ВМК МГУ – ШКОЛЕ

ПОСОБИЕ
для подготовки к ЕГЭ

ИНФОРМАТИКА

Издательство БИНОМ

ВМК МГУ – ШКОЛЕ

ИНФОРМАТИКА

ПОСОБИЕ
для подготовки к ЕГЭ

2-е издание,
исправленное и дополненное

Под редакцией
Е. Т. Вовк

Москва
БИНОМ. Лаборатория знаний

УДК 004.9
ББК 32.97
И74

Авторы:
Е. Т. Вовк, Н. В. Глинка, Т. Ю. Грацианова, Е. И. Гуревич,
О. Р. Лапонина, Н. Б. Линев, К. Б. Мурашкина, Е. В. Рыбко,
К. С. Филиппов, Е. Ю. Фоменко, А. Л. Яковлев

Информатика: пособие для подготовки к ЕГЭ / Е. Т. Вовк
И74 [и др.] ; под ред. Е. Т. Вовк. — М. : БИНОМ. Лаборатория знаний, 2013. — 322 с. : ил. — (ВМК МГУ — школе).

ISBN 978-5-9963-1224-5

Данная книга рекомендуется в качестве пособия при подготовке к ЕГЭ по информатике.

Разделы книги соответствуют темам, включенным в ЕГЭ. В начале каждого раздела приведена краткая теоретическая информация по теме, содержащая основные определения и описание методов решения задач. Основу разделов составляют задачи для самостоятельного решения. В конце книги приводятся ответы, а для наиболее сложных задач дается разбор решения или рекомендации по решению.

Пособие разработано коллективом преподавателей факультета вычислительной математики и кибернетики (ВМК) МГУ имени М. В. Ломоносова — ведущим учебным заведением страны в области информационных технологий.

УДК 004.9
ББК 32.97

Учебное издание

Серия: «ВМК МГУ — школе»

Вовк Елена Тимофеевна
Глинка Надежда Владимировна
Грацианова Татьяна Юрьевна и др.

ИНФОРМАТИКА: ПОСОБИЕ ДЛЯ ПОДГОТОВКИ К ЕГЭ

Редактор *Д. Ю. Усенков*
Технический редактор *Е. В. Денюкова*. Корректор *Е. Н. Клитина*

Подписано в печать 16.10.12. Формат 70 × 100/16.
Усл. печ. л. 26,65. Тираж 2000 экз. Заказ 1648.

Издательство «БИНОМ. Лаборатория знаний»
125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272, e-mail: binom@Lbz.ru, http://www.Lbz.ru

Отпечатано в ООО ПФ «Полиграфист»,
160001, г. Вологда, ул. Челюскинцев, 3.
Тел.: 8(817-2) 72-61-75; 8(817-2) 72-60-63.

ISBN 978-5-9963-1224-5

Оглавление

Глава 1

Информация и ее кодирование

Системы счисления

Системой счисления называется совокупность правил именования и изображения чисел с помощью конечного набора символов, называемых цифрами.

Системы счисления бывают позиционные и непозиционные. Примером непозиционной системы счисления является римская система, в которой существует следующий базовый набор чисел:

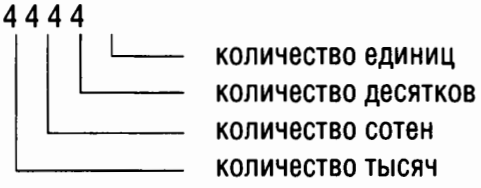
I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Все остальные числа получаются в результате сложения или вычитания чисел базового набора по следующему правилу: если меньшая цифра стоит перед большей (слева от большей), то ее значение вычитается. Например, число MCMXCVII можно представить как

$$1000 - 100 + 1000 - 10 + 100 + 5 + 1 + 1 = 1997.$$

Классическая римская система позволяет составлять числа в диапазоне от 1 до 3999.

Система счисления называется *позиционной*, если значение цифры в записи числа зависит от позиции, которую она занимает в последовательности цифр, изображающей число. Например:



Основание системы счисления – количество цифр, используемых для записи числа. В таблице даны примеры нескольких систем счисления с указанием их основания и алфавита (набора цифр).

Название системы	Основание	Используемые цифры
Десятичная	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Двоичная	2	0, 1
Восьмеричная	8	0, 1, 2, 3, 4, 5, 6, 7
Шестнадцатеричная	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

В следующей таблице приведены первые 17 чисел, записанных в различных системах счисления:

Основание										
10	0	1	2	3	4	5	6	7	8	9
2	0	1	**10**	11	100	101	110	111	1000	1001
8	0	1	2	3	4	5	6	7	**10**	11
16	0	1	2	3	4	5	6	7	8	9

Основание							
10	**10**	11	12	13	14	15	16
2	1010	1011	1100	1101	1110	1111	10000
8	12	13	14	15	16	17	20
16	A	B	C	D	E	F	**10**

Обратите внимание: при последовательном счете, начиная с нуля, в любой системе счисления обязательно наступает момент, когда число обозначается как «10». Появление двух знаков в изображении числа означает, что знаки алфавита данной системы счисления закончились и приходится использовать комбинацию из двух цифр.

В общем случае имеет место равенство:

$$q = 10_q,$$

где q – основание позиционной системы счисления, а 10_q (читается как «один, ноль») – способ обозначения, что число записано в q-ичной системе счисления. Пример: $2 = 10_2$, $8 = 10_8$, $16 = 10_{16}$.

Перевод в десятичную систему

Любое число в десятичной системе счисления можно разложить по степеням числа «10», т.е. представить в виде:
$4444 = 4 \times 1000 + 4 \times 100 + 4 \times 10 + 4 \times 1 = 4 \times 10^3 + 4 \times 10^2 + 4 \times 10^1 + 4 \times 10^0$.

Число с дробной частью записывается по тем же правилам:
$33{,}5 = 3 \times 10^1 + 3 \times 10^0 + 5 \times 10^{-1}$.

Аналогичное утверждение имеет место для чисел в любой позиционной системе счисления.

Пример 1:
$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = 13_{10}$.

$\times 2^3$	$\times 2^2$	$\times 2^1$	$\times 2^0$
1	1	0	1

Пример 2:
$452{,}14_8 = 4 \times 8^2 + 5 \times 8^1 + 2 \times 8^0 + 1 \times 8^{-1} + 4 \times 8^{-2} = 298{,}1875_{10}$.

$\times 8^2$	$\times 8^1$	$\times 8^0$	$\times 8^{-1}$	$\times 8^{-2}$
4	5	2,	1	4

Пример 3:

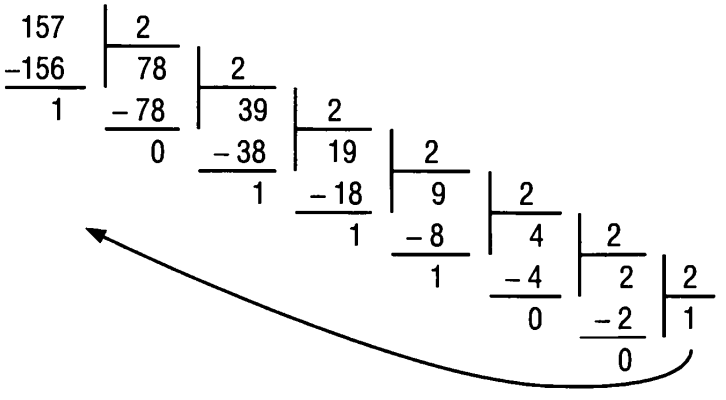
$1001101{,}11_2 = 1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 101{,}75_{10}$.

$\times 2^6$	$\times 2^5$	$\times 2^4$	$\times 2^3$	$\times 2^2$	$\times 2^1$	$\times 2^0$	$\times 2^{-1}$	$\times 2^{-2}$
1	0	0	1	1	0	1,	1	1

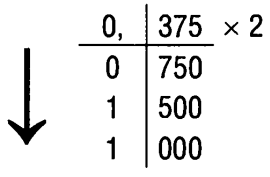
Перевод из десятичной системы в другие системы счисления

Принципы перехода от десятичной системы счисления к другим позиционным системам рассмотрим на примере перевода в двоичную систему.

Для перевода чисел из десятичной системы в двоичную применяют метод последовательного деления целой части на 2, как показано на рисунке ниже. Пусть, например, требуется перевести число 157 из десятичной системы в двоичную. Деление продолжается до тех пор, пока частное не окажется равным числу, меньшему делителя. Результат записывается как обычно, слева направо, по правилу: начинаем с последнего частного, а за ним записываем каждый остаток по порядку, указанному стрелкой. В нашем случае получится число 10011101_2.



Дробную часть числа, если таковая имеется, переводят по другому правилу. Пусть требуется перевести число 0,375 из десятичной системы в двоичную. Для этого дробная часть числа последовательно умножается на 2.



Справа от вертикальной черты записываются цифры дробной части, получаемые в процессе умножения. В нашем примере мы умножаем число 375 на 2 (в десятичной системе). Получим 750. Слева от черты ставим «0». Далее 750

умножаем на 2. Получаем 1500. При этом справа от вертикальной черты должно находиться ровно столько цифр, сколько их было в дробной части исходного числа. В нашем случае 3 цифры. Цифра «1» попадает в разряд единиц, поэтому окажется слева от черты.

Обратите внимание на то, что умножение проводится только с числом, стоящим *справа* от вертикальной черты. Таким образом, следующим действием будет $500 \times 2 = 1000$. При записи результата умножения единица окажется слева от черты, а справа будут нули «000». Умножение закончено. Теперь осталось записать ответ. В дробной части двоичного числа будут находиться цифры, оказавшиеся *слева* от черты в порядке, указанном стрелкой, т.е. $0{,}011_2$.

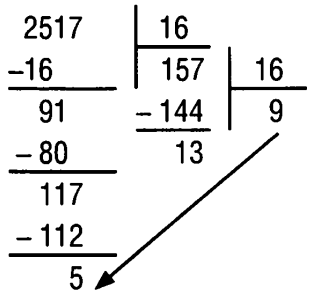
Бывают случаи, когда в результате умножения не получается конечной дроби. Тогда умножение проводят столько раз, сколько это требуется по условию задачи, например дробную часть вычисляют до пятого знака.

Для перевода десятичного числа в другие позиционные системы правила аналогичны: *целую часть* нужно последовательно *делить* на основание системы счисления, в которую переводится число, а *дробную часть – умножать* на это основание.

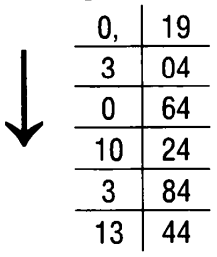
1 Перевести число 2517,19 из десятичной системы в шестнадцатеричную. Дробную часть вычислять до пятого знака.

Решение.

1. Переводим целую часть методом деления. Последнее частное равно 9. Остатки – 13 и 5. Записываем результат, помня о том, что число 13 в шестнадцатеричной системе записывается как «D». Получаем $9D5_{16}$.



2. Переводим дробную часть.



Записываем результат, помня о том, что число 13 в шестнадцатеричной системе записывается как «D», а число 10 как «A». Получаем $0{,}30A3D_{16}$.

Ответ: $2517{,}19_{10} = 9D5{,}30A3D_{16}$.

Прямой перевод между 16-, 8-, 4- и 2-й системами счисления

Существует взаимно однозначное соответствие между цифрами, используемыми в четверичной, восьмеричной и шестнадцатеричной системах счисления и числами двоичной системы. Это соответствие можно представить в виде таблицы:

X_{10}	X_{16}	$X_{2\text{-}16}$	$X_{2\text{-}8}$	$X_{2\text{-}4}$
0	0	0000	000	00
1	1	0001	001	01
2	2	0010	010	10
3	3	0011	011	11
4	4	0100	100	
5	5	0101	101	
6	6	0110	110	
7	7	0111	111	
8	8	1000		
9	9	1001		
10	A	1010		
11	B	1011		
12	C	1100		
13	D	1101		
14	E	1110		
15	F	1111		

В последней колонке представлено соответствие между *цифрами* четверичной системы и двоичными числами. Рассуждения таковы: в четверичной системе счисления используются 4 цифры: 0, 1, 2 и 3. Чтобы закодировать каждую цифру, в этой системе требуется 2 бита информации. При этом двоичные числа не только поставлены в соответствие четверичным *цифрам*, но и равны им по величине:

$$0_2 = 0_4;\ 1_2 = 1_4;\ 10_2 = 2_4;\ 11_2 = 3_4.$$

Получается, что каждый разряд четверичного числа может быть представлен в виде двухразрядного двоичного числа.

Аналогичны рассуждения и для восьмеричных и шестнадцатеричных цифр. Такое взаимно однозначное соответствие позволяет легко переводить числа из двоичной системы в четверичную, восьмеричную и шестнадцатеричную и наоборот.

2 Перевести число $5A2{,}4E_{16}$ в двоичную систему.

Решение. Существует взаимно однозначное соответствие между шестнадцатеричными цифрами и числами двоичной системы. Каж-

дая шестнадцатеричная цифра может быть представлена четырехразрядным двоичным числом, равным по величине этой цифре. Цифра 5_{16} представляется как 0101_2 (в двоичном числе должно быть ровно 4 разряда, поэтому хотя $5_{16} = 101_2$, надо добавить к двоичному числу незначащий нуль, не влияющий на величину числа). Распишем по этому правилу каждый разряд исходного шестнадцатеричного числа:

5	A	2	4	E
0101	1010	0010	0100	1110

В результате мы получили:
$5A2,4E_{16} = 010110100010,01001110_2$.

Незначащие нули слева и справа можно отбросить.

Ответ: $5A2,4E_{16} = 10110100010,0100111_2$.

3 Перевести число $10010111010,1001101_2$ в восьмеричную систему.

Решение. Существует взаимно однозначное соответствие между восьмеричными цифрами и числами двоичной системы. Каждая восьмеричная цифра может быть представлена трехразрядным двоичным числом, равным по величине этой цифре, т.е. надо разбить исходное двоичное число на группы цифр по три в каждой. Здесь важно помнить, что разбиение должно проводиться от запятой в обе стороны, а если число целое – то справа. В данном случае разбиение будет таким:

$10\ 010\ 111\ 010,\ 100\ 110\ 1_2$.

Поскольку в группах слева и справа цифр не хватает до трех, надо добавить необходимое количество незначащих нулей слева и справа, что не изменит величины исходного двоичного числа. В итоге получим: $010\ 010\ 111\ 010,\ 100\ 110\ 100_2$. Теперь каждую тройку цифр надо представить соответствующей цифрой восьмеричной системы:

010	010	111	010,	100	110	100
2	2	7	2,	4	6	4

Ответ: $10010111010,1001101_2 = 2272,464_8$.

Задачи к разделу «Системы счисления»

1 В системе счисления с некоторым основанием десятичное число 43 записывается в виде «111». Укажите это основание.

2 В системе счисления с некоторым основанием десятичное число 85 записывается в виде «151». Укажите это основание.

3 В системе счисления с некоторым основанием десятичное число 34 записывается в виде «202». Укажите это основание.

4 В системе счисления с некоторым основанием десятичное число 148 записывается в виде «125». Укажите это основание.

5 В системе счисления с некоторым основанием десятичное число 202 записывается в виде «244». Укажите это основание.

6 Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись десятичного числа 53 оканчивается на 3.

7 Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись десятичного числа 26 оканчивается на 2.

8 Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись десятичного числа 33 оканчивается на 1.

9 Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись десятичного числа 40 оканчивается на 4.

10 Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись десятичного числа 39 оканчивается на 7.

11 Укажите через запятую в порядке возрастания все числа, не превосходящие 30, запись которых в двоичной системе счисления оканчивается на «101». Числа в ответе указывайте в десятичной системе счисления.

12 Укажите через запятую в порядке возрастания все числа, не превосходящие 17, запись которых в двоичной системе счисления оканчивается на «11». Числа в ответе указывайте в десятичной системе счисления.

13 Укажите через запятую в порядке возрастания все числа, не превосходящие 33, запись которых в двоичной системе счисления оканчивается на «100». Числа в ответе указывайте в десятичной системе счисления.

14 Укажите через запятую в порядке возрастания все числа, не превосходящие 15, запись которых в двоичной системе счисления оканчивается на «10». Числа в ответе указывайте в десятичной системе счисления.

15 Укажите через запятую в порядке возрастания все числа, не превосходящие 35, запись которых в двоичной системе счисления оканчивается на «110». Числа в ответе указывайте в десятичной системе счисления.

16 Количество единиц в двоичной записи числа 12,25 равно...

17 Количество единиц в двоичной записи числа 22,5 равно...

18 Количество единиц в двоичной записи числа 35,625 равно...

19 Количество значащих нулей в двоичной записи числа 29,25 равно...

20 Количество значащих нулей в двоичной записи числа 18,125 равно...

21 Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа (от 00 до 11 соответственно). Если таким образом закодировать последовательность символов ГБАВ и перевести результат в шестнадцатеричную систему счисления, то получится...

22 Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа (от 00 до 11 соответственно). Если таким образом закодировать последовательность символов ВАБГ и перевести результат в восьмеричную систему счисления, то получится...

23 Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа (от 00 до 11 соответственно). Если таким образом закодировать последовательность символов ВГБА и перевести результат в четверичную систему счисления, то получится...

24 Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа (от 00 до 11 соответственно). Если таким образом закодировать последовательность символов ГАВБ и перевести результат в шестнадцатеричную систему счисления, то получится...

25 Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа (от 00 до 11 соответственно). Если таким образом закодировать последовательность символов ГВБА и перевести результат в шестнадцатеричную систему счисления, то получится...

26 Как представляется десятичное число 13,5 в двоичной системе счисления?

27 Как представляется десятичное число 21,375 в двоичной системе счисления?

28 Как представляется десятичное число 41,25 в двоичной системе счисления?

29 Как представляется десятичное число 38,75 в двоичной системе счисления?

30 Как представляется десятичное число 27,625 в двоичной системе счисления?

31 Как представляется десятичное число 416 в восьмеричной системе счисления?

32 Как представляется десятичное число 510 в восьмеричной системе счисления?

33 Как представляется десятичное число 235 в восьмеричной системе счисления?

34 Как представляется десятичное число 333 в восьмеричной системе счисления?

35 Как представляется десятичное число 247 в восьмеричной системе счисления?

36 Как представляется десятичное число 523 в шестнадцатеричной системе счисления?

37 Как представляется десятичное число 680 в шестнадцатеричной системе счисления?

38 Как представляется десятичное число 495 в шестнадцатеричной системе счисления?

39 Как представляется десятичное число 637 в шестнадцатеричной системе счисления?

40 Как представляется десятичное число 892 в шестнадцатеричной системе счисления?

41 Вычислите значение суммы: $101_2 + 11_8 + 10_{16}$. Результат представьте в виде десятичного числа.

42 Вычислите значение суммы: $12_8 + 12_{10} + 12_{16}$. Результат представьте в виде двоичного числа.

43 Вычислите значение суммы: $1011_2 + 15_8 + 1E_{16}$. Результат представьте в виде четверичного числа.

