходимости $x_{10} = 1$. 4) После анализа первого уравнения надо вернуться к последнему и учесть, что x_1 также должно быть равно 1, что уменьшит общее число вариантов вдвое.	8

C1.

1.

Область	Условие 1 (y >= −x)	Условие 2 ((x−1)²+(y−1)²>=1)	Условие 3 (x²+y²<=1)	Программа выведет	Область обрабатывается верно
A	нет	—	—	—	нет
B	да	да	нет	не принадлежит	да
C	да	нет	—	—	нет
D	нет	—	—	—	нет
E	да	нет	—	—	нет
F	да	да	да	принадлежит	нет
G	да	да	да	принадлежит	да

2. Пример пары чисел (x, y), обрабатываемых ошибочно: (0,5; −0,5).

3. Возможная доработка программы:

Паскаль

```
var x, y: real;
begin
readln(x, y);
if (y >= -x) and (y >= 0) and
   ((x - 1) * (x - 1) + (y - 1) * (y - 1) >= 1)
   and (x * x + y * y <= 1) then
  write('принадлежит')
else
  write('не принадлежит')
end.
```

Бейсик

```
Module Program
Sub Main()
Dim x, y As Double
x = Val(Console.ReadLine())
y = Val(Console.ReadLine())
If y >= -x And y >= 0 And
  (x - 1) * (x - 1) + (y - 1) * (y - 1) >= 1
  And x * x + y * y <= 1 Then
Console.Write("принадлежит")
Else
Console.Write("не принадлежит")
End If
End If
End If
End Sub
End Module
```

№ занятия при неправильном решении	27

C2.

```
const N = 30;
var a: array [1..N] of integer;
    i, j, k, max: integer;
begin
  for i := 1 to N do
    readln(a[i]);
  k := 0;
  // счетчик чисел
  max := -101;
  // предполагаемый максимум меньше самого малого значения элемента
```

```
  for i := 1 to N do
    // условия: четный индекс И нечетный элемент И не кратный 3
    if (i mod 2 = 0) and (a[i] mod 2 <> 0) and (a[i] mod 3 <> 0) then begin
k := k + 1;
      // подсчет количества таких элементов
      if a[i] > max then max := a[i];
     // если текущий такой элемент больше предполагаемого max,
     // то он становится max
    end;
  writeln('Всего таких элементов: ', k, '; из них максимальный - ', max);
end.
```

№ занятия при неправильном решении	24

С3.

Ответ: 60.

Обоснование:

Пусть $R(n)$ — количество программ, которые преобразуют число 1 в число n.

Рассмотрим обратный процесс — получение числа 1 из числа 21 при помощи обратных команд «прибавить 1» и «делить на 2».

Тогда $R(n) = R(n/2) + R(n - 1)$ *{в общем случае очередное число может быть получено или делением на 2, или вычитанием единицы}.*

Для получения числа 21 из числа 1 используются следующие шаги программы:

$R(1) = 1$

$R(2) = R(2/2) + R(2 - 1) = R(1) + R(1) = 1 + 1 = 2.$

$R(3) = R(3/2) + R(3 - 1) = R(2) = 2.$

Слагаемые, дающие нецелый результат, исключаются; значения других слагаемых берутся из ранее вычисленных (если же таких не было, то тоже исключаются).