44 Вычислите значение суммы: $110110_2 + 33_4 + 33_{10}$. Результат представьте в виде восьмеричного числа.

45 Вычислите значение суммы: $221_4 + 55_8 + 55_{10}$. Результат представьте в виде шестнадцатеричного числа.

46 Вычислите значение разности двух чисел: $11_{16} - 11_8$. Результат представьте в виде десятичного числа.

47 Вычислите значение разности двух чисел: $31_{16} - 31_{10}$. Результат представьте в виде двоичного числа.

48 Вычислите значение разности двух чисел: $573_{10} - 573_8$. Результат представьте в виде шестнадцатеричного числа.

49 Вычислите значение разности двух чисел: $101_8 - 10011_2$. Результат представьте в виде четверичного числа.

50 Вычислите значение разности двух чисел: $A7_{16} - 11010_2$. Результат представьте в виде восьмеричного числа.

Измерение информации

Вероятностный подход

Пусть должно произойти какое-то событие. Обозначим количество равновероятно возможных результатов этого события через N.

Например, бросаем монету. Выпадает «орел» или «решка» – это равновероятные события, значит, $N = 2$.

Количество информации i, содержащееся в сообщении о том, что произошло одно из N равновероятных событий, определяется решением уравнения $2^i = N$. Отсюда

$$i = \log_2 N.$$

Однако значение $\log_2 N$ не всегда оказывается целым числом, тогда как ответом в задачах, по смыслу их постановки, может быть только целое число.

Следовательно, в этих случаях в качестве ответа надо взять **ближайшее целое число, большее** $\log_2 N$.

Для решения подобного уравнения удобно воспользоваться таблицей степеней числа 2:

n	2^n	n	2^n	n	2^n
0	1	6	64	12	4096
1	2	7	128	13	8192
2	4	8	256	14	16384
3	8	9	512	15	32768
4	16	10	1024	16	65536
5	32	11	2048	17	131072

За единицу измерения количества информации принят *1 бит*.

Если в формулировке задачи присутствует термин *«бит»*, это автоматически подразумевает двоичное кодирование. Ведь этот термин справедлив только для двоичных алфавитов и означает *«binary digit» – «двоичный знак»*. Такие задачи решаются с применением рассмотренных выше формул: $2^i = N$ и $i = \log_2 N$.

В случае с монетой $2^i = 2$. Отсюда $i = \log_2 2$, откуда $i = 1$, т.е. количество информации, полученной после броска монеты, равно 1 биту.

1 В коробке 16 кубиков различных цветов. Сколько битов информации несет сообщение о том, что из коробки достали зеленый кубик?

Решение. Результат вытаскивания из коробки любого из 16 кубиков – событие равновероятное. Поэтому используем формулу $2^i = N$, где $N = 16$. Следовательно, $i = 4$.

Ответ: 4 бита.

2 При угадывании целого числа в диапазоне от *1* до N получено 7 битов информации. Чему равно N?

Решение. Используем формулу $2^i = N$, где $i = 7$. Следовательно, $N = 128$.

Ответ: $N = 128$.

3 Сколько информации содержит сообщение о том, что некий человек сидит в 12-м ряду на 19-м месте, если в кинотеатре 16 рядов по 32 места в каждом?

Решение. Всего в кинотеатре $16 \times 32 = 512$ мест, т.е. $N = 512$. Используем формулу $2^i = N$. Следовательно, $i = \log_2 512 = 9$.

Ответ: 9 битов.

Вопрос измерения количества информации тесно связан со способом ее кодирования. Например, предыдущая задача может быть сформулирована иначе: какое минимальное количество битов потребуется для кодирования номера каждого зрительского места, если в кинотеатре 16 рядов по 32 места в каждом?

4 Кодовый замок сейфа должен допускать не менее 15 000 уникальных комбинаций. Код устанавливается с помощью двухпозиционных переключателей. Сколько таких переключателей необходимо использовать в конструкции замка?

Решение. Воспользуемся формулой $i = \log_2 N$. $\log_2 15000 \approx 13{,}872$. Получается, что 13 переключателей обеспечат 8192 различных комбинаций, а 14 переключателей – 16384 комбинаций, что удовлетворяет условию задачи.

Ответ: Не менее 14 двухпозиционных переключателей.

Теперь представим себе, что в конструкции кодового замка применяются не двух-, а **трехпозиционные** переключатели. Сколько переключателей потребуется а этом случае?

Если переключатель трехпозиционный, то он может находиться в одном из *трех* различных состояний (например, в верхнем, среднем или нижнем). В этом случае для решения мы будем использовать более общую формулу:

$$x = \log_a N,$$

где N – количество уникальных комбинаций,
x – требуемое количество переключателей,
a – число состояний, в которых может находиться каждый переключатель.

Тогда, подставляя значения из предыдущей задачи, получим:

$$x = \log_3 15000 \approx 8{,}753.$$

Округлив до ближайшего большего целого, получаем **ответ**: 9 переключателей.

5 Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из трех состояний (включено, выключено или мигает). Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 50 различных сигналов?

Решение. Воспользуемся формулой $x = \log_a N$. $\log_3 50 \approx 3{,}56$. Округлив до ближайшего большего целого, получаем 4.

Ответ: Минимум 4 лампочки.

Алфавитный подход

Сообщение можно рассматривать как дискретную последовательность знаков, принадлежащих некоторому алфавиту. Количество символов в алфавите называется *мощностью алфавита*. Количество битов, необходимое для кодирования одного символа алфавита, называется *информационным весом символа алфавита*. Чтобы узнать, сколько битов информации содержится в некотором сообщении, надо количество символов в сообщении умножить на информационный вес символа.

Если N – мощность алфавита, то информационный вес одного символа рассчитывается по формуле: $i = \log_2 N$. Результат округляется до *ближайшего большего целого* значения.

Если k – количество символов в сообщении, то информационный объем всего сообщения рассчитывается по формуле:

$$V = k \times \log_2 N.$$

6 Алфавит некоторого языка состоит из 20 символов. Какое количество информации в битах будет содержать сообщение длиной в 80 символов?

Решение. Для определения информационного веса одного символа алфавита в битах воспользуемся формулой $i = \log_2 N$. $\log_2 20 \approx 4{,}321$. Округлив до ближайшего большего целого, получаем 5. Умножаем этот результат на 80 и получаем 400 битов.

Ответ: 400 битов.

7 Электронный термометр холодильника способен измерять температуру в диапазоне от минус 20°C до плюс 12°C с точностью до 0,5°C. Какое минимальное количество битов потребуется для кодирования каждой величины температуры?

Решение. Нужно определить, сколько различных величин температуры нужно регистрировать. Точность 0,5°C указывает, что на 1°C приходится два измерения (1/0,5 = 2). На отрицательный диапазон приходится $20 \times 2 = 40$ измерений, а на положительный $12 \times 2 = 24$ измерения.

Учитывая температуру 0°, получаем 20 + 24 + 1 = 65 измерений. Применяем формулу $i = \log_2 N$. $\log_2 65 \approx 6{,}022$. Округляем до ближайшего большего целого и получаем 7.

Ответ: 7 битов.

Единицы измерения количества информации

1 байт = 8 бит;
1 килобайт (Кбайт) = 1024 байт = 2^{10} байт;
1 Мегабайт (Мбайт) = 1024 Кбайт = 2^{20} байт;
1 Гигабайт (Гбайт) = 1024 Мбайт = 2^{30} байт;
1 Терабайт (Тбайт) = 1024 Гбайт = 2^{40} байт.

8 Сколько мегабайт информации содержит сообщение объемом 2^{27} битов?

Решение. Учитывая, что 1 Мбайт = 1024 Кбайт = 2^{20} байт, а 1 байт = 8 бит = 2^3 бит, раскладываем исходную величину: 2^{27} бит = $2^{20} \times 2^4 \times 2^3$ бит = 2^4 Мбайт = 16 Мбайт.

Ответ: 16 Мбайт.

9 Сколько битов информации содержит сообщение объемом 4 Мбайта?

Решение. 4 Мбайта = 4×2^{20} байт = $2^2 \times 2^{20} \times 2^3$ бит = 2^{25} битов.

Ответ: 2^{25} битов.

10 Сколько дискет емкостью 1,44 Мбайт потребуется для хранения 2^{28} битов информации?

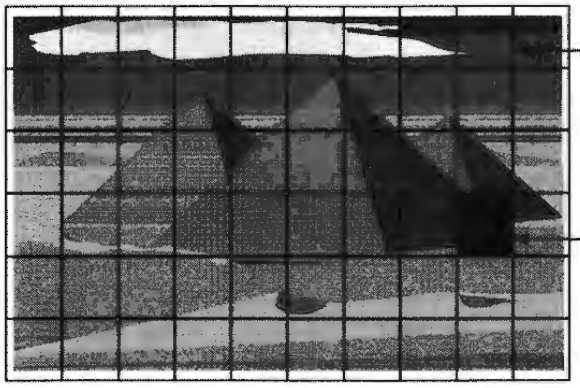
Решение. 2^{28} бит = $2^{20} \times 2^5 \times 2^3$ бит = $2^{20} \times 2^5$ байт = 2^5 Мбайт = 32 Мбайт. 32 / 1,44 = 22,222... Искомое количество дискет может быть только целым. Получается, что потребуется 23 дискеты.

Ответ: 23 дискеты.

Кодирование информации

Кодирование изображений

Представьте себе, что прямоугольную фотографию разлиновали горизонтальными и вертикальными прямыми линиями, другими словами – наложили на нее мелкую сетку (растр). В результате картинка разбилась на ячейки. Каждая ячейка окрашена в один цвет и называется точкой (или пикселем). Цвет можно закодировать, т.е. поставить ему в соответствие уникальное целое число. Тогда изображение превращается в набор целых чисел. Закодированное таким образом изображение называется *растровым*.



Введем обозначения:

K – количество разных цветов, используемых для кодирования изображения;

n – количество битов, необходимое для кодирования цвета одной точки изображения.

Между K и n существует связь: $K = 2^n$.

Пользуясь этой формулой, можно рассчитать, сколько битов необходимо для кодирования цвета одной точки, если известно количество разных цветов, с помощью которых может быть представлено изображение:

$$n = \log_2 K.$$

Количество битов, необходимое для хранения одной точки изображения, называется *глубиной цвета*.

Примеры типов изображений и их кодирование:

Тип изображения	**Количество цветов**	**Кодировка**
Черно-белое изображение	$K = 2$: черный, белый	$2^n = 2$, т. е. n=1: для кодирования цвета достаточно одного бита. Например, 0 – черный; 1 – белый
Изображение из трех цветов	$K = 3$, например, черный, белый, красный.	$2^n = 3$. Одного бита недостаточно для кодирования трех цветов, поэтому берется ближайшее целое с избытком – 2. Например: 00, 01,11
Изображение из четырех цветов	$K = 4$, например, черный, темно-серый, светло-серый, белый	$2^n = 4$, т. е. $n = 2$. Например: 00, 01, 10, 11

Рассмотрим, как представляется цвет на экране монитора. Окраска одного пикселя экрана формируется с помощью трех базовых цветов: красного, зеленого, синего. Эти три цвета являются основой цветовой модели RGB (RGB – Red, Green, Blue). С их помощью можно получить $8 = 2^3$ разных цветов. В данном случае для кодирования каждого из трех базовых цветов достаточно 1 бита: либо этот цвет присутствует в окраске одного пикселя, либо нет. Двоичный код восьмицветной палитры представлен в таблице:

Красный	Зеленый	Синий	Получаемый цвет
0	0	0	Черный
0	0	1	Синий
0	1	0	Зеленый
0	1	1	Голубой
1	0	0	Красный
1	0	1	Лиловый
1	1	0	Желтый
1	1	1	Белый

Однако каждый цвет базовой модели характеризуется не только его наличием, но и яркостью (интенсивностью).

Яркость каждого цвета кодируется восьмиразрядным двоичным числом, т.е. глубина цвета равна 8. Следовательно, количество оттенков одного базового цвета равно $2^8 = 256$ (256 оттенков красного, 256 оттенков зеленого и 256 оттенков синего). Из трех базовых цветов можно получить $256 \times 256 \times 256 = (256)^3 = =(2^8)^3= 2^{24} = 16\ 777\ 216$ цветов и их оттенков. Информация о каждом пикселе в видеопамяти займет $n = 8 \times 3 = 24$ бита = 3 байта.

Таким образом, для хранения одного образа экрана потребуется объем памяти, равный произведению ширины экрана (в пикселях) на высоту экрана (в пикселях) и на n (глубину цвета). Например, при размере экрана 1280×720 формула примет вид: $1280 \times 720 \times 24$.

В общем случае объем памяти, необходимый для хранения растрового изображения, рассчитывается по формуле:

$$V = W \times H \times n \text{ (битов)},$$

где W – ширина изображения в точках (пикселях);

H – высота изображения в точках (пикселях);

V – объем памяти, необходимый для хранения изображения.

1 Рассчитайте объем видеопамяти, необходимой для хранения растрового изображения, занимающего весь экран монитора с разрешающей способностью 640 × 480 пикселей, если используется палитра из 65536 цветов.

Решение. Глубина цвета рассчитывается по формуле $n = \log_2 K =$ $= \log_2 65536 = 16$ битов. Следовательно, для хранения одного пикселя изображения требуется 16/8 = 2 байта видеопамяти. Таким образом, для хранения всего изображения потребуется 2 × 640 × 480 = 614400 байт = 600 Кбайт.

Ответ: 600 Кбайт.

2 Для хранения растрового изображения размером 320×400 пикселей потребовалось 125 Кбайт памяти. Определите количество цветов в палитре.

Решение. Изображение состоит из 320 × 400 = 128000 точек. Для хранения этого изображения отводится 125 × 1024 × 8 = 1024000 битов памяти. Следовательно, для хранения одной точки нужно 1024000 / 128000 = 8 битов памяти. Тогда $n = 8$ бит, а палитра содержит $K = 2^n = 2^8 = 256$ цветов.

Ответ: 256 цветов.

3 (*Демо-версия 2009 г., задача А15*)
Для кодирования цвета фона страницы Интернет используется атрибут **bgcolor= «#FFFFFF»**, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели. Какой цвет будет у страницы, заданной тэгом **<body bgcolor=«#FFFFFF»>**?

1) белый 2) зеленый 3) красный 4) синий

Решение. При описании Интернет-страницы на языке HTML допускается описывать цвет в виде 16-ричного числа, состоящего ровно из 6 цифр. Правила таковы: под каждый цвет модели RGB (как уже отмечалось, именно в этой модели представляется цвет на экране монитора) отводится 2 цифры.
Чтобы узнать вклад каждого базового цвета, последовательность «FFFFFF» делим на 3 группы по две цифры в каждой:

FF – красный FF – зеленый FF – синий.

Другими словами, запись «FFFFFF» мы представили как «RRGGBB».

Числа, закодированные двумя 16-ричными цифрами, означают яркость цвета. $FF_{16} = 255_{10}$, что означает максимальную яркость.

При максимальной яркости каждого цвета в модели RGB получается белый цвет.

Ответ: белый цвет.

При переводе в рассматриваемое представление базовых цветов правила таковы:

- яркость данного базового цвета должна быть максимальной (255_{10} или FF_{16});
- остальные базовые цвета отсутствуют, т.е. их яркость минимальная (00_{16} или 0_{10}).

Далее в таблице приведены примеры некоторых цветов и их представление в 16-ричном виде.

Название цвета	**16-ричное представление**		
	Красный (R)	**Зеленый (G)**	**Синий (B)**
Белый	FF	FF	FF
Красный	FF	00	00
Зеленый	00	FF	00
Синий	00	00	FF
Черный	00	00	00

Помимо растрового, существует и другой тип изображения – векторный. Векторные изображения представлены набором линий (как прямых, так и кривых) и элементарных фигур. Каждая линия или фигура описывается математической формулой с указанием ее характеристик (цвет, толщина и др.). Такое описание занимает меньше места в памяти, чем аналогичное изображение в растровом виде. Кроме того, при масштабировании объектов векторной графики качество изображения не изменяется.

Кодирование текстовой информации

Кодирование текстовой информации заключается в том, что каждому текстовому символу (букве, цифре, знаку препинания и др.) приписывается уникальный код – целое число. В зависимости от количества битов, отведенных под кодирование символов, все виды кодировок де-

лятся на две группы: 8-разрядные и 16-разрядные. Для каждого вида кодировки символы вместе с их кодами образуют кодировочную таблицу.

В 8-разрядной кодировке для кодирования одного символа отводится 8 битов (1 байт). С их помощью можно записать 2^8=256 разных целых чисел и, следовательно, закодировать 256 различных символов.

В кодировочной таблице первая половина кодов отводится под кодирование управляющих (невидимых) символов, а также букв английского алфавита, цифр и знаков препинания. Оставшаяся часть отведена под кодирование символов национальных алфавитов. В результате народы, говорящие на разных языках (не на английском), не могут использовать одну и ту же кодировочную таблицу. Чтобы правильно отобразить текст на экране монитора, необходимо выбрать для него подходящую кодировку. Это делает невозможным правильное восприятие текста на любом языке, кроме английского.

К 8-разрядным кодировкам, включающим в себя кодировку символов русского языка, относятся ASCII, ДКОИ-8, Win 1251 (или CP1251).

16-разрядная кодировка Unicode позволяет представить 2^{16}=65536 различных символов. В кодовой таблице Unicode присутствуют символы всех современных национальных языков, при этом символы первых 128 кодов совпадают с ASCII.

4 Определите информационный объем пословицы в битах в кодировке Unicode: `Мал золотник, да дорог.`

Указание. Нужно подсчитать количество символов в предложенной фразе, включая пробелы и знаки препинания. (При этом подразумевается, что между словами стоит ровно один пробел, а перед знаками препинания пробелов нет.) Полученное значение надо умножить на 16, чтобы получить ответ в битах.

Ответ: 368 бит.

Задачи к разделу «Информация и ее кодирование»

1 Считая, что каждый символ кодируется двумя байтами, оцените информационный объем в битах следующего предложения:

`Один пуд – около 16,4 килограмм.`

2 Считая, что каждый символ кодируется двумя байтами, оцените информационный объем в битах следующего предложения:

`У семи нянек дитя без глаза.`

3 Считая, что каждый символ кодируется одним байтом, оцените информационный объем в битах следующего предложения:

`Один дюйм равен 2,54 сантиметра.`

4 Считая, что каждый символ кодируется двумя байтами, оцените информационный объем в битах следующего предложения:

```
Коренное население Америки - индейцы.
```

5 Считая, что каждый символ кодируется одним байтом, оцените информационный объем в битах следующего предложения:

```
Информатика - наука об информации и информационных процессах.
```

6 Информационный объем предложения

```
Одна морская миля равна 1852 метра.
```

составляет 280 бит. Определите, сколькими байтами кодируется один символ.

7 Информационный объем предложения

```
Скорость в один узел равна 1,852 км/ч.
```

составляет 608 бит. Определите, сколькими байтами кодируется один символ.