$R(4) = R(4/2) + R(4 - 1) = R(2) + R(3) = 2 + 2 = 4.$

$R(5) = R(5/2) + R(5 - 1) = R(4) = 4.$

$R(6) = R(6/2) + R(6 - 1) = R(3) + R(5) = 2 + 4 = 6.$

$R(7) = R(7/2) + R(7 - 1) = R(6) = 6.$

$R(8) = R(8/2) + R(8 - 1) = R(4) + R(7) = 4 + 6 = 10.$

$R(9) = R(9/2) + R(9 - 1) = R(8) = 10.$

$R(10) = R(10/2) + R(10 - 1) = R(5) + R(9) = 4 + 10 = 14.$

$R(11) = R(11/2) + R(11 - 1) = R(10) = 14.$

$R(12) = R(12/2) + R(12 - 1) = R(6) + R(11) = 6 + 14 = 20.$

$R(13) = R(13/2) + R(13 - 1) = R(12) = 20.$

$R(14) = R(14/2) + R(14 - 1) = R(7) + R(13) = 6 + 20 = 26.$

$R(15) = R(15/2) + R(15 - 1) = R(14) = 26.$

$R(16) = R(16/2) + R(16 - 1) = R(8) + R(15) = 10 + 26 = 36.$

$R(17) = R(17/2) + R(17 - 1) = R(16) = 36.$

$R(18) = R(18/2) + R(18 - 1) = R(9) + R(17) = 10 + 36 = 46.$

$R(19) = R(19/2) + R(19 - 1) = R(18) = 46.$

$R(20) = R(20/2) + R(20 - 1) = R(10) + R(19) = 14 + 46 = 60.$

$R(21) = R(21/2) + R(21 - 1) = R(20) = 60.$

№ занятия при неправильном решении	24

C4.

1. Нужен подсчёт:

- среднего значения для каждой поликлиники — требуется два массива-счётчика для подсчёта общей суммы страховых выплат и количества пациентов соответственно; индексы обоих массивов-счётчиков меняются от 1 до 200; тип массива для сумм — real (в нём позже будет вычисляться среднее), для количества пациентов — integer;
- среднего значения по всем записям — требуются две переменные-счётчика: общая сумма (тип real — в ней позже будет вычисляться среднее) и общее количество пациентов (тип integer).

```
var sum: array [1..200] of real;
    num: array [1..200] of integer;
    allsum: real;
    allnum: integer;
```

Отдельным циклом обнуляются оба массива-счётчика. Обнуляются переменные-счётчики.

2. Ввод данных.

Количество входных строк заранее не известно; признак окончания ввода — входная строка «End of text». Слова в ней разделены пробелами. Следовательно, при чтении они будут восприняты как три отдельных данных. Обычно первое данное — фамилия, записанная русскими буквами. В финальной строке в качестве «фамилии» будет считано «End». Это можно использовать как однозначный признак конца ввода в условии для цикла ПОКА.

```
// посимвольное чтение фамилии включая завершающий пробел
fam := '';
repeat
  read(c);
  fam := fam + c;
until c = ' ';
// цикл ПОКА работает, если fam не равно слову 'End '
// (с пробелом в конце)
while (fam <> 'End ') do begin
  <обработка данных>
  // чтение фамилии для следующего прохода цикла
  fam := '';
  repeat
    read(c);
    fam := fam + c;
  until c = ' ';
end;
```

3. Обработка входных строк.

Фамилия пациента уже считана; используется только для проверки окончания ввода строк.

Имя пациента и код обращения пропускаются путём посимвольного считывания.

Номер поликлиники и страховая сумма считываются в целые переменные (nump и sump).

Значение sump прибавляется к значению элемента массива-счётчика sum, индекс которого равен nump.

Значение элемента массива-счётчика num, индекс которого равен nump, увеличивается на 1.

Значение sump прибавляется к значению переменной-счётчика allsum.

Значение переменной — счётчика allnum увеличивается на 1.

4. Обработка массивов-счётчиков по завершении чтения входных данных.

1) В переменной allsum вычисляется среднее значение по городу делением allsum на allnum.

2) В качестве предполагаемого минимума (min) берётся заведомо большее значение модуля разности среднего по городу и среднего по поликлинике: 10001. В качестве предполагаемого максимума (max) берётся заведомо меньшее значение –1.

3) Строится цикл просмотра массивов для всех номеров поликлиник (*i* от 1 до 200):

- если num[i] не равно нулю (т.е. записи для этой поликлиники были в обработке), то:
- вычисляется среднее для данной поликлиники: sum[i] := sum[i] / num[i];
- сравнивается модуль разности среднего по городу (allsum) и среднего по поликлинике (sum[i]): если он больше max, то приравнивается ему max и запоминается *i* в переменной nummax; иначе если он меньше min, то приравнивается ему min и запоминается *i* в переменной nummin.

По завершении работы цикла в переменных max, nummax, min и nummin содержатся искомые результаты.

5. Вывод данных: производится в соответствии с требуемым форматом (включая текстовые комментарии).

Полный текст программы:

```
Program C4;
var sum: array [1..200] of real;
    num: array [1..200] of integer;
    allsum: real;
    allnum: integer;
    max, min: real;
    nummax, nummin: integer;
    fam: string[40];
    c: char;
    nump, sump: integer;
    i: integer;
begin
  // обнуления счетчиков
  for i := 1 to 200 do begin
    sum[i] := 0;
    num[i] := 0;
  end;
  allsum := 0;
  allnum := 0;
  // посимвольное чтение фамилии включая завершающий пробел
  fam := '';
  repeat
    read(c);
    fam := fam + c;
  until c = ' ';
  // цикл ПОКА работает, если fam не равно слову 'End'
  // (с пробелом в конце)
  while (fam <> 'End ') do begin
    // пропуск имени (включая пробел)
    repeat
      read(c);
    until c = ' ';
    // пропуск кода обращения (включая пробел)
```

```
      repeat
        read(c);
      until c = ' ';
      // чтение номера поликлиники и суммы
      readln(nump, sump);
      // обработка введенных данных
      allsum := allsum + sump;
      allnum := allnum + 1;
      sum[nump] := sum[nump] + sump;
      num[nump] := num[nump] + 1;
      // чтение фамилии для следующего прохода цикла
      fam := '';
      repeat
        read(c);
        fam := fam + c;
      until c = ' ';
    end;
    // обработка счетчиков
    allsum := allsum / allnum;
    min := 10001;
    max := -1;
    for i := 1 to 200 do
      if num[i] > 0 then begin
        sum[i] := sum[i] / num[i];
        if abs(allsum - sum[i]) > max then begin
         max := abs(allsum - sum[i]);
         nummax := i;
        end
        else if abs(allsum - sum[i]) < min then begin
         min := abs(allsum - sum[i]);
         nummin := i;
        end;
      end;
    // вывод результатов
     writeln('Средняя страховая сумма по городу: ', round(allsum), '
руб.');
    writeln('Минимальное отклонение — поликлиника № ',nummin,':
    ',round(min),' руб.');
    writeln('Максимальное отклонение — поликлиника № ',nummax,':
    ',round(max),' руб.');
  end.
```

№ занятия при неправильном решении	28, 29, 30

Тесты

Богомолова Ольга Борисовна

ЕГЭ
за 30 дней

ИНФОРМАТИКА

ЭКСПРЕСС-РЕПЕТИТОР

Редакция «Образовательные проекты»

Ответственный редактор *Н.А. Шармай*
Художественный редактор *Т.Н. Войткевич*
Технический редактор *А.Л. Шелудченко*
Корректор *И.Н. Мокина*
Оригинал-макет подготовлен ООО «БЕТА-Фрейм»

Подписано в печать 24.12.2013. Формат 84×108 $^{1}/_{16}$.
Бумага газетная. Печать офсетная.
Усл. печ. л. 47,0. Тираж 3000 экз. Заказ 230.

Общероссийский классификатор продукции ОК-005-93, том 2;
953005 — литература учебная

Сертификат соответствия № РОСС.RU.АЕ51.Н16526 от 26.09.2013

ООО «Издательство АСТ».
129085, г. Москва, Звёздный бульвар, д. 21, стр. 3, комн. 5.

ООО «Издательство Астрель».
129085, г. Москва, пр-д Ольминского, д. 3а.

Издано при участии ООО «Харвест». Свидетельство о ГРИИРПИ № 1/17 от 16.08.2013.
Ул. Кульман, д. 1, корп. 3, эт. 4, к. 42, 220013, г. Минск, Республика Беларусь.
E-mail редакции: harvest@anitex.by

Республиканское унитарное предприятие
«Издательство «Белорусский Дом печати».
ЛП № 02330/0494179 от 03.04.2009.
Пр. Независимости, 79, 220013, г. Минск, Республика Беларусь.

По вопросам приобретения книг обращаться по адресу:
123317, г. Москва, Пресненская наб., д. 6, стр. 2, БЦ «Империя», а/я № 5.
Отдел реализации учебной литературы издательств «АСТ» и «Астрель».
Справки по телефону 8 (499) 951-60-00, доб. 107, 565, 566, 578.