8 Информационный объем предложения

```
Кашу маслом не испортишь.
```

составляет 50 байт. Определите, сколькими битами кодируется один символ.

9 Информационный объем предложения

```
Третья информационная революция произошла в конце XIX века и была связана с изобретением электричества.
```

составляет 103 байта. Определите, сколькими битами кодируется один символ.

10 Информационный объем предложения

```
Один фут - около 30 сантиметров.
```

составляет 512 бит. Определите, сколькими байтами кодируется один символ.

11 Сколько мегабайт информации содержит сообщение объемом 2^{28} бит?

12 Сколько мегабайт информации содержит сообщение объемом 2^{33} бит?

13 Сколько мегабайт информации содержит сообщение объемом 2^{25} бит?

14 Сколько килобайт информации содержит сообщение объемом 2^{26} бит?

15 Сколько килобайт информации содержит сообщение объемом 2^{17} бит?

16 Сколько битов информации содержит сообщение объемом в 8 мегабайт?

17 Сколько битов информации содержит сообщение объемом в 16 мегабайт?

18 Сколько байтов информации содержит сообщение объемом в 32 мегабайта?

19 Сколько байтов информации содержит сообщение объемом в 4 гигабайта?

20 Сколько битов информации содержит сообщение объемом в 8 гигабайт?

21 Для хранения растрового изображения размером 160 x 128 пикселей отвели 5 килобайт памяти. Каково максимально возможное количество цветов в палитре изображения?

22 Для хранения растрового изображения размером 64 x 128 пикселей отвели 3 килобайта памяти. Каково максимально возможное количество цветов в палитре изображения?

23 Для хранения растрового изображения размером 64 x 32 пикселей отвели 256 байт памяти. Каково максимально возможное количество цветов в палитре изображения?

24 Для хранения растрового изображения размером 32 x 32 пикселей отвели 512 байт памяти. Каково максимально возможное количество цветов в палитре изображения?

25 Для хранения растрового изображения размером 64 x 64 пикселей отвели 3 килобайта памяти. Каково максимально возможное количество цветов в палитре изображения?

26 Какой объем памяти необходимо выделить под хранение растрового изображения размером 128 x 128 пикселей, если в палитре изображения 64 цвета?

27 Какой объем памяти необходимо выделить под хранение растрового изображения размером 128 x 128 пикселей, если в палитре изображения 256 цветов?

28 Какой объем памяти необходимо выделить под хранение растрового изображения размером 640 x 480 пикселей, если в палитре изображения 16 миллионов цветов?

29 Какой объем памяти необходимо выделить под хранение растрового изображения размером 240 x 192 пикселей, если в палитре изображения 65 тысяч цветов?

30 Какой объем памяти необходимо выделить под хранение растрового изображения размером 256 x 512 пикселей, если в палитре изображения 16 цветов?

31 Скорость передачи данных через ADSL-соединение равна 2 500 000 бит/с. Через данное соединение передают файл размером 10 мегабайт. Определите время передачи файла в секундах. (Служебной информацией пренебречь.)

32 Скорость передачи данных через ADSL-соединение равна 5 000 000 бит/с. Через данное соединение передают файл размером 1 гигабайт. Определите время передачи файла в минутах. (Служебной информацией пренебречь.)

33 Скорость передачи данных через ADSL-соединение равна 256 000 бит/с. Через данное соединение передают файл размером 3,5 мегабайта. Определите время передачи файла в секундах. (Служебной информацией пренебречь.)

34 Скорость передачи данных через ADSL-соединение равна 1 000 000 бит/с. Через данное соединение передают файл размером 100 мегабайт. Определите время передачи файла в минутах. (Служебной информацией пренебречь.)

35 Скорость передачи данных через ADSL-соединение равна 128 000 бит/с. Через данное соединение передают файл размером 8 мегабайт. Определите время передачи файла в секундах. (Служебной информацией пренебречь.)

36 Скорость передачи данных через ADSL-соединение равна 256 000 бит/с. Определите наибольший размер файла, который может быть передан через данное соединение за 10 минут. (Служебной информацией пренебречь.)

37 Скорость передачи данных через ADSL-соединение равна 128 000 бит/с. Определите наибольший размер файла, который может быть передан через данное соединение за 7 минут. (Служебной информацией пренебречь.)

38 Скорость передачи данных через ADSL-соединение равна 512 000 бит/с. Определите наибольший размер файла, который может быть передан через данное соединение за 3 минуты. (Служебной информацией пренебречь.)

39 Скорость передачи данных через ADSL-соединение равна 1 000 000 бит/с. Определите наибольший размер файла, который может быть передан через данное соединение за 11 секунд. (Служебной информацией пренебречь.)

40 Скорость передачи данных через ADSL-соединение равна 3 000 000 бит/с. Определите наибольший размер файла, который может быть передан через данное соединение за 9 секунд. (Служебной информацией пренебречь.)

41 Сколько секунд потребуется модему, передающему информацию со скоростью 19200 бит/с, чтобы передать цветное растровое изображение размером 1024 x 768 пикселей при условии, что в палитре 65 тысяч цветов? Результат представьте целым числом. (Служебной информацией пренебречь.)

42 Сколько секунд потребуется модему, передающему информацию со скоростью 14400 бит/с, чтобы передать цветное растровое изображение размером 800 x 600 пикселей при условии, что в палитре 256 цветов? Результат представьте целым числом. (Служебной информацией пренебречь.)

43 Сколько секунд потребуется модему, передающему информацию со скоростью 33600 бит/с, чтобы передать цветное растровое изображение размером 640 x 480 пикселей при условии, что в палитре 16 млн цветов? Результат представьте целым числом. (Служебной информацией пренебречь.)

44 Сколько секунд потребуется модему, передающему информацию со скоростью 9600 бит/с, чтобы передать цветное растровое изображение размером 1024 x 768 пикселей при условии, что в палитре 16 цветов? Результат представьте целым числом. (Служебной информацией пренебречь.)

45 Сколько секунд потребуется модему, передающему информацию со скоростью 24000 бит/с, чтобы передать цветное растровое изображение размером 1440 x 800 пикселей при условии, что в палитре 65 тысяч цветов? Результат представьте целым числом. (Служебной информацией пренебречь.)

46 Сколько секунд потребуется модему, передающему информацию со скоростью 28800 бит/с, чтобы передать 100 страниц текста в 40 строк по 60 символов каждая при условии, что каждый символ кодируется двумя байтами? Результат представьте целым числом. (Служебной информацией пренебречь.)

47 Сколько секунд потребуется модему, передающему информацию со скоростью 33600 бит/с, чтобы передать 250 страниц текста в 45 строк по 63 символов каждая при условии, что каждый символ кодируется одним байтом? Результат представьте целым числом. (Служебной информацией пренебречь.)

48 Сколько секунд потребуется модему, передающему информацию со скоростью 14400 бит/с, чтобы передать 120 страниц текста в 30 строк по 45 сим-

волов каждая при условии, что каждый символ кодируется двумя байтами? Результат представьте целым числом. (Служебной информацией пренебречь.)

49 Сколько секунд потребуется модему, передающему информацию со скоростью 19200 бит/с, чтобы передать 150 страниц текста в 35 строк по 50 символов каждая при условии, что каждый символ кодируется одним байтом? Результат представьте целым числом. (Служебной информацией пренебречь.)

50 Сколько секунд потребуется модему, передающему информацию со скоростью 9600 бит/с, чтобы передать 70 страниц текста в 60 строк по 75 символов каждая при условии, что каждый символ кодируется двумя байтами? Результат представьте целым числом. (Служебной информацией пренебречь.)

51 Световое табло состоит из лампочек, каждая из которых может находиться в трех состояниях: «включено», «выключено» или «мигает». Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 195 различных сигналов?

52 Световое табло состоит из лампочек, каждая из которых может находиться в трех состояниях: «включено», «выключено» или «мигает». Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 75 различных сигналов?

53 Световое табло состоит из лампочек, каждая из которых может находиться в трех состояниях: «включено», «выключено» или «мигает». Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 255 различных сигналов?

54 Световое табло состоит из лампочек, каждая из которых может находиться в двух состояниях: «включено» или «выключено». Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 196 различных сигналов?

55 Световое табло состоит из лампочек, каждая из которых может находиться в двух состояниях: «включено» или «выключено». Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 64 различных сигнала?

56 Световое табло состоит из лампочек, каждая из которых может находиться в трех состояниях: «включено», «выключено» или «мигает». Сколько различных сигналов можно передать при помощи такого табло, если на нем 6 лампочек?

57 Световое табло состоит из лампочек, каждая из которых может находиться в трех состояниях: «включено», «выключено» или «мигает». Сколько различных сигналов можно передать при помощи такого табло, если на нем 5 лампочек?

58 Световое табло состоит из лампочек, каждая из которых может находиться в трех состояниях: «включено», «выключено» или «мигает». Сколько различных сигналов можно передать при помощи такого табло, если на нем 4 лампочки?

59 Световое табло состоит из лампочек, каждая из которых может находиться в двух состояниях: «включено» или «выключено». Сколько различных сигналов можно передать при помощи такого табло, если на нем 7 лампочек?

60 Световое табло состоит из лампочек, каждая из которых может находиться в двух состояниях: «включено» или «выключено». Сколько различных сигналов можно передать при помощи такого табло, если на нем 10 лампочек?

61 Компьютерная игра состоит из 16 уровней, на каждом из которых игроку нужно отыскать восемь секретных ключей. При переходе с уровня на уровень у игрока остаются все найденные ключи. Какое минимальное количество битов потребуется для кодирования секретных ключей?

62 Шахматная доска состоит из 64 полей: 8 столбцов на 8 строк. Какое минимальное количество битов потребуется для кодирования координат одного шахматного поля?

63 В зрительном зале две прямоугольные области зрительских кресел: одна 10 на 12, а другая 17 на 8. Какое минимальное количество битов потребуется для кодирования каждого места в автоматизированной системе учета продажи билетов?

64 Некоторое игровое поле состоит из 28 клеток. Какое минимальное количество битов потребуется для кодирования каждой клетки поля?

65 В двенадцатиэтажном одноподъездном доме устанавливают цифровой домофон. Какое минимальное количество битов потребуется для кодирования номера каждой квартиры, если на каждом этаже 8 квартир?

66 Сколько существует различных последовательностей из символов «плюс» и «минус» длиной ровно 5 символов?

67 Сколько существует различных последовательностей из символов «X» и «O» длиной ровно 6 символов?

68 Сколько существует различных последовательностей из символов «плюс» и «минус» длиной ровно 7 символов?

69 Сколько существует различных последовательностей из символов «A», «B», «C» и «D» длиной ровно 3 символа?

70 Сколько существует различных последовательностей из символов «А», «Б» и «В» длиной ровно 4 символа?

71 Алфавит некоторого языка состоит из 4 символов. Оцените информационный объем сообщения в байтах длиной в 16 символов.

72 Алфавит некоторого языка состоит из 7 символов. Оцените информационный объем сообщения в битах длиной в 21 символ.

73 Алфавит некоторого языка состоит из 12 символов. Оцените информационный объем сообщения в байтах длиной в 64 символа.

74 Алфавит некоторого языка состоит из 20 символов. Оцените информационный объем сообщения длиной в 40 символов.

75 Алфавит некоторого языка состоит из 54 символов. Оцените информационный объем сообщения в байтах длиной в 120 символов.

76 Информационный объем сообщения длиной в 64 символа составляет 24 байта. Определите, из какого максимального количества символов может состоять алфавит языка, на котором написано данное сообщение.

77 Информационный объем сообщения длиной в 24 символа составляет 12 байт. Определите, из какого максимального количества символов может состоять алфавит языка, на котором написано данное сообщение.

78 Информационный объем сообщения длиной в 144 символа составляет 36 байт. Определите, из какого максимального количества символов может состоять алфавит языка, на котором написано данное сообщение.

79 Информационный объем сообщения длиной в 120 символа составляет 75 байт. Определите, из какого максимального количества символов может состоять алфавит языка, на котором написано данное сообщение.

80 Информационный объем сообщения длиной в 204 символа составляет 51 байт. Определите, из какого максимального количества символов может состоять алфавит языка, на котором написано данное сообщение.

81 Метеорологическая станция ведет наблюдение за направлением ветра. Результатом одного измерения является одно из восьми возможных направлений, которое записывается при помощи минимально возможного количества битов. Станция сделала 384 измерения. Каков информационный объем результатов наблюдений? Ответ укажите в байтах.

82 Метеорологическая станция ведет наблюдение за направлением ветра. Результатом одного измерения является одно из восьми возможных направлений, которое записывается при помощи минимально возможного количества битов. Станция сделала 264 измерения. Каков информационный объем результатов наблюдений? Ответ укажите в байтах.

83 Метеорологическая станция ведет наблюдение за направлением ветра. Результатом одного измерения является одно из восьми возможных направлений, которое записывается при помощи минимально возможного количества битов. Станция сделала 216 измерений. Каков информационный объем результатов наблюдений? Ответ укажите в байтах.

84 Метеорологическая станция ведет наблюдение за направлением ветра. Результатом одного измерения является одно из восьми возможных направлений, которое записывается при помощи минимально возможного количества битов. Станция сделала 360 измерений. Каков информационный объем результатов наблюдений? Ответ укажите в байтах.

85 Метеорологическая станция ведет наблюдение за направлением ветра. Результатом одного измерения является одно из восьми возможных направлений, которое записывается при помощи минимально возможного количества битов. Станция сделала 408 измерений. Каков информационный объем результатов наблюдений? Ответ укажите в байтах.

86 Метеорологическая станция ведет наблюдение за температурой воздуха. Считается, что температура должна быть представлена целым числом. При этом она не может опускаться ниже минус 43 градусов и подниматься выше плюс 51 градуса. Каждое значение температуры записывается при помощи минимально возможного количества битов. За некоторый период времени станция сделала 112 измерений. Каков информационный объем результатов наблюдений?

87 Метеорологическая станция ведет наблюдение за температурой воздуха. Считается, что температура должна быть представлена целым числом. При этом она не может опускаться ниже минус 14 градусов и подниматься выше плюс 18 градусов. Каждое значение температуры записывается

при помощи минимально возможного количества битов. За некоторый период времени станция сделала 144 измерения. Каков информационный объем результатов наблюдений?

88 Метеорологическая станция ведет наблюдение за температурой воздуха. Считается, что температура должна быть представлена целым числом. При этом она не может опускаться ниже минус 8 градусов и подниматься выше плюс 8 градусов. Каждое значение температуры записывается при помощи минимально возможного количества битов. За некоторый период времени станция сделала 40 измерений. Каков информационный объем результатов наблюдений?

89 Метеорологическая станция ведет наблюдение за температурой воздуха. Считается, что температура должна быть представлена целым числом. При этом она не может опускаться ниже минус 28 градусов и подниматься выше плюс 36 градусов. Каждое значение температуры записывается при помощи минимально возможного количества битов. За некоторый период времени станция сделала 168 измерений. Каков информационный объем результатов наблюдений?

90 Метеорологическая станция ведет наблюдение за температурой воздуха. Считается, что температура должна быть представлена целым числом. При этом она не может опускаться ниже минус 25 градусов и подниматься выше плюс 25 градусов. Каждое значение температуры записывается при помощи минимально возможного количества битов. За некоторый период времени станция сделала 96 измерений. Каков информационный объем результатов наблюдений?

91 Обычный дорожный светофор без дополнительных секций подает шесть видов сигналов (непрерывные красный, желтый и зеленый, мигающие желтый и зеленый, красный и желтый одновременно). Электронное устройство управления светофором последовательно воспроизводит записанные сигналы. Подряд записано 96 сигналов светофора. Оцените данный информационный объем.

92 Обычный дорожный светофор без дополнительных секций подает шесть видов сигналов (непрерывные красный, желтый и зеленый, мигающие желтый и зеленый, красный и желтый одновременно). Электронное устройство управления светофором последовательно воспроизводит записанные сигналы. Подряд записано 45 сигналов светофора. Оцените данный информационный объем.

93 Обычный дорожный светофор без дополнительных секций подает шесть видов сигналов (непрерывные красный, желтый и зеленый, мигающие желтый и зеленый, красный и желтый одновременно). Электронное устройство управления светофором последовательно воспроизводит записанные сигналы. Подряд записано 72 сигнала светофора. Оцените данный информационный объем.

94 Обычный дорожный светофор без дополнительных секций подает шесть видов сигналов (непрерывные красный, желтый и зеленый, мигающие желтый и зеленый, красный и желтый одновременно). Электронное устройство управления светофором последовательно воспроизводит записанные сигналы. Подряд записано 120 сигналов светофора. Оцените данный информационный объем.

95 Обычный дорожный светофор без дополнительных секций подает шесть видов сигналов (непрерывные красный, желтый и зеленый, мигающие желтый и зеленый, красный и желтый одновременно). Электронное устройство управления светофором последовательно воспроизводит записанные сигналы. Подряд записано 144 сигнала светофора. Оцените данный информационный объем.

96 Цифровой вольтметр измеряет величину напряжения с точностью до 0,01 В. Определите минимальную разрядность аналого-цифрового преобразователя (количество битов на каждое значение напряжения), если максимальное напряжение, которое может измерить данный вольтметр, составляет 6 В.

97 Цифровой вольтметр измеряет величину напряжения с точностью до 0,01 В. Определите минимальную разрядность аналого-цифрового преобразователя (количество битов на каждое значение напряжения), если максимальное напряжение, которое может измерить данный вольтметр, составляет 5 В.

98 Цифровой вольтметр измеряет величину напряжения с точностью до 0,1 В. Определите минимальную разрядность аналого-цифрового преобразователя (количество битов на каждое значение напряжения), если максимальное напряжение, которое может измерить данный вольтметр, составляет 12 В.

99 Цифровой вольтметр измеряет величину напряжения с точностью до 0,1 В. Определите минимальную разрядность аналого-цифрового преоб-

разователя (количество битов на каждое значение напряжения), если максимальное напряжение, которое может измерить данный вольтметр, составляет 20 В.

100 Цифровой вольтметр измеряет величину напряжения с точностью до 0,1 В. Определите минимальную разрядность аналого-цифрового преобразователя (количество битов на каждое значение напряжения), если максимальное напряжение, которое может измерить данный вольтметр, составляет 5 В.

Глава 2

Основы математической логики

Алгебра логики

Алгебра логики (алгебра высказываний) – это формальная логическая теория, раздел математической логики, разработанный в XIX в. в трудах английского математика Джорджа Буля. В алгебре логики используются *алгебраические методы* для решения логических задач.

Объектами алгебры логики являются *высказывания.*

Алгебра логики предоставляет математический аппарат, с помощью которого записывают (кодируют, формализуют), упрощают, вычисляют и преобразовывают логические высказывания.

Логическое высказывание – это повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно. В алгебре логики отвлекаются от содержания (смысла) высказывания и рассматривают лишь его значение.

В булевой алгебре (алгебре логики) возможны только два значения для логических высказываний и логических констант:

- ИСТИНА – обозначается как True или 1;
- ЛОЖЬ – обозначается как False или 0.

Примеры высказываний: «сегодня теплое утро», «$\sqrt{5}$ – иррациональное число».

Не являются высказываниями следующие предложения:

- «пойдет ли сегодня дождь?» – вопросительное предложение, не имеющее своим значением ИСТИНА или ЛОЖЬ;
- «сделайте домашнее задание» – повелительное предложение; невозможно определить значение высказывания – истинно оно или ложно;
- «это предложение ложно» – противоречивое утверждение.

Законы алгебры логики помогают определить истинность или ложность составных высказываний, не вникая в их содержание.Поэтому простым высказываниям ставятся в соответствие логические переменные. Например: A – для высказывания «Андрей Иванов – врач», B – «Андрей Иванов – шахматист», x_1 – «этот карандаш красного цвета», x_2 – «этот карандаш синего цвета».

Над простыми высказываниями в алгебре логики определяют логические операции, в результате действия которых получаются новые составные высказывания.

Основные логические операции в алгебре логики

В алгебре логики существуют три основные операции.

1. *Логическое отрицание* (*инверсия*).
   Обозначается: $\bar{A}$, $\neg A$, **not** A, **не** A.
   Высказывание $\neg A$ истинно при ложном A и ложно при истинном A.

2. *Логическое умножение* (*конъюнкция*).
   Обозначается $A\&B$, A **and** B, $A \cdot B$, $A \wedge B$, AB, A **и** B.
   Высказывание $A \wedge B$ истинно тогда и только тогда, когда оба высказывания A и B истинны.

3. *Логическое сложение* (*дизъюнкция*).
   Обозначается: $A \vee B$, A **or** B, $A + B$, A **или** B.
   Высказывание $A \vee B$ ложно тогда и только тогда, когда оба высказывания A и B ложны.

Остальные операции алгебры логики выражаются через первые три операции – отрицание, конъюнкцию и дизъюнкцию. Перечислим их.

4. *Логическое следование* (*импликация*).
   Обозначается: $A \rightarrow B$, $A \Rightarrow B$.
   Высказывание $A \rightarrow B$ ложно только тогда, когда A истинно, а B ложно.
   Важно: в операции импликации посылка A не обязана быть истинной, в отличие от логического оператора в языках программирования «если A то B».
   Импликация выражается через дизъюнкцию и отрицание:
   $A \Rightarrow B = \neg A \vee B$.

5. *Эквивалентность* (равносильность, необходимость и достаточность).
   Обозначается: $A \sim B$, $A \Leftrightarrow B$, $A \equiv B$.
   Высказывание $A \Leftrightarrow B$ истинно тогда и только тогда, когда значения A и B совпадают.
   Эквивалентность выражается через отрицание, дизъюнкцию и конъюнкцию:
   $A \Leftrightarrow B = (\neg A \vee B) \wedge (\neg B \vee A)$.

6. *Исключающее ИЛИ.*
   Обозначается: A **XOR** B, $A \oplus B$
   Высказывание A **XOR** B истинно, когда A и B не равны.

Порядок выполнения логических операций задается круглыми скобками. При отсутствии скобок порядок выполнения операций следующий: отрицание, конъюнкция, дизъюнкция, исключающее ИЛИ, импликация, эквивалентность.

Составное высказывание (логическая формула) состоит из нескольких высказываний, соединенных логическими операциями. Исходные высказывания

могут быть логическими переменными или логическими константами (имеющими постоянное значение ИСТИНА или ЛОЖЬ).

Логическая функция определяется на множестве логических переменных и логических констант, принимающих значение ИСТИНА или ЛОЖЬ. Значение функции вычисляется в результате выполнения логических операций с (или над) логическими операндами. Например,

$$F(A, B, C) = A \wedge (\neg B \vee C); \qquad F(x_1, x_2, x_3) = \neg x_1 \vee x_2 \wedge \neg x_3.$$

Логическую функцию можно задать двумя способами: логической формулой или таблицей истинности. Таблица истинности задает значения функции на всех возможных наборах ее переменных.

Таблицы истинности простейших логических функций:

A	$\neg A$
0	1
1	0

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

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

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

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

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Законы алгебры логики

Закон	Для ИЛИ	Для И
Коммутативный (переместительный): логические переменные можно менять местами	$X \vee Y = Y \vee X$	$X \wedge Y = Y \wedge X$
Ассоциативный (сочетательный): логические переменные в дизъюнкциях и конъюнкциях можно объединять в группы	$(X \vee Y) \vee Z = X \vee (Y \vee Z)$	$(X \wedge Y) \wedge Z = X \wedge (Y \wedge Z)$
Дистрибутивный (распределительный): одинаковые переменные в дизъюнкциях и конъюнкциях можно выносить за скобки	$(X \vee Y) \wedge Z =$ $= (X \wedge Z) \vee (Y \wedge Z)$	$(X \wedge Y) \vee Z =$ $= (X \vee Z) \wedge (Y \vee Z)$

Окончание таблицы

Закон	Для ИЛИ	Для И
Закон непротиворечия: высказывание может быть только истинным или ложным, третьего не дано		$X \wedge \neg X = 0$
Закон исключенного третьего: высказывание может быть только истинным или ложным, третьего не дано	$X \vee \neg X = 1$	
Правила де Моргана	$\neg (X \vee Y) = \neg X \wedge \neg Y$	$\neg (X \wedge Y) = \neg X \vee \neg Y$
Законы склеивания	$(X \wedge Y) \vee (\neg X \wedge Y) = Y$	$(X \vee Y) \wedge (\neg X \vee Y) = Y$
Исключение констант	$X \vee 0 = X,\ X \vee 1 = 1$	$X \wedge 0 = 0,\ X \wedge 1 = X$
Двойного отрицания: двойное отрицание исключает отрицание	$\neg\neg X = X$	
Идемпотентности	$X \vee X = X$	$X \wedge X = X$
Контрапозиции	$A \to B = \neg B \to \neg A$	
Снятие импликации	$X \to Y = \neg X \vee Y$	
Снятие эквивалентности	$A \leftrightarrow B = (A \wedge B) \vee (\neg A \wedge \neg B)$ $A \leftrightarrow B = (A \vee \neg B) \wedge (\neg A \vee B)$	
Закон поглощения	$A \vee (A \wedge C) = A$	$A \wedge (A \vee C) = A$

Преобразование логических выражений

Под преобразованием логических выражений или упрощением логической формулы понимается изменение исходного логического выражения в соответствии с законами алгебры логики, приводящее к логической формуле, в которой меньше операций конъюнкции и дизъюнкции и нет отрицаний неэлементарных формул. Также выражение считается упрощенным, если получившаяся формула содержит меньшее количество переменных.

1 $\neg (x \wedge \neg x) \vee (y \vee \neg y) = \neg 0 \vee 1 = 1.$

2

$\neg(x \vee y) \wedge (x \wedge \neg y) =$ *Правило де Моргана*

$= \neg x \wedge \neg y \wedge (x \wedge \neg y) =$ *Ассоциативный закон*

$= \neg x \wedge x \wedge \neg y \wedge \neg y =$ *Операция переменной с ее инверсией*

$= 0 \wedge \neg y \wedge \neg y = 0.$ *Операция с константами*

3

$\neg x \wedge y \vee \neg(x \vee y) \vee x =$ *Правило де Моргана*

$= \neg x \wedge y \vee \neg x \wedge \neg y \vee x =$ *Выносится за скобки общий множитель*

$= \neg x \wedge (y \vee \neg y) \vee x = \neg x \vee x = 1.$ *Правило операции переменной с ее инверсией*

4 $(x \vee y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y) =$

Повторяется второй множитель – разрешено законом идемпотентности

$= (x \vee y) \wedge (\neg x \vee y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y) =$

Затем группируются два первых и два последних множителя, к каждой группе из двух множителей применяется операция склеивания

$= y \wedge \neg x.$

5 $\neg(x \wedge y \vee \neg z) =$ *Снимаем знак отрицания, чтобы он стоял перед отдельными переменными, а не перед их комбинациями – правило де Моргана*

$= \neg(x \wedge y) \wedge \neg\neg z =$ *Правило де Моргана и закон двойного отрицания*

$= (\neg x \vee \neg y) \wedge z.$

6 $x \wedge y \vee x \wedge y \wedge z \vee x \wedge z \wedge p = x \wedge (y \wedge (1 \vee z) \vee z \wedge p) =$
$= x \wedge (y \vee z \wedge p) = x \wedge y \vee x \wedge z \wedge p.$

Выносятся за скобки общие множители, применяется правило операций с константами

Построение таблиц истинности для логических выражений

Таблица истинности логической формулы (функции) выражает соответствие между всеми возможными наборами значений логических переменных и значением функции.

Для функции от двух переменных существует $2^2 = 4$ комбинации наборов значений переменных; для функции трех переменных – $2^3 = 8$; для функции четырех переменных – $2^4 = 16$ комбинаций значений наборов переменных.

В общем случае для функции от N переменных количество строк M в таблице истинности вычисляется по формуле:

$$M = 2^N.$$

Последовательность построения таблицы истинности:

1) определить количество N используемых переменных в логическом выражении;

2) вычислить количество всевозможных наборов значений переменных $M = 2^N$, равное количеству строк в таблице истинности;

3) подсчитать количество логических операций в логическом выражении и определить число столбцов в таблице, которое равно сумме числа переменных и количества логических операций;

4) озаглавить столбцы таблицы названиями переменных и названиями логических операций;
5) заполнить столбцы логических переменных наборами значений, например, от 0000 до 1111 с шагом 0001 в случае для четырех переменных;
6) заполнить таблицу истинности по столбцам со значениями промежуточных операций слева направо;
7) заполнить окончательный столбец значений для функции F.

Далее приводятся задачи на построение таблицы истинности по логической формуле.

1 Функция с двумя переменными: $F(x,y) = \neg x \wedge y \vee \neg (x \vee y) \vee x$.

Переменные		Промежуточные логические формулы					Функция F
x	y	$\neg x$	$\neg x \wedge y$	$x \vee y$	$\neg (x \vee y)$	$\neg x \wedge y \vee \neg (x \vee y)$	$\neg x \wedge y \vee \neg (x \vee y) \vee x$
0	0	1	0	0	1	1	1
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	1	0	0	1	0	0	1

2 Функция с тремя переменными: $F = \neg (x \vee \neg y) \vee \neg x \wedge z$.

Переменные			Промежуточные логические формулы					Функция F
x	y	z	$\neg y$	$x \vee \neg y$	$\neg (x \vee \neg y)$	$\neg x$	$\neg x \wedge z$	$\neg (x \vee \neg y) \vee \neg x \wedge z$
0	0	0	1	1	0	1	0	0
0	0	1	1	1	0	1	1	1
0	1	0	0	0	1	1	0	1
0	1	1	0	0	1	1	1	1
1	0	0	1	1	0	0	0	0
1	0	1	1	1	0	0	0	0
1	1	0	0	1	0	0	0	0
1	1	1	0	1	0	0	0	0

Построение логической функции по таблице истинности

Если известна таблица истинности некоторой функции, то для построения формулы логической функции необходимо построить дизъюнкцию всех полных элементарных конъюнкций на всех наборах переменных, принимающих значение 1.

Говорят, что функция представлена в *дизъюнктивной нормальной форме* (*ДНФ*), если она представлена только в виде трех основных логических операций (отрицания, конъюнкции, дизъюнкции) и не содержит отрицаний неэлементарных формул. Например:

$(A \wedge \neg B \wedge C) \vee (\neg A \wedge B) \vee (\neg B \wedge \neg C)$.

Любое логическое выражение можно привести к ДНФ.

ДНФ называется *совершенной*, если все конъюнкции состоят из одного и того же набора переменных, причем каждая переменная входит только один раз (возможно, с отрицанием). Например:

$(A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge \neg C)$.

Правила построения логической функции по таблице истинности лучше всего разобрать на примере.

1 Пусть задана полная таблица истинности некоторой функции:

x	*y*	*z*	*F*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

1. Из таблицы надо удалить строки, в которых значение функции равно 0:

x	*y*	*z*	*F*
0	1	0	1
1	0	0	1
1	0	1	1
1	1	1	1

2. Для каждой строки, в которой значение функции равно 1, составляется конъюнкция из всех переменных. Переменная входит в конъюнкцию со знаком отрицания, если эта переменная в этой строке равна 0:

x	*y*	*z*	*F*	**Вспомогательный столбец**
0	1	0	1	$\neg x \wedge y \wedge \neg z$
1	0	0	1	$x \wedge \neg y \wedge \neg z$
1	0	1	1	$x \wedge \neg y \wedge z$
1	1	1	1	$x \wedge y \wedge z$

3. Выражения, полученные в каждой строке, объединяются операцией дизъюнкции:

$F(x, y, z) = (\neg x \wedge y \wedge \neg z) \vee (x \wedge \neg y \wedge \neg z) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge z)$.

Разбор типовых задач ЕГЭ

1 (*Демо-вариант 2009 г., задача А7*)
Для какого из указанных значений X истинно высказывание $\neg((X > 2) \rightarrow (X>3))$?

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

Решение. К решению данного типа задач следует приступать «с конца». Последняя операция в выражении – отрицание к импликации. Построим таблицу истинности для операции отрицания импликации:

A	B	$A \rightarrow B$	$\neg (A \rightarrow B)$
0	0	1	0
0	1	1	0
1	0	0	1
1	1	1	0

В задаче ставится вопрос об истинности высказывания, т.е. нас интересует значение «1» в последнем столбце таблицы:

A	B	$\neg (A \rightarrow B)$
1	0	1

Вместо A и B вставляем выражения из исходной формулы:

$X > 2$	$X>3$	$\neg((X > 2) \rightarrow (X>3))$
1	0	1

Выражение $X > 2$ должно принимать значение «истина», т.е. $X > 2$.

Выражение $X > 3$ должно принимать значение «ложь», т.е. $X <= 3$.

Решая систему неравенств, получаем, что $X = 3$

Ответ: 3.

2 (*Демо-вариант 2009 г., задача А9*)
Символом F обозначено одно из указанных ниже логических выражений от трех аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F:

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

Какое выражение соответствует F?

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

Решение. Если подходить к решению задачи формально, то для каждого из четырех предложенных вариантов логических выражений надо вычислить значение выражения на трех наборах данных, представленных в таблице. Полученные ответы надо сравнить с теми, которые даны в последнем столбце. Если результат вычислений не совпадает с табличным хотя бы на одном наборе данных, то рассматриваемое выражение не является ответом.

Для данной задачи процесс решения можно сделать более коротким, если заметить, что:

1) выражение (2) – это конъюнкция трех переменных, которая на наборе «1 1 1» (последняя строка) принимает значение 1, а в таблице указан 0, то есть это выражение не может быть ответом;

2) выражение (3) – это дизъюнкция трех переменных, которая на наборе «1 1 1» (последняя строка) принимает значение 1, а в таблице указан 0, то есть это выражение тоже не может быть ответом.

Тем самым мы быстро отбросили два варианта, а для оставшихся двух придется делать вычисления для всех наборов данных, представленных в таблице.

Ответ: 4.

3 (*Демо-вариант 2007 г., задача А10*)
Какое логическое выражение равносильно выражению $\neg (A \wedge B) \wedge \neg C$?

1) $\neg A \vee B \vee \neg C$ 2) $(\neg A \vee \neg B) \wedge \neg C$
3) $(\neg A \vee \neg B) \wedge C$ 4) $\neg A \wedge \neg B \wedge \neg C$

Решение. Данную задачу можно решить двумя способами.

Способ 1. *Логические преобразования*

К выражению $\neg (A \wedge B)$ применяем правило де Моргана:

$\neg (A \wedge B) = (\neg A \vee \neg B)$.

В результате получим $(\neg A \vee \neg B) \wedge \neg C$, что соответствует ответу № 2.

Способ 2. *Табличный*

Для функции $\neg (A \wedge B) \wedge \neg C$ надо построить таблицу истинности. Это логическое выражение содержит три логические переменные, поэтому всего будет восемь наборов значений переменных, для которых нужно вычислить значение функции. Построение такой таблицы лучше производить поэтапно. По правилу приоритета логических операций сначала выполняется действие в скобках (т.е. конъюнкция $A \wedge B$), а затем к ней строится отрицание. На следующем шаге необходимо построить отрицание третьей логической переменной C (чему соответствует шестой столбец таблицы). Значение исходного выражения $\neg (A \wedge B) \wedge \neg C$ вычисляется как конъюнкция двух последних получившихся столбцов.

A	B	C	$A \wedge B$	$\neg (A \wedge B)$	$\neg C$	$\neg (A \wedge B) \wedge \neg C$
0	0	0	0	1	1	1
0	0	1	0	1	0	0
0	1	0	0	1	1	1
0	1	1	0	1	0	0
1	0	0	0	1	1	1
1	0	1	0	1	0	0
1	1	0	1	0	1	0
1	1	1	1	0	0	0

В качестве возможных вариантов ответа нам даны четыре выражения. Вычисляем их значения для каждого из входных наборов данных полученной таблицы, причем первое же несовпадение с данными последнего столбца означает, что рассматриваемая функция не является правильным ответом.

Ответ: 2.

4 Какое из приведенных имен удовлетворяет условию:

(Первая буква согласная) $\vee$ (Вторая буква гласная)$\rightarrow$(В слове 4 буквы)

1) МИХАИЛ 2) ГРИГОРИЙ 3) ЕВГЕНИЙ 4) ИОЛАНТА?

Решение. Введем обозначения для высказываний:

- первая буква согласная – A;
- вторая буква гласная – B;
- в слове 4 буквы – K.

Тогда исходное высказывание примет вид $(A \vee B) \rightarrow K$.

Преобразуем его: $\neg (A \vee B) \vee K = (\neg A \wedge \neg B) \vee K$.

Составим для высказывания таблицу истинности. При этом заметим, что ни в одном слове нет четырех букв, поэтому достаточно рассмотреть только случаи, когда K ложно.

A	B	$\neg A$	$\neg B$	$\neg A \wedge \neg B$	K	$(\neg A \wedge \neg B) \vee K$
0	0	1	1	1	0	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	0	0	0

Логическая функция принимает значение ИСТИНА только на одном наборе входных данных: первая буква – гласная, вторая – согласная и в слове не 4 буквы. Из предложенных вариантов ответов подходит только третий.

Ответ: 3.

5 Какое из приведенных имен удовлетворяет условию:

(Вторая буква гласная → Первая буква гласная) ∧ Последняя буква согласная

1) АЛИНА 2) МАРК 3) ДАРЬЯ 4) АЛЕКСАНДР?

Решение. Введем обозначения для высказываний:

- вторая буква гласная – A,
- первая буква гласная – B,
- последняя буква согласная – P

Тогда исходное высказывание примет вид $(A \rightarrow B) \wedge P$.

Составим для него таблицу истинности. При этом заметим, что последняя буква должна быть согласной, поэтому достаточно рассмотреть только те случаи, когда P истинно.

A	B	$A \rightarrow B$	P	$(A \rightarrow B) \wedge P$
0	0	1	1	1
0	1	1	1	1
1	0	0	1	0
1	1	1	1	1

Логическая функция принимает значение ИСТИНА на трех наборах входных данных:

1) первая, вторая и последняя буквы – согласные (ответ №4);

2) первая буква – гласная, вторая – согласная, последняя – согласная (таких вариантов нет);

3) первая и вторая буквы – гласные, последняя – согласная (таких вариантов нет).

Из предложенных вариантов ответа подходит только четвертый.

Ответ: 4.

6 Каково наименьшее натуральное число X, при котором истинно высказывание

$(X \cdot X<80) \rightarrow ((X+1) \cdot (X+1)>80)$?

Решение. Раскроем импликацию, получим $\neg (X^2 < 80) \vee ((X+1)^2 > 80)$.

Избавимся от отрицания: $(X^2 \geqslant 80) \vee ((X+1)^2 > 80)$.

Это выражение истинно, когда истинно либо $X^2 \geqslant 80$, либо $(X+1)^2 > 80$. Рассмотрим оба этих случая.

1. Так как речь идет о натуральных (т.е. положительных) числах, $X^2 > 80$ истинно при $X > 4\sqrt{5}$. Округлив, получаем $X > 9$.

2. $(X+1)^2 > 80$ истинно при $X + 1 > 4\sqrt{5}$ т.е $X > 4\sqrt{5} - 1$. Округлив, получим $X \geqslant 8$.

Сопоставив этих два результата, получаем $X \geqslant 8$.

Ответ: 8.

7 Укажите значения переменных K, L, M, N, при которых ложно логическое выражение

$$(\neg K \vee M) \rightarrow (\neg L \vee M \vee N).$$

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. Надо решить уравнение $(\neg K \vee M) \rightarrow (\neg L \vee M \vee N)=0$.

1. Преобразуем выражение, используя формулу замены импликации $A \rightarrow B = \neg A \vee B$:

$\neg (\neg K \vee M) \vee (\neg L \vee M \vee N) = 0$.

2. По закону де Моргана можно записать:

$(K \wedge \neg M) \vee (\neg L \vee M \vee N) = 0$.

3. Это логическое выражение будет ложным, когда ложны оба выражения в скобках. Тогда можно записать систему уравнений:

$$\begin{cases} \neg L \vee M \vee N = 0, & (1) \\ K \wedge \neg M = 0. & (2) \end{cases}$$

4. Из (1) получаем: $\neg L=0$, $M=0$, $N=0$, значит $L = 1$, $M = 0$, $N = 0$.

5. Подставим $M=0$ в (2), получим: $K \wedge 1 = 0$. Значит, $K = 0$.

6. Запишем значения переменных K, L, M и N в указанном порядке: 0100.

Ответ: 0100.

8 Сколько различных решений имеет уравнение

$((J \wedge \neg J) \vee \neg K \vee L) \wedge M = 1$,

где J, K, L, M – логические переменные? В ответе не нужно перечислять все различные наборы значений J, K, L, M, при которых выполнено данное равенство. В качестве ответа нужно указать количество таких наборов.

Решение. Надо решить уравнение $(\neg K \vee M) \rightarrow (\neg L \vee M \vee N)=0$.

1. Упростим исходное выражение. По закону непротиворечия $J \wedge \neg J = 0$. Запомним, что J может принимать любое значение. Получаем: $(0 \vee \neg K \vee L) \wedge M = 1$, что равносильно $(\neg K \vee L) \wedge M = 1$.

2. Полученное выражение истинно, когда истинны оба члена конъюнкции: $M = 1$ и $\neg K \vee L = 1$.

Нетрудно видеть, что полученная система двух уравнений имеет три различных решения, но при этом J – любое.

Ответ: 6.

9 Сколько различных решений имеет уравнение

$(\neg K \vee \neg L \vee \neg M) \wedge (L \vee \neg M \vee \neg N) = 0$,

где J, K, L, M – логические переменные? В ответе не нужно перечислять все различные наборы значений J, K, L, M, при которых выполнено данное равенство. В качестве ответа нужно указать количество таких наборов.

Решение. 1. Преобразуем выражение и вынесем $\neg M$ за скобку, используя дистрибутивный закон:
$\neg M \vee ((\neg \mathrm{K} \vee \neg L) \wedge (L \vee \neg N)) = 0.$

2. Полученное выражение ложно, когда оба члена дизъюнкции принимают значение «ложь», т.е. мы имеем систему из двух уравнений:

$$\begin{cases} \neg M = 0, & (1) \\ ((\neg K \vee \neg L) \wedge (L \vee \neg N)) = 0. & (2) \end{cases}$$

3. Из (1) следует, что $M = 1$.

4. Выражение (2) будет ложным, если ложно хотя бы одно выражение в скобках. Перепишем (2) как систему двух уравнений:

$$\begin{cases} \neg K \vee \neg L = 0, & (3) \\ L \vee \neg N = 0. & (4) \end{cases}$$

5. (3): $\neg K \vee \neg L = 0$, тогда $K = 1$ и Ш = 1, а N – любое (2 решения).

6. (4): $L \vee \neg N = 0$, тогда $L = 0$ и $N = 1$, а K – любое (еще 2 решения).

Общее количество решений равно сумме количеств решений каждого уравнения, то есть 4.

Ответ: 4.

10 Сколько различных решений имеет система уравнений

$$\begin{cases} X1 \vee \neg X2 = 1, & (1) \\ X2 \vee \neg X3 = 1, & (2) \\ \ldots \\ X9 \vee \neg X10 = 1. & (9) \end{cases}$$

Решение. Можно решать эту систему следующим образом.

Способ 1.

1. Предположим, что у нас есть только одно уравнение (1). Это импликация. Данное уравнение имеет 3 решения.

2. Предположим, что у нас есть система из первых двух уравнений. Если составить таблицу истинности, то будет 4 решения.

3. Для трех уравнений получим 5 решений.

4. Для 9 уравнений (10 переменных) будет 11 решений.

Способ 2.

1. Пусть $X1 = 0$, тогда из (1) $X2 = 0$, из (2) $X3 = 0$, …, из (9) $X10 = 0$. Получаем первый набор:

X1	*X2*	*X3*	*X4*	*X5*	*X6*	*X7*	*X8*	*X9*	*X10*
0	0	0	0	0	0	0	0	0	0

2. Пусть $X1 = 1$, тогда если $X2 = 0$, то из (2) $X3 = 0$, …, из (9) $X10 = 0$. Получаем второй набор:

X1	*X2*	*X3*	*X4*	*X5*	*X6*	*X7*	*X8*	*X9*	*X10*
1	0	0	0	0	0	0	0	0	0

3. Если $X2 = 1$, тогда из (3), если $X3 = 0$, то из (4) $X4 = 0$, …, из (9) $X10 = 0$. Получаем третий набор:

X1	*X2*	*X3*	*X4*	*X5*	*X6*	*X7*	*X8*	*X9*	*X10*
1	1	0	0	0	0	0	0	0	0

и т.д.

Если занести все решения в таблицу, то получим:

X1	*X2*	*X3*	*X4*	*X5*	*X6*	*X7*	*X8*	*X9*	*X10*
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1

Ответ: 11.

Задачи к разделу «Алгебра логики»

В задачах 1 – 6 символом *F* обозначено одно из указанных в задаче логических выражений от трех аргументов: *x*, *y*, *z*.

Дан фрагмент таблицы истинности выражения *F*. Какое выражение соответствует *F*?

1 :

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

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

2

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

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

3

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

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

4

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

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

5 :

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

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

6 :

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

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

7 Укажите, для какого из указанных значений X истинно высказывание $\neg(\neg(X > 15) \rightarrow (X > 16))$

А) 15 Б) 16 В) 17 Г) 18

8 Укажите, для какого из указанных значений X истинно высказывание $(X > 10) \rightarrow (50 > (X + 1))$.

А) 40 Б) 51 В) 52 Г) 50

9 Для какого из указанных значений X истинно высказывание $(X*(X-8) > -25+2*X) \rightarrow (X > 7)$?

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

10 Для какого из указанных значений X истинно высказывание $(X*(X-16) > -64) \rightarrow (X > 8)$?

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

11 Для какого из указанных значений X ложно высказывание $(X>7) \rightarrow (X$ – нечетное) ?

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

12 Для какого из указанных значений X ложно высказывание $(X$ – нечетное$) \rightarrow (X>7)$?

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

13 Для каких из указанных значений X и Y истинно высказывание: $(Y>X) \wedge (Y+X>0) \wedge (Y<1)$?

1) $X=0$, $Y=0$ 2) $X=0$, $Y=1/2$
3) $X=0$, $Y=1$ 4) $X=1/2$, $Y=1/2$

14 Для каких из указанных значений X и Y истинно высказывание: $(Y<X) \wedge (Y+X<0) \wedge (Y>-1)$?

1) $X=0$, $Y=0$ 2) $X=0$, $Y=-1/2$
3) $X=0$, $Y=-1$ 4) $X=1/2$, $Y=-1/2$

15 Для какого из указанных значений X ложно высказывание: $(X < 15) \wedge (X > 12) \rightarrow (X < 14)$?

16 Каково наибольшее целое число X, при котором ложно высказывание $((X-12)\cdot X+36 > 0) \rightarrow ((X+1)\cdot(X-1) > 39)$?

17 Каково наибольшее целое число X, при котором ложно высказывание $((X-10)\cdot X+25>0) \rightarrow (X\cdot X>30)$?

18 Каково наибольшее целое число X, при котором истинно высказывание $(X\cdot(X+1) > 99) \rightarrow (X\cdot X < 80)$?

19 Каково наибольшее целое число X, при котором истинно высказывание $(X\cdot(X+1) > 60) \rightarrow (X\cdot X < 50)$?

20 Каково наибольшее целое число X, при котором истинно высказывание $(90<X\cdot X) \rightarrow (X < (X - 1))$?

21 Каково наибольшее целое число X, при котором истинно высказывание $(50<X\cdot X) \rightarrow (50>(X+1)\cdot(X+1))$?

В задачах 22 – 48 преобразуйте логические выражения.

22 $(A \wedge \neg B) \vee (\neg A \wedge B) \wedge (A \vee B)$

23 $\neg (A \vee B) \wedge \neg(A \wedge \neg (B \vee \neg A))$

24 $(\neg A \wedge B) \vee (A \wedge B \wedge (A \vee \neg B))$

25 $\neg (\neg A \vee B) \wedge \neg (A \wedge \neg (B \vee \neg A))$

26 $(A \wedge B) \vee (\neg A \vee B) \wedge (A \vee \neg B)$

27 $\neg (A \vee \neg B) \wedge \neg (\neg A \wedge \neg (\neg B \vee A))$

28 $(A \vee \neg B) \wedge (\neg A \wedge B \vee A \wedge B)$

29 $\neg (A \wedge B) \wedge \neg(A \vee \neg (\neg B \vee A))$

30 $(A \vee B) \wedge ((A \wedge \neg B) \vee (\neg A \wedge B))$

31 $\neg (A \wedge \neg B) \wedge \neg (A \vee \neg (\neg B \wedge A))$

32 $(\neg A \wedge \neg B) \vee ((\neg A \wedge \neg B) \vee (A \wedge B))$

33 $\neg (A \wedge B) \wedge \neg (A \wedge \neg (B \wedge A))$

34 $(A \wedge \neg B) \vee \neg(\neg A \vee \neg(B \vee A))$

35 $(\neg A \vee B) \wedge (\neg A \wedge B) \wedge (\neg A \vee \neg B)$

36 $\neg (A \wedge \neg B) \wedge \neg (A \wedge B) \wedge (A \wedge \neg B)$

37 $(A \wedge \neg B) \vee (\neg A \wedge B) \vee (A \wedge B)$

38 $\neg (\neg A \vee B) \vee \neg (A \vee \neg (\neg B \wedge \neg A))$

39 $\neg (A \wedge B \wedge \neg C) \wedge (\neg A \vee \neg (B \wedge \neg C))$

40 $\neg (\neg A \wedge \neg B \wedge C) \wedge (A \vee \neg (\neg B \wedge \neg C))$

41 $\neg (A \vee B \vee \neg C) \wedge \neg (A \vee \neg (B \wedge C))$

42 $\neg (A \vee B \vee C) \vee \neg (\neg A \vee \neg (B \wedge \neg C))$

43 $(A \wedge \neg (B \wedge \neg C)) \wedge \neg (A \vee \neg(\neg B \vee C))$

44 $\neg (A \wedge B) \wedge \neg (C \wedge \neg A) \wedge \neg (B \wedge \neg C))$

45 $\neg (A \vee B) \vee \neg (C \vee \neg A) \vee \neg (\neg B \vee C))$

46 $\neg (A \vee B \vee \neg C) \vee \neg (\neg (A \wedge \neg B) \vee C)$

47 $X \wedge Y \wedge Z \vee \neg X \wedge Y \wedge Z$

48 $\neg (X \wedge \neg X) \vee Y \wedge (X \wedge Y \vee Y)$

49 Укажите, какое логическое выражение равносильно выражению $\neg(\neg A \wedge \neg B) \vee \neg C$.

1) $A \wedge B \wedge \neg C$ 2) $A \vee B \vee C$

3) $\neg(A \wedge B) \rightarrow \neg C$ 4) $A \vee B \vee \neg C$

50 Укажите, какое логическое выражение равносильно выражению $(\neg A \wedge \neg B) \vee \neg C$.

1) $(\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$ 2) $A \vee B \vee C$

3) $A \wedge B \vee \neg C$ 4) $\neg A \wedge \neg C \vee \neg B \wedge \neg C$

51 Укажите, какое логическое выражение равносильно выражению $(A \vee \neg B) \wedge \neg A$.

1) $(A \vee \neg B) \wedge A$ 2) $\neg A \wedge \neg B$

3) $\neg A \vee \neg B$ 4) $(\neg A \wedge B) \wedge \neg A$

52 Укажите, какое логическое выражение равносильно выражению $\neg(\neg A \vee \neg B) \wedge \neg C \wedge D$.

1) $A \wedge B \wedge \neg C \wedge D$ 2) $A \vee \neg B \wedge C \wedge D$

3) $\neg A \vee B \vee \neg C \wedge D$ 4) $\neg A \wedge \neg B \wedge C \wedge D$

53 Укажите, какое логическое выражение равносильно выражению $\neg A \vee A \wedge B$.

1) B 2) $\neg B$ 3) $\neg A \vee B$ 4) $\neg A \wedge B$

54 Укажите, какое логическое выражение равносильно выражению $\neg A \wedge B \vee A$.

1) B 2) $\neg B$ 3) $A \vee B$ 4) $A \wedge B$

55 Укажите, какое логическое выражение равносильно выражению $\neg(A \wedge \neg B) \wedge C$.

1) $\neg A \vee B \wedge C$ 2) $(\neg A \vee B) \wedge C$

3) $\neg A \vee B \wedge \neg C$ 4) $\neg A \wedge B \wedge C$

56 Укажите, какое логическое выражение равносильно выражению $\neg(\neg A \wedge \neg B) \vee \neg C$.

1) $A \wedge B \vee \neg C$ 2) $A \vee B \vee C$

3) $\neg A \wedge \neg B \vee C$ 4) $A \vee B \vee \neg C$

В задачах 57 – 63 по известной таблице истинности запишите логическую функцию и преобразуйте ее.

57

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

58

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

59

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

60

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

61

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

62

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

63

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

В задачах 64 – 66 по заданному логическому выражению постройте таблицу истинности.

64 $(B \wedge C) \vee (\neg B \wedge \neg C)$

65 $(A \wedge \neg B) \vee (\neg A \wedge C) \vee (\neg B \wedge C)$

66 $(\neg A \wedge B \wedge \neg C) \wedge (A \wedge B \wedge C) \wedge (A \wedge \neg B \wedge \neg C)$

В задачах 67 – 74 выберите правильный ответ.

67 Для какого слова ИСТИННО высказывание
(1 буква – гласная) ∧ ¬ (3 буква – согласная) ∨ (4 буква – гласная)?

1) abcde 2) bceda 3) abedc 4) bcade

68 Для какого символьного набора ЛОЖНО высказывание:
(Первая буква - гласная) → ((Вторая буква - согласная) ∧ (последняя буква - согласная))

1) Арбалет 2) Пробка 3) Кран 4) Арка

69 Для какого символьного набора ЛОЖНО высказывание:
((Первая буква - гласная) ∧ ¬(Вторая буква - гласная)) → (Третья буква - согласная)

1) Крот 2) Атака 3) Арбуз 4) Оазис

70 Какое из приведенных имен удовлетворяет логическому условию:
(первая буква согласная → вторая буква согласная) ∧ последняя буква гласная

1) ЕГОР 2) АЛЕНА 3) СТАНИСЛАВ 4) ТАТЬЯНА?

71 Для какого из названий животных ЛОЖНО высказывание:
Третья буква гласная → Заканчивается на гласную букву ∧ В слове 7 букв?

1) Леопард 2) Страус 3) Кенгуру 4) Верблюд

72 Для какого из названий животных ЛОЖНО высказывание:
Третья буква гласная → Заканчивается на гласную букву ∨ В слове 6 букв?

1) Пума 2) Леопард 3) Кенгуру 4) Страус

73 Какое из приведенных имен удовлетворяет логическому условию:
¬ (последняя буква гласная → первая буква согласная) ∧ вторая буква согласная

1) ИРИНА 2) АРТЕМ 3) СТЕПАН 4) МАРИЯ?

74 Какое из приведенных имен удовлетворяет логическому условию
¬ (первая буква гласная → вторая буква гласная) ∧ последняя буква гласная

1) ИРИНА 2) МАКСИМ 3)АРТЕМ 4) МАРИЯ?

В задачах 75 – 80 необходимо найти значения логических переменных *K*, *L*, *M* и *N*, удовлетворяющих указанному логическому уравнению. Ответ запишите в виде строки из четырех символов – значений переменных *K*, *L*, *M* и *N* (в указанном порядке). Так, например, строка 1001 соответствует тому, что *K*=1, *L*=0, *M*=0, *N*=1.

75 $(K \to \neg M) \vee (\neg L \wedge M \wedge K) \vee \neg N = 0$

76 $(\neg(M \vee L) \wedge K) \to (\neg K \wedge \neg M) \vee N) = 0$

77 $(\neg (\neg M \vee K) \wedge N) \wedge (M \to L) = 1$

78 $(K \to M) \vee (N \vee \neg L) \vee \neg (\neg M \wedge L) = 0$

79 $\neg K \vee K \wedge \neg L \wedge \neg M \vee K \wedge L \wedge \neg M \vee K \wedge L \wedge M = 0$

80 $(K \vee M) \to (M \vee \neg L \vee N) = 0$

В задачах 81 – 99 определите количество решений логического уравнения с четырьмя логическими переменными. В ответе не нужно перечислять все различные наборы значений, при которых выполнено данное равенство. В качестве ответа нужно указать КОЛИЧЕСТВО таких наборов.

81 $((K \vee L) \to (L \wedge M \wedge N)) = 0$

82 $J \wedge \neg K \wedge L \wedge \neg M \wedge (N \vee \neg N) = 0$

83 $\neg M \wedge K \wedge \neg N \wedge \neg J \wedge (L \vee \neg L) = 0$

84 $((A \vee \neg A) \wedge \neg B \wedge C \wedge D) \vee \neg B = 0$

85 $(K \to L \wedge M) \wedge (\neg N \vee N) = 0$

86 $(M \wedge \neg M) \vee (K \rightarrow \neg N) \vee (N \rightarrow L)=0$

87 $((A \rightarrow B) \wedge C) \vee (D \wedge \neg D) = 0$

88 $((A \rightarrow B) \wedge C) \vee (D \wedge \neg D) = 1$

89 $(M \rightarrow K) \wedge (\neg M \rightarrow \neg N) \wedge \neg(L \rightarrow M) = 1$

90 $(M \rightarrow K) \wedge (\neg M \rightarrow L) \wedge (K \rightarrow L) = 1$

91 $A \vee B \vee \neg C \vee (D \wedge \neg D) = 1$

92 $(A \wedge \neg A) \vee B \vee \neg C \vee (D \wedge \neg D) = 1$

93 $(K \rightarrow J) \wedge (M \wedge N \wedge L) \vee (K \wedge \neg J) \wedge \neg(M \wedge N \wedge L) = 1$

94 $(N \rightarrow (M \wedge K)) \wedge (\neg M \rightarrow L) \wedge (K \rightarrow L) \wedge (M \rightarrow K)= 1$

95 $(M \rightarrow L) \wedge (L \rightarrow K) \wedge (L \rightarrow \neg M) \wedge (K \rightarrow M) \wedge (O \rightarrow N) = 1$

96 $((J \rightarrow K) \rightarrow (M \wedge N \wedge L)) \wedge ((J \wedge \neg K) \rightarrow \neg(M \wedge N \wedge L)) \wedge (M \rightarrow J) = 1$

97 $((\neg N \rightarrow P) \rightarrow (K \wedge L \wedge M)) \wedge (\neg(\neg N \wedge \neg P) \rightarrow (\neg K \vee \neg L \vee \neg M)) = 1$

98 $((K \wedge L \wedge M) \rightarrow (\neg N \rightarrow P)) \wedge ((\neg K \vee \neg L \vee \neg M) \rightarrow (N \vee P)) = 1$

Логические задачи

Рассмотрим три типа логических задач. Для каждого типа будет приведено одно или несколько решений.

Задачи 1-го типа

В условии приводятся несколько двойных или одинарных утверждений и дается оценка их истинности, т.е. сообщается, сколько участников говорят только правду, сколько лгут и сколько говорят то правду, то ложь.

1.1 (*Демо-вариант 2009 г., задача В6*)
Классный руководитель пожаловался директору, что у него в классе появилась компания из трех учеников, один из которых всегда говорит

правду, другой всегда лжет, а третий говорит через раз то ложь, то правду. Директор знает, что их зовут Коля, Саша и Миша, но не знает, кто из них правдив, а кто – нет. Однажды все трое прогуляли урок астрономии. Директор знает, что никогда раньше никто из них не прогуливал астрономию. Он вызвал всех троих в кабинет и поговорил с мальчиками. Коля сказал: "Я всегда прогуливаю астрономию. Не верьте тому, что скажет Саша". Саша сказал: "Это был мой первый прогул этого предмета". Миша сказал: "Все, что говорит Коля, – правда". Директор понял, кто из них кто. Расположите первые буквы имен мальчиков в порядке: "говорит всегда правду", "всегда лжет", "иногда говорит правду, а иногда лжет". (Пример: если бы имена мальчиков были Рома, Толя и Вася, ответ мог бы быть: РТВ.)

Решение. Обозначим буквой И истинное (правдивое) утверждение и соответственно Л – ложное утверждение.

Запишем утверждения каждого мальчика:

Саша	И	утверждение Саши ИСТИННО, т. к. астрономию никто не прогуливал
Коля	Л Л	первое утверждение Коли ЛОЖЬ, т. к. астрономию никто не прогуливал; второе утверждение тоже ЛОЖЬ, так как Саша говорит правду
Миша	Л	утверждение, что Коля говорит правду, ЛОЖЬ

Теперь сразу видно, что Коля лжет всегда (он дважды подряд солгал),

Саша говорит правду (он единственный, кто сказал правду),

Получается, что Миша может сказать правду, а может и солгать.

Ответ: СКМ.

1.2 Петя, Вася и Маша остались дома одни. Кто-то из них съел варенье. На вопрос мамы, кто это сделал, они сказали:

A. Петя: "Я не ел. Маша тоже не ела".

B. Вася: "Маша действительно не ела. Это сделал Петя".

C. Маша: "Вася врет. Это он съел".

Выясните, кто ел варенье, если известно, что двое из них оба раза сказали правду, а третий один раз соврал и один раз сказал правду.

Решение. Выделим из условия задачи простые высказывания и обозначим их буквами

Петя ¬П ¬М
Вася ¬М П
Маша В Вася врет

Построим таблицу, в которой совместим высказывания детей и все возможные варианты их высказываний, когда двое говорят правду (И – ИСТИНА), а один то правду, то неправду (Л – ЛОЖЬ). В нашей задаче возможных вариантов три.

Условие задачи		**1**	**2**	**3**
Петя	¬П ¬М	Л	И	И
Вася	¬М П	И	Л	И
Маша	В Вася врет	И	И	Л

Проанализируем таблицу.

Первая колонка противоречит условию задачи, так как Маша утверждает, что Вася врет.

Третья колонка тоже противоречит условию задачи, так как Петя говорит, что он не виноват, а Вася, что он виноват.

Из второй колонки следует, что Маша сказала правду, и виноват Вася, который сказал правду про Машу и неправду про Петю.

Ответ: Варенье съел Вася.

Задачи 2-го типа

В условии приводятся несколько двойных утверждений, в которых одно утверждение истинно, а другое ложно. Результат – расстановка участников по местам.

2.1 (*Демо-вариант 2008 г., задача B4*)
Перед началом турнира болельщики (эксперты) высказали следующие предположения по поводу своих кумиров:

A. Макс победит, Билл – второй.
B. Билл – третий, Ник – первый.
C. Макс – последний, а первый Джон.

Когда соревнования закончились, оказалось, что каждый болельщик был прав только в одном из своих прогнозов. Какое место на турнире заняли

Джон, Билл, Ник, Макс? В ответе перечислите без пробелов места участников в указанном порядке имен.

Сначала проведем решение этой задачи методом преобразования логических выражений, затем применим метод рассуждений, потом решим эту же задачу, используя граф.

Решение методом преобразования логических выражений

Введем обозначения для логических утверждений:

A. *М1* – Макс победит *Б2* – Билл на втором месте.
B. *Б3* – Билл на 3-м месте *Н1* – Ник будет первым.
C. *М4* – Макс – последний *Д1* – первый – Джон.

Составим выражения для каждого эксперта

A. $(M1 \wedge \neg Б2) \vee (\neg M1 \wedge Б2)$.

Это значит, что если *М1* истинно, то *Б2* ложно, ИЛИ если *Б2* истинно, то *М1* ложно, так как каждый эксперт прав только в одном из своих прогнозов.

B. $(Б3 \wedge \neg Н1) \vee (\neg Б3 \wedge Н1)$.

C. $(М4 \wedge \neg Д1) \vee (\neg М4 \wedge Д1)$.

Решим совместно полученные логические выражения:

$(M1 \wedge \neg Б2 \vee \neg M1 \wedge Б2) \wedge (Б3 \wedge \neg Н1 \vee \neg Б3 \wedge Н1) \wedge (М4 \wedge \neg Д1 \vee \neg М4 \wedge Д1)$.

Сначала перемножим первую и третью скобку:

$(M1 \wedge \neg Б2 \wedge М4 \wedge \neg Д1) \vee (M1 \wedge \neg Б2 \wedge \neg М4 \wedge Д1) \vee (\neg M1 \wedge Б2 \wedge М4 \wedge \neg Д1) \vee$
$\vee (\neg M1 \wedge Б2 \wedge \neg М4 \wedge Д1)$.

1-е слагаемое равно 0, т. к. $M1 \wedge М4 = 0$ (Макс не может занимать одновременно первое и четвертое место).

2-е слагаемое равно 0, т. к. $M1 \wedge Д1 = 0$ (два участника не могут занимать одновременно первое место).

Оставляем только 3-е и 4-е слагаемые.

Производя подстановку упрощенного выражения в исходное, получаем:

$(\neg M1 \wedge Б2 \wedge М4 \wedge \neg Д1 \vee \neg M1 \wedge Б2 \wedge \neg М4 \wedge Д1) \wedge (Б3 \wedge \neg Н1 \vee \neg Б3 \wedge Н1)$.

Перемножаем:

$(\neg M1 \wedge Б2 \wedge М4 \wedge \neg Д1 \wedge Б3 \wedge \neg Н1) \vee (\neg M1 \wedge Б2 \wedge М4 \wedge \neg Д1 \wedge \neg Б3 \wedge Н1) \vee$
$\vee (\neg M1 \wedge Б2 \wedge \neg М4 \wedge Д1 \wedge Б3 \wedge \neg Н1) \vee (\neg M1 \wedge Б2 \wedge \neg М4 \wedge Д1 \wedge \neg Б3 \wedge Н1)$

Проанализируем полученные слагаемые.

1-е и 3-е слагаемые равны 0, так как $Б2 \wedge Б3 = 0$ (Билл не может занимать два места сразу).

4-е слагаемое равно 0, так как $Д1 \wedge Н1 = 0$ (два участника не могут занимать первое место).

2-е слагаемое равно 1, если $Н1 = 1$, $Б2 = 1$, $М4 = 1$; для Джона осталось 3-е место!

Ответ: 3214.

Решение методом рассуждений

Используем те же обозначения утверждений и высказывания экспертов.

Напомним, что каждый эксперт прав только в одном из своих прогнозов.

A. *М1* – Макс победит, *Б2* – Билл на втором месте.

B. *Б3* – Билл на 3-м месте, *Н1* – Ник будет первым.

C. *М4* – Макс последний, *Д1* – первый Джон.

A. $(М1 \wedge \neg Б2) \vee (\neg М1 \wedge Б2)$.

B. $(Б3 \wedge \neg Н1) \vee (\neg Б3 \wedge Н1)$.

C. $(М4 \wedge \neg Д1) \vee (\neg М4 \wedge Д1)$.

Предположим, что *М1* истинно, тогда *М4* ложно.

Из строки С: если *М4* ложно, то *Д1* истинно. Это противоречит тому, что *М1* истинно.

Если *М1* ложно, то *Б2* истинно, а *Б3* ложно.

Из строки В: *Н1* истинно, так как *Б3* ложно.

Поскольку *М1* ложно, то *М4* истинно и *Д1* ложно, а *Д3* истинно, так как все остальные места заняты.

Метод рассуждений можно дополнить таблицей, которая представляет собой компактное описание условия задачи и наглядно демонстрирует процесс задачи.

Заполним таблицу в соответствии с условием задачи.

Эксперт / Место	A	B	C
1	Макс	Ник	Джон
2	Билл		
3		Билл	
4			Макс

Вот как лаконично теперь выглядит условие задачи.

Будем зачеркивать ложные утверждения и подчеркивать истинные.

Предположим, что *Б2* – ИСТИННО, тогда *М1* – ЛОЖЬ, *Б3* – ЛОЖЬ, *Н1* – ИСТИННО, *Д1* – ЛОЖЬ, *М4* – ИСТИННО.

Тогда таблица примет вид:

Эксперт / Место	A	B	C
1	~~Макс~~	Ник	~~Джон~~
2	Билл		
3		Билл	
4			Макс

При заполнении таблицы надо иметь в виду, что значение ИСТИННО в клетке позволяет сделать вывод, что остальные клетки в этой строке и в этом столбце имеют значение ЛОЖНО.

Места 1, 2, 4 определились, Джону осталось 3-е место.

Теперь сделаем другое начальное предположение, при котором мы должны прийти к противоречию

Пусть *М1* — ЛОЖЬ, тогда *М4* – ЛОЖЬ, *Д1* – ИСТИННО. Получили противоречие, так как *М1* уже ИСТИННО.

Эксперт / Место	A	B	C
1	Макс	~~Ник~~	Джон
2	Билл		
3		Билл	
4			Макс

Тогда надо сделать обратное начальное предположение, а именно *М1* – ЛОЖЬ. Получаем, что *М4* – ИСТИННО, *Б2* – ИСТИННО, *Б3* – ЛОЖЬ, *Д1* – ЛОЖЬ, *Н1* – ИСТИННО.

Эксперт / Место	A	B	C
1	~~Макс~~	Ник	~~Джон~~
2	Билл		
3		Билл	
4			Макс

Места 1, 2, 4 определились, Джону осталось 3-е место

Всего надо сделать не более двух проверок.

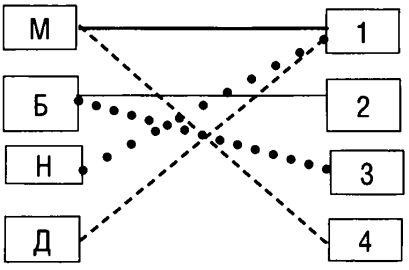
Конечно, заранее сказать, какое начальное предположение будет более удачным, нельзя.

Ответ: 3214.

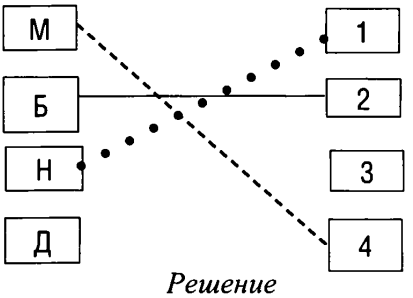
Решение с применением графа

Рассуждения можно облегчить, если нарисовать граф, наглядно показывающий связи между именами и местами, которые упоминаются в утверждениях экспертов.

Представим утверждения в виде графа.



Исходный граф



Решение

Для рассматриваемой задачи вершинами графа являются имена участников и места, которые они могут занять. Для каждого из экспертов используются линии разных типов. В результате решения на графе должна остаться только одна линия каждого типа, и из каждой вершины должна выходить только одна линия.

Предположим, что победил Макс, т. е. имеет место линия *М1*. Следовательно, *М4* – ЛОЖНО, и мы должны убрать пунктирную линию *М4*. Однако при этом в вершину 1 придут две линии: *М1* и *Д1*, т.е. *Д1* тоже ИСТИННО. Получилось противоречие. Значит, линию *М1* надо убрать, а линию *М4* оставить. Так как *М1* убрали, линия *Б2* остается.

Теперь перейдем к высказыванию второго эксперта. Поскольку линия *Б3* убирается, то в высказывании истинно, что Ник победил, т. е. *Н1* остается. Джон на 3-м месте.

Правильные утверждения отмечены на правом графе, представляющем собой решение задачи.

Ответ 3214.

Задачи 3-го типа

В условии приводятся несколько (обычно три) двойных утверждений, в которых одно утверждение истинно, а другое ложно.

3.1 Трое свидетелей так рассказали о машине, которую они видели:

1) Это была Хонда черного цвета.
2) Это был Форд синего цвета.
3) Это Мерседес, но не синий.

Каждый из них был прав только в одном из своих утверждений. Какая это была машина?

Решение методом рассуждений

Сначала предположим, что машина – Хонда, тогда утверждения свидетелей можно записать в виде:

1) 1 0 – одно утверждение истинно, а другое ложно, иными словами, если утверждение «машина – Хонда» истинно, то машина была не черная;
2) 0 1 – Хонда, значит, не Форд и синий;

0 0 – Мерседесом быть не может, но и не может быть черной машиной. Следовательно, оба высказывания третьего свидетеля ложны. Таким образом, мы пришли к противоречию, так как одно из высказываний свидетеля обязательно истинно.

Теперь предположим, что это Форд.

1) 0 1 – не Хонда, значит, истинно высказывание, что машина черная;
2) 1 0 – если Форд, то не синий;
3) 0 1 – не Мерседес и не синий.

Следовательно, это черный Форд.

Для самопроверки сделаем еще одно предположение, что машина – Мерседес. Утверждения свидетелей запишутся в виде:

1) 0 1 – не Хонда, значит, машина черная;
2) 0 0 – при анализе высказывания второго свидетеля сразу получили противоречие. На этом решение можно закончить и принять полученный ранее ответ.

Ответ: черный Форд.

Решение методом преобразования логических выражений

Введем обозначения для логических утверждений

1) X – марка машины Хонда, $Ч$ – цвет машины черный.
2) $Ф$ – марка машины Форд, C – цвет машины синий.
3) M – марка машины Мерседес, $\neg C$ – цвет машины не синий.

Составим логические выражения для каждого свидетеля:

1) $(X \wedge \neg Ч) \vee (\neg X \wedge Ч)$;
2) $(Ф \wedge \neg C) \vee (\neg Ф \wedge C)$;
3) $(M \wedge C) \vee (\neg M \wedge \neg C)$.

Решим совместно полученные логические выражения:

$$((X \wedge \neg Ч) \vee (\neg X \wedge Ч)) \wedge (Ф \wedge \neg C \vee \neg Ф \wedge C) \wedge (M \wedge C \vee \neg M \wedge \neg C).$$

Перемножим 2-й и 3-й сомножители:

$(Ф \wedge \neg С \wedge М \wedge С) \vee (Ф \wedge \neg С \wedge \neg М \wedge \neg С) \vee (\neg Ф \wedge С \wedge М \wedge С) \vee$

$\vee (\neg Ф \wedge С \wedge \neg М \wedge \neg С)$.

1-е слагаемое равно 0, так как $С \wedge \neg С = 0$.
4-е слагаемое равно 0, так как $С \wedge \neg С = 0$.
Остались 2-е и 3-е слагаемые, которые надо умножить на первый сомножитель:

$(Ф \wedge \neg С \wedge \neg М \vee \neg Ф \wedge М \wedge С) \wedge (Х \wedge \neg Ч \vee \neg Х \wedge Ч) =$

$= (Ф \wedge \neg С \wedge \neg М \wedge Х \wedge \neg Ч) \vee (Ф \wedge \neg С \wedge \neg М \wedge \neg Х \wedge Ч) \vee$

$\vee (\neg Ф \wedge М \wedge С \wedge Х \wedge \neg Ч) \vee (\neg Ф \wedge М \wedge С \wedge \neg Х \wedge Ч)$.

1-е слагаемое равно 0, так как $Ф \wedge Х = 0$.
3-е слагаемое равно 0, так как $М \wedge Х = 0$.
4-е слагаемое равно 0, так как $М \wedge Ч = 0$.
В соответствии со 2-м слагаемым это был черный Форд.

Решение табличным способом

Составим таблицу истинности. В первой строке перечислим логические переменные и формулы, как в предыдущем решении.

Х	Ф	М	Ч	С	$(\underbrace{Х \wedge \neg Ч}) \vee (\underbrace{\neg Х \wedge Ч}) \wedge (\underbrace{Ф \wedge \neg С} \vee \underbrace{\neg Ф \wedge С}) \wedge (\underbrace{М \wedge С} \vee \underbrace{\neg М \wedge \neg С})$						*F*
		1		1	0	0					0
	1			1	0	0					0
1				1	1	0		1	0	0	0
		1	1		0	1	0	0			0
	1		1		0	1	1	0	0	1	1
1			1		0	0					0

Поскольку переменных 5, таблица истинности должна была бы содержать 32 строки. Однако одна машина не может быть двух марок, поэтому переменные *Х*, *Ф* и *М* не могут одновременно принимать значение 1. Цвет машины не может быть и синим, и черным, следовательно, переменные *С* и *Ч* не могут одновременно принимать значение 1. Это рассуждение позволяет нам сократить таблицу до шести строк и заполнить ее всеми возможными сочетаниями значений переменных. Для удобства чтения в первых пяти столбцах мы не будем прописывать значение 0.

В конечном столбце записывается результат вычисления формулы. Рассмотрим, как вычисляется этот результат.

Заполним столбцы, относящиеся к формуле. Под каждым выделенным

элементом формулы запишем значения, которые он принимает в зависимости от значений входящих в него переменных. Так как формула представляет собой произведение трех выражений, если один из сомножителей равен 0, а дальше заполнение можно не продолжать.

После заполнения таблицы видно, что только на одном сочетании исходных данных функция принимает значение 1, это и есть решение задачи. Как и ожидалось, это будет черный форд.

Советы по решению логических задач

Для решения логических задач мы рекомендуем соблюдать следующие правила.

- Выделить из условия задачи простые высказывания и обозначить их буквами. Переход к записи условия задачи в виде логических переменных позволяет представить условие задачи в более компактном виде, отбросив все ненужные для решения задачи детали.
- Записать условие задачи на языке алгебры логики, соединив простые высказывания в сложные с помощью логических операций или построить таблицу, помогающую более наглядно представить условие задачи и использовать метод рассуждений.
- Попытаться упростить полученное выражение.
- Очень важно проверить соответствие полученного решения условию задачи.

Задачи к разделу «Логические задачи»

1 (*Демо-вариант 2008 г., задача B4*)
Классный руководитель пожаловался директору, что у него в классе появилась компания из трех учеников, один из которых всегда говорит правду, другой всегда лжет, а третий говорит через раз то ложь, то правду. Директор знает, что их зовут Коля, Саша и Миша, но не знает, кто из них правдив, а кто – нет. Однажды все трое прогуляли астрономию. Директор знает, что никогда раньше никто из них не прогуливал астрономию. Он вызвал всех троих в кабинет и поговорил с мальчиками. Коля сказал: “Я раньше никогда не прогуливал астрономию”. Саша сказал: “Бывает, что я говорю неправду”. Миша сказал: “Коля соврал Вам. Саша никогда не врет”. Директор понял, кто из них кто. Расположите первые буквы имен мальчиков в порядке: “говорит всегда правду”, “всегда лжет”, “говорит правду через раз”.

2 (*Демо-вариант 2007 г., задача B4*)
В школьном первенстве по настольному теннису в четверку лучших вошли девушки Наташа, Маша, Люда, Рита. Болельщики высказали свои предположения о распределении мест:

A. Первой будет Наташа, Маша будет второй.
B. Вторая будет Люда, а Рита займет четвертое место.
C. Рита займет третье место, а Наташа будет второй.

Каждый болельщик был прав только в одном из своих прогнозов
Какое место на турнире заняли Наташа, Маша, Люда, Рита?
В ответе перечислите без пробелов места участников в указанном порядке имен.

3 В полуфинал чемпионата по футболу вышли Англия, Германия, Бразилия и Франция. Эксперты стали высказывать свои предположения о результатах чемпионата.
Один эксперт считает, что на первом месте будет Англия, а на втором Германия.
Второй эксперт утверждает, что на втором месте будет Бразилия, а на четвертом Франция.
Третий эксперт сказал, что на втором месте Англия, а на третьем Франция.
После чемпионата выяснилось, каждый из экспертов был прав, только в одном своем утверждении. Какое место на чемпионате заняли Англия, Германия, Бразилия и Франция? В ответе перечислите без пробелов места участников в указанном порядке стран.

4 В соревнованиях по гимнастике участвовало пять девушек: Света, Маша, Наташа, Рита, Вера. Об итогах соревнований имеется пять высказываний, про которые известно, что в каждом из них одно утверждение истинно, а другое ложно:

1) Первое место заняла Вера, а Наташа была второй.
2) Первое место заняла Света, а Рита оказалась третьей.
3) Рита на последнем месте, а Маша была предпоследней.
4) Маша была действительно четвертой, а первой – Света.
5) Пятой была Наташа, а Маша на первом месте.

Какие места в соревновании заняли участницы? Напишите в ответе первые буквы имен участниц в порядке занятых ими мест (например, МНРСВ).

5 Андрей, Виктор, Миша и Николай приехали учиться в Москву из Киева, Ростова Саратова и Тамбова (имена и города приведены по алфавиту). Друзья про них высказали следующие утверждения:

1) Андрей приехал из Киева, Миша из Ростова.

2) В Ростове живет Андрей, Виктор приехал из Тамбова .
3) Николай приехал из Ростова, а Виктор живет в Саратове.

Определите, кто из какого города приехал, если стало известно, что половина каждого утверждения истинна, а половина – ложна?
Используя первые буквы имен и городов, напишите, где живут Андрей, Виктор, Миша и Николай (например, АТВКНСМР).

Совет: используйте метод рассуждений с таблицей.

6 При раскопках был найден древний кувшин. Один из экспертов утверждает, что это кувшин сделан в городе Владимире в X в., другой считает, кувшин изготовлен в Твери в XV в., третий утверждает, что кувшин сделан в Суздале, но не в X в. Как показала последующая экспертиза, каждый из экспертов был прав только в одном из своих утверждений. Когда и где был изготовлен кувшин?

7 В состав экспедиции входят Виталий, Петр и Сергей. На обсуждении распределения обязанностей с руководством проекта были высказаны предположения, что командиром будет назначен Виталий, Петр не будет механиком, а Сергей будет утвержден радистом, но командиром не будет. Позже выяснилось, что только одно из этих четырех утверждений оказалось верным. Перечислите, кто занял должности командира, механика, радиста, записав подряд без запятых (в указанном порядке) первые буквы соответствующих имен членов экипажа.

8 На балу четыре юноши – Сергей, Андрей, Михаил и Борис – танцевали с четырьмя девушками – Еленой, Натальей, Ольгой и Татьяной. Очевидцы сообщили следующее:

1) Наталья танцевала с Сергеем, а Татьяна с Андреем.
2) Ольга танцевала с Михаилом, а Елена с Андреем.
3) Ольга танцевала с Борисом, а Наталья с Андреем.

Известно, что в каждом из трех сообщений одно утверждение истинно, а другое ложно. В ответе запишите первую букву девушки, танцевавшей с Борисом и, через запятую, первую букву юноши, танцевавшем с Натальей.

9 В состав экспедиции входят Родион, Антон и Виктор. На обсуждении распределения обязанностей с руководством проекта были высказаны предположения, что командиром будет назначен Родион, Антон не будет механиком, а Виктор будет утвержден радистом, но командиром не будет.
Позже выяснилось, что только одно из этих четырех утверждений оказалось верным. Перечислите, кто занял должности командира, механика,

радиста, записав подряд без запятых (в указанном порядке) первые буквы соответствующих имен членов экипажа.

10 В состав экипажа входят Павел, Леонид и Егор. На обсуждении распределения обязанностей с руководством колонны были высказаны предположения, что командиром будет назначен Павел, Леонид не будет техником, а Егор будет утвержден штурманом, но командиром не будет.
Позже выяснилось, что только одно из этих четырех утверждений оказалось верным. Перечислите, кто занял должности командира, штурмана, техника, записав подряд без запятых (в указанном порядке) первые буквы соответствующих имен членов экипажа.

11 На перемене ученики уронили с подоконника цветок. Учитель опросил всех учеников, находившихся в классе, и получил следующие утверждения:

Толя – Мы с Ваней не входили в класс на перемене.
Игорь – Это кто-то из девчонок.
Ваня – Это не я и не Петя.
Оля – Это Петя.
Петя – Это Оля.
Лена – Цветок уронил кто-то из мальчишек.

Если бы учительница знала, что двое соврали, а четверо сказали правду, она бы без труда вычислила виновника. Кто уронил цветок? В ответе укажите первую букву имени.

12 На перемене ученики выкинули портфель из окна кабинета математики. Учитель опросил всех учеников, находившихся в классе, и получил следующие утверждения:

Толя – Мы с Мишей были на перемене в спортзале.
Илья – Это точно не Миша.
Ваня – Я не брал портфель.
Миша – Это не я и не Толя.
Коля – Это не Денис.
Денис – Это или Толя или Миша.
Сергей – Мы с Колей были в столовой.

Если бы учительница знала, что трое соврали, а четверо сказали правду, она бы без труда вычислила виновника. Кто выкинул портфель в окно? В ответе укажите первую букву имени.

13 На перемене ученики сломали парту. Учитель опросил всех учеников, находившихся в классе, и получил следующие утверждения:

Оля – Я не ломала парту.
Ира – Это Лена, я все видела.

Лена – Ира говорит неправду.
Катя – Сломали мальчики.
Вася – Оля не ломала парту.
Сережа – Это был либо Миша, либо Оля.
Миша – Лена никогда не врет.
Дима – Это девочки сломали.

Если бы учительница знала, что только 6 человек сказали правду, она бы без труда вычислила виновника. Кто сломал парту? В ответе укажите первую букву имени.

14 На одной улице стоят в ряд 4 дома, в которых живут 4 человека: Алексей, Егор, Виктор и Михаил. Известно, что каждый из них владеет ровно одной из следующих профессий: Токарь, Столяр, Хирург и Окулист, но неизвестно, кто какой и неизвестно, кто в каком доме живет. Однако известно, что:

1) Токарь живет левее Столяра.
2) Хирург живет правее Окулиста.
3) Окулист живет рядом со Столяром.
4) Токарь живет не рядом со Столяром.
5) Виктор живет правее Окулиста.
6) Михаил не Токарь.
7) Егор живет рядом со Столяром.
8) Виктор живет левее Егора.

Выясните, кто какой профессии и кто где живет, и дайте ответ в виде заглавных букв имени людей, в порядке слева направо. Например, если бы в домах жили (слева направо) Константин, Николай, Роман и Олег, ответ был бы: КНРО

15 На одной улице стоят в ряд 4 дома, в которых живут 4 человека: Анна, Виктория, Галина и Елена. Известно, что каждая из них владеет ровно одной из следующих профессий: Учитель, Врач, Секретарь и Менеджер, но неизвестно, кто какой и неизвестно, кто в каком доме живет. Однако известно, что:

1) Врач живет с краю.
2) Учитель живет правее Секретаря.
3) Менеджер живет правее Секретаря.
4) Врач живет рядом с Менеджером.
5) Галина живет рядом с Менеджером.
6) Виктория живет правее Учителя.
7) Елена не Секретарь и не Учитель.
8) Галина живет через дом от Виктории.

Выясните, кто какой профессии, и кто где живет, и дайте ответ в виде заглавных букв имени людей, в порядке слева направо. Например, если бы

в домах жили (слева направо) Константин, Николай, Роман и Олег, ответ был бы: КНРО.

16 На одной улице стоят в ряд 4 дома, в которых живут 4 человека: Анна, Виктория, Галина и Елена. Известно, что каждая из них владеет ровно одной из следующих профессий: Учитель, Врач, Секретарь и Менеджер, но неизвестно, кто какой и неизвестно, кто в каком доме живет. Однако известно, что:

1) Учитель живет с краю.
2) Менеджер живет левее Врача.
3) Секретарь живет правее Менеджера.
4) Учитель живет рядом с Врачом.
5) Анна живет рядом с Врачом.
6) Виктория живет правее Секретаря.
7) Елена не Менеджер и не Учитель.
8) Анна живет через дом от Виктории.

Выясните, кто какой профессии и кто где живет, и дайте ответ в виде заглавных букв имени людей, в порядке слева направо. Например, если бы в домах жили (слева направо) Константин, Николай, Роман и Олег, ответ был бы: КНРО.

17 На этапе кубка мира по биатлону была проведена эстафета, претендентами на победу были 5 команд: Россия, Швеция, Германия, Франция и Беларусь. До начала соревнований эксперты высказали свои предположения об итогах:

1) Первое место, конечно, займет Россия, а Швеция будет третьей.
2) Французы будут пятыми, а победит Беларусь.
3) Шведы придут последними, а Белоруссия финиширует четвертой.
4) Россия будет первой, а Франция – на втором месте.
5) Белорусы будут четвертыми, а первое место займет Германия.

После подведения итогов выяснилось, что в каждом предположении часть утверждения – истинна, а часть – ложна. В каком порядке финишировали команды? В ответе перечислите первые буквы названий стран без пробелов и запятых (начиная с первого места и заканчивая пятым).

18 На этапе кубка мира по биатлону была проведена эстафета, претендентами на победу были 5 команд: Россия, Германия, Франция, Норвегия, и Австрия. До начала соревнований эксперты высказали свои предположения об итогах:

1) Норвежцы будет пятыми, а Франция займет 2 место.
2) Германия будет третьей, а команда Австрии займет четвертое место.
3) Первое место займет Австрия, а Германия будет четвертой.
4) Австрия придет второй, а Франция – четвертой.

После подведения итогов выяснилось, что в каждом предположении часть утверждения – истинна, а часть – ложна. В каком порядке финишировали команды? В ответе перечислите первые буквы названий стран без пробелов и запятых (начиная с первого места и заканчивая пятым).

19 Пропал журнал 8 класса. Он был украден из кабинета русского языка одним из трех учеников, оставшихся на дополнительные занятия. Учитель опросил учеников и получил следующие ответы:

Игорь: 1) Я не видел журнал в кабинете, поэтому не мог его взять.
2) Олег невиновен.
3) Это Сергей.

Сергей: 1) Я этого не делал.
2) Это был Олег.
3) Первое высказывание Игоря справедливо.

Олег: 1) Я не брал журнал.
2) Это сделал Игорь, а если не Игорь, то Сергей.
3) Нет, это точно не Сергей.

Один ученик дал три правдивых ответа, другой дал одно правдивое утверждение, а третий два раза сказал правду.

Кто украл журнал? В ответе укажите первую букву имени вора, значение 2 высказывания Сергея (И или Л) и первую букву имени правдивого ученика. (Например, если вор Олег, второе высказывание Сергея ложно и правду сказал Сергей, то ответ нужно записать ОЛС)

20 Пропал журнал 8 класса. Он был украден из кабинета русского языка одним из трех учеников, оставшимся на дополнительные занятия. Учитель опросил учеников и получил следующие ответы:

Игорь: 1) Я не видел журнал в кабинете, поэтому не мог его взять.
2) Олег невиновен.
3) Это Сергей.

Сергей: 1) Я этого не делал.
2) Олег тоже невиновен.
3) Первое высказывание Игоря справедливо.

Олег: 1) Я не брал журнал.
2) Это сделал Игорь, а если не Игорь, то Сергей.
3) Нет, это точно не Сергей.

Один ученик дал три правдивых ответа, другой дал одно правдивое утверждение, а третий два раза сказал правду.

Кто украл журнал? В ответе укажите первую букву имени вора, значение 2 высказывания Олега (И или Л) и первую букву имени правдивого ученика. (Например, если вор Олег, второе высказывание Сергея ложно и правду сказал Сергей, то ответ нужно записать ОЛС).

21 Пять человек (Артур, Максим, Настя, Олег и Рита) убирались в кабинете. Когда учитель их спросила, кто догадался протереть подоконники, ученики ответили следующее:

Максим: «Ни я, ни Олег подоконники не мыли».

Артур: «Их помыли Максим или Настя».

Рита: «Один из ребят сказал правду, а другой обманул».

Олег: «Нет, Рита, ты не права».

Настя: «Это был Олег».

Учитель знает, что трое учеников всегда говорят правду, а двое лгут. Кто протер подоконники (в ответе укажите имя ученика)?

22 На международных соревнованиях по прыжкам в воду первые пять мест заняли спортсмены из Германии, Италии, Китая, России и Украины. Эксперты высказали свои предположения об итогах соревнований:

1) Первое место займет спортсмен из Китая, а спортсмен из Украины будет третьим.
2) Украина будет на последнем месте, а Германия – на предпоследнем.
3) Германия точно будет четвертой, а первое место займет Китай.
4) Россия будет первой, а спортсмен из Италии – на втором месте.
5) Нет, спортсмен из Италии будет пятым, а победит Германия.

По окончании соревнований выяснилось, что каждый эксперт был прав только в одном утверждении. Какие места в соревновании заняли участники? В ответе перечислите первые буквы стран (без пробелов и знаков препинания) в порядке занятых спортсменами мест начиная с первого.

23 В сборе макулатуры участвовало 5 классов 1а, 2а, 2б, 3а и 4а. Было высказано 4 предположения об итогах:

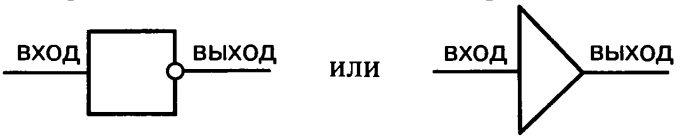
1) Первое место займет 2а, а 2б будет четвертым.
2) 2б будет третьим, а 2а – четвертым.
3) 2а займет второе место, а 3а – четвертое.
4) 4а будет пятым, а 3а – вторым.

После подведения итогов выяснилось, что в каждом предположении часть утверждения – истинна, а часть – ложна. Какие места в соревновании заняли участники? В ответе перечислите классы (без пробелов и знаков препинания) в порядке занятых ими мест, начиная с первого (например, 4а2б1а3а2а).

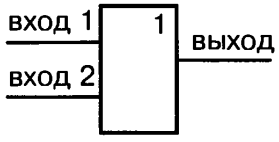
Логические схемы

В цифровой технике существуют специальные логические схемы, реализующие три базовые логические операции: инверсию, конъюнкцию и дизъюнкцию.

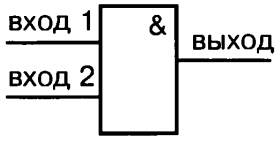
1. *Инвертор.* Реализует операцию отрицания.



2. *Элемент 2–ИЛИ.* Реализует операцию дизъюнкции (логического сложения). Цифра 2 обозначает количество входов элемента. Их может быть более двух.



3. *Элемент 2–И.* Реализует операцию конъюнкции (логического умножения). Цифра 2 обозначает количество входов элемента. Их может быть более двух.



1 Построить логическую схему для функции $F = \neg A \wedge B \vee C$.

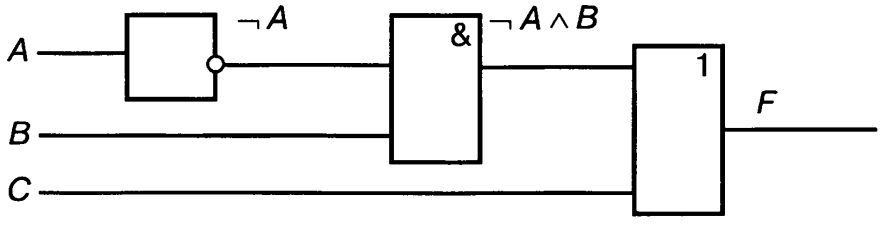
Решение. В соответствии с приоритетами выполнения логических операций логическую схему строим в следующем порядке:

1) инвертируем значение параметра A;

2) при помощи элемента 2–И умножаем $\neg A$ на B;

3) выход элемента 2–И соединяем со входом элемента 2–ИЛИ. На другой вход этого элемента подаем сигнал C.

Логическая схема будет иметь вид:



2 Построить логическую схему функции $F = (\neg(A \vee \neg B) \vee C) \wedge (B \vee \neg C)$.

Решение. В соответствии с приоритетами выполнения логических операций логическую схему строим в следующем порядке:

1) инвертируем значение параметров B и C;

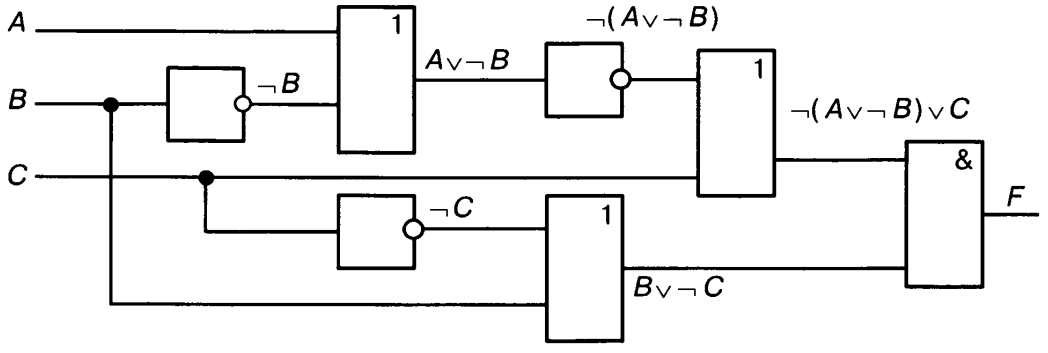
2) при помощи 1-го элемента 2–ИЛИ складываем A и $\neg B$. Далее инвертируем получившийся на выходе сигнал;

3) выход 1-го элемента 2–ИЛИ после инверсии соединяем со входом 2-го элемента 2–ИЛИ. На другой вход этого элемента подаем сигнал C;

4) отдельно при помощи 3-го элемента 2–ИЛИ складываем B и $\neg C$;

5) соединяем выходы 2-го и 3-го элементов 2–ИЛИ со входами элемента 2–И.

Логическая схема будет иметь вид:



Обратите внимание на то, что на вход логической схемы подаются все сигналы (A, B, C), участвующие в логической функции, в неинвертированном виде. Количество входов логической схемы равно количеству переменных, участвующих в логической функции, а выход всегда один.

3 Построить логическую схему функции $F = A \vee B \wedge A$. Вычислить значение при A=0, B=1.

4 Построить логическую схему функции $F = A \wedge B \vee (A \vee B)$. Вычислить значение при A=1, B=0.

5 Построить логическую схему функции $F = A \vee B \wedge C$.

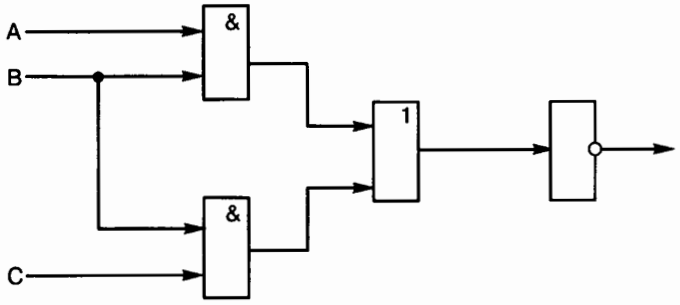
6 Построить логическую схему функции $F = \neg(A \vee B \wedge C)$.

7 Построить логическую схему функции $F = A \vee B \vee C$.

8 Построить логическую схему функции $F = (A \vee B) \wedge (B \vee C)$.

9 Построить логическую схему функции $F = A \wedge B \wedge C$.

10 Составить логическую функцию по логической схеме:



Решение. Построение функции начинаем с конца.

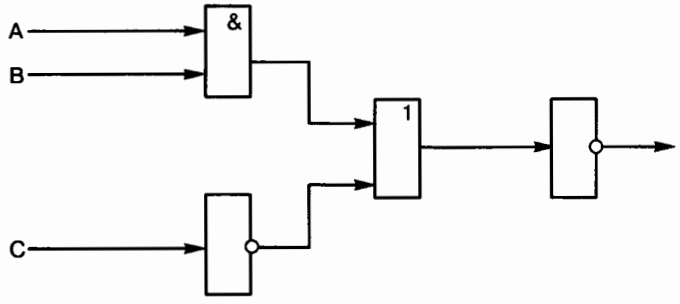
1. Последним в схеме является инвертор, который реализует операцию отрицания: $\neg X$. На месте X стоит какая-то формула.

2. Перед инвертором расположен элемент 2-ИЛИ, имеющий два входа, то есть он реализует операцию логического сложения, зависящую от двух аргументов: $\neg(Y \vee Z)$. На месте Y и Z стоят какие-то формулы.

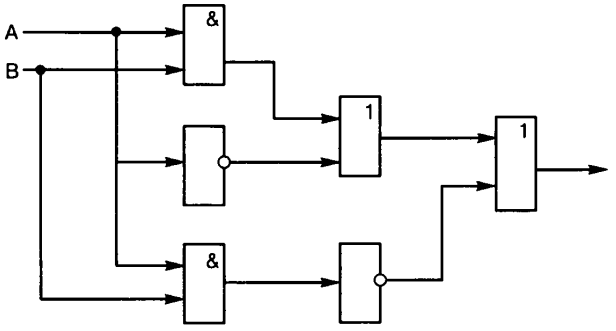
3. Y – результат операции логического умножения величин A и B: $A \wedge B$. Аналогично, Z – результат операции логического умножения величин B и C: $B \wedge C$. Делаем подстановку и получаем результат.

Ответ. $\neg(A \wedge B \vee B \wedge C)$.

11 Составить логическую функцию по логической схеме:

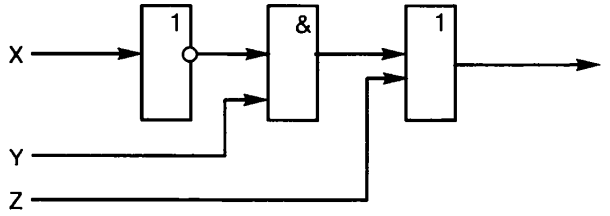


12 Составить логическую функцию по логической схеме:

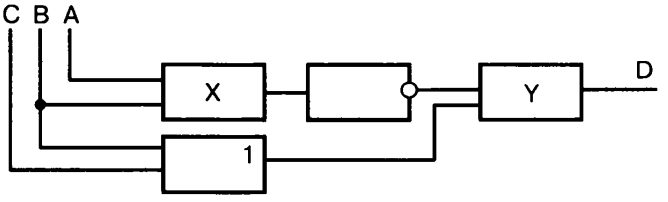


13 Какое логическое выражение, соответствует приведенной логической схеме:

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



14 (*Многопрофильная олимпиада ГУ ВШЭ, 2 тур, Информатика 2009-2010*) Логическая схема содержит два неопределенных элемента *X* и *Y*. Было проведено тестирование логической схемы, заключающееся в подаче на входы *A*, *B*, *C* всех возможных комбинаций нулевых и единичных сигналов. Выяснилось, что при любых комбинациях на входах сигнал на выходе *D* всегда является единичным. Какими элементами являются *X* и *Y*: *инвертор*, *2-ИЛИ* или *2-И*?



Глава 3

Алгоритмизация и программирование

Исполнители алгоритмов

Калькулятор

1 У исполнителя Калькулятор две команды, которым присвоены номера:

1) вычти 1;

2) умножь на 3.

Первая из них уменьшает число на экране на 1, вторая – увеличивает его в 3 раза. Запишите порядок команд в программе получения из 4 числа 17, содержащей не более 5 команд, указывая лишь номера команд.

Например, 12211 – это программа:

вычти 1;
умножь на 3;
умножь на 3;
вычти 1;
вычти 1,

которая преобразует число 2 в 7.

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

Решение: При выполнении задания поиск программы удобнее вести от ответа, приближаясь к исходному числу. Ближайшее делящееся на 3 число, из которого вычитанием единиц можно получить 17, будет 18.

Следовательно, последняя команда в программе будет 1. Число 18 можно получить, умножая 6 на 3, а число 6 – умножая 2 на 3.

Получаем, что три последние команды в программе – 221. Ну, а 2 получается, если дважды вычесть 1 из исходного числа 4. Тогда две первые команды в программе – 11. В целом программа будет выглядеть так: 11221. Программа содержит 5 команд, значит условие задачи выполнено.

Ответ: 11221.

2 У исполнителя Калькулятор две команды, которым присвоены номера:

1) возведи в квадрат;

2) вычти 1.

Первая из них возводит число на экране в квадрат, вторая уменьшает его на 1. Запишите порядок команд в программе получения из числа 5 числа 8, содержащей не более 4 команд, указывая лишь номера команд.
Например, программа 12122 – это программа:

возведи в квадрат;
вычти 1;
возведи в квадрат;
вычти 1;
вычти 1,

которая преобразует число 2 в 7.

3 У исполнителя Калькулятор две команды, которым присвоены номера:

1) прибавь 3;
2) умножь на 2.

Выполняя первую из них, Калькулятор прибавляет к числу на экране 3, а выполняя вторую, удваивает его. Запишите порядок команд в программе получения из 2 числа 31, содержащей не более 6 команд, указывая лишь номера команд.
Например, программа 12211 – это программа:

прибавь 3;
умножь на 2;
умножь на 2;
прибавь 3;
прибавь 3,

которая преобразует число 0 в 18.

4 У исполнителя Калькулятор две команды, которым присвоены номера:

1) вычти 2;
2) умножь на 3.

Первая из них уменьшает число на экране на 2, вторая – утраивает его. Запишите порядок команд в программе получения из 5 числа 19, содержащей не более 5 команд, указывая лишь номера команд.
Например, 21211 – это программа:

умножь на 3;
вычти 2;
умножь на 3;
вычти 2;
вычти 2,

которая преобразует число 3 в 17.
Если таких программ более одной, то запишите любую из них.

5 У исполнителя Калькулятор две команды, которым присвоены номера:

1) прибавь 2;
2) умножь на 3.

Выполняя первую из них, Калькулятор прибавляет к числу на экране 2, а выполняя вторую, утраивает его. Запишите порядок команд в программе получения из 1 числа 31, содержащей не более 5 команд, указывая лишь номера команд.

Например, программа 12211 – это программа:

прибавь 2;
умножь на 3;
умножь на 3;
прибавь 2;
прибавь 2,

которая преобразует число 0 в 22.

6 (*Демо-вариант 2011 г.*)

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: Обозначим $R(n)$ – количество программ, которые преобразуют число 1 в число n. Обе команды исполнителя увеличивают исходное число, поэтому общее число команд в программе не может превосходить 28. $R(1) = 1$.

Для каждого следующего числа рассмотрим, сколькими способами это число может быть получено из предыдущих чисел за одну команду исполнителя. Если число не делится на три, то последней командой может быть только «Прибавь 1». Следовательно, количество программ для этого числа равно количеству программ для предыдущего числа:

$R(i) = R(i\text{-}1)$, если i не делится на 3.

Если число делится на 3, то вариантов последней команды два: «Прибавь 1» или «Умножь на 3». Следовательно, $R(i) = R(i\text{-}1) + R(i/3)$, если i делится на 3.

Ответ: Для числа 29 количество программ можно подсчитать, составив таблицу:

Число	1	2	3	4	5	6
Количество команд исполнителя	1	1	1+1=2	2	2	2+1=3
Число	7	8	9	10	11	12
Количество команд исполнителя	3	3	3+2=5	5	5	5+2=7
Число	13	14	15	16	17	18
Количество команд исполнителя	7	7	7+2=9	9	9	9+3=12
Число	19	20	21	22	23	24
Количество команд исполнителя	12	12	12+3=15	15	15	15+3=18
Число	25	26	27	28	29	
Количество команд исполнителя	18	18	18+5=23	23	23	

7 (*Демо-вариант 2011 г., задача С3*)

У исполнителя Удвоитель две команды, которым присвоены номера:

1) прибавь 1;
2) умножь на 2.

Первая из них увеличивает число на экране на 1, вторая – удваивает его. Программа для Удвоителя – это последовательность команд. Сколько есть программ, которые число 1 преобразуют в число 16? Ответ обоснуйте.

Черепашка

8 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*, где *n* – целое число, вызывающая передвижение черепашки на *n* шагов в направлении движения;

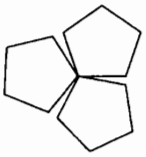
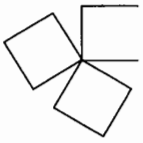
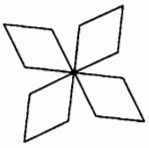
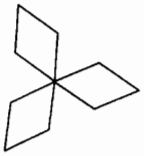
Направо *m*, где *m* – целое число, вызывающая изменение направления движения на *m* градусов по часовой стрелке.

Запись **Повтори 5 [Команда1 Команда2]** означает, что последовательность команд в скобках повторится 5 раз. Исполнитель интерпретирует эту запись как одну команду.

Черепашке был дан для исполнения следующий алгоритм:

Повтори 5 [Повтори 2 [Вперед 40 Направо 60 Вперед 40 Направо 120] Направо 90]

Какая фигура появится на экране?

1)  2)  3)  4) 

Решение: Исполняя внутренний цикл, черепашка рисует ромб, т. к. за один шаг она рисует две линии одинаковой длины и поворачивается на 180 градусов, а за 2 шага совершает полный оборот вокруг своей оси. Внешний цикл заставляет черепашку 5 раз нарисовать ромб, но при этом за 4 шага она поворачивается на 360 градусов. Следовательно, будет нарисована фигура, представленная на рис. 3, так как первый и последний ромб совпадут.

Ответ: 3.

9 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 надо записать вместо *n* в следующем алгоритме, чтобы на экране появился правильный восьмиугольник:

Повтори 9 [Вперед 40 Направо *n*]?

10 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*, где *n* – целое число, вызывающая передвижение черепашки на *n* шагов в направлении движения;

Направо *m*, где *m* – целое число, вызывающая изменение направления движения на *m* градусов по часовой стрелке.

Запись **Повтори 5 [Команда1 Команда2]** означает, что последовательность команд в квадратных скобках повторится 5 раз.

Черепашке был дан для исполнения следующий алгоритм:

Повтори 7 [Вперед 20 Направо 60]

Какая фигура появится на экране?

11 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 три команды:

Вперед *n*, где *n* – целое число, вызывающая передвижение черепашки на *n* шагов в направлении движения;

Направо *m*, где *m* – целое число, вызывающая изменение направления движения на *m* градусов по часовой стрелке;

Налево *m*, где *m* – целое число, вызывающая изменение направления движения на *m* градусов против часовой стрелки.

Запись **Повтори 5 [Команда1 Команда2]** означает, что последовательность команд в квадратных скобках повторится 5 раз.

Черепашке был дан для исполнения следующий алгоритм:

Повтори 5 [Вперед 30 Налево 72]

Какая фигура появится на экране? Где будет находиться эта фигура по отношению к Черепашке?

12 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 три команды:

Вперед *n*, где *n* – целое число, вызывающая передвижение черепашки на *n* шагов в направлении движения;

Направо *m*, где *m* – целое число, вызывающая изменение направления движения на *m* градусов по часовой стрелке;

Налево *m*, где *m* – целое число, вызывающая изменение направления движения на *m* градусов против часовой стрелки.

Запись **Повтори 5 [Команда1 Команда2]** означает, что последовательность команд в квадратных скобках повторится 5 раз.

Черепашке был дан для исполнения следующий алгоритм:

Повтори 22 [Вперед 10 Назад 10 Налево 18]

Какая фигура появится на экране?

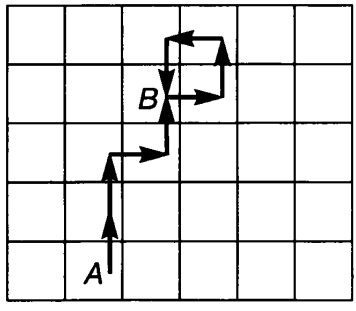
Робот

13 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

11313142

Какую последовательность из четырех 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: Начертим траекторию, по которой двигался Робот. Начальную точку движения обозначим буквой *A*, конечную – *B*. Из рисунка видно, что вернуться из точки *B* в точку *A* можно по программе из четырех команд: вниз, влево, вниз, вниз, т. е. 2422.



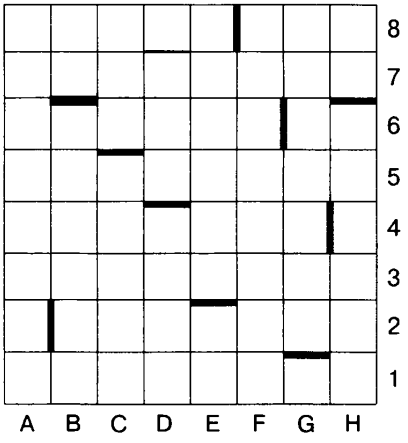
Ответ: 2422.

14 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

4242413

Какую последовательность из трех 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?

15 Дана система команд исполнителя Робот, «живущего» в прямоугольном лабиринте на клетчатой плоскости:



вверх
вниз
влево
вправо

При выполнении любой из этих команд Робот перемещается на соответствующую клетку.

Команды проверки истинности условия на наличие стены у той клетки, где он находится:

сверху свободно
снизу свободно
слева свободно
справа свободн

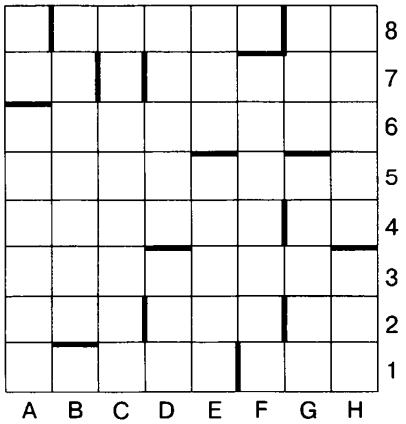
Если Робот начнет движение в сторону стены, то он разрушится.

Сколько клеток данного лабиринта соответствуют требованию, что, выполнив предложенную программу, Робот остановится в той же клетке, с которой он начал движение?

```
НАЧАЛО
ПОКА справа свободно ДЕЛАТЬ вправо
ПОКА снизу свободно ДЕЛАТЬ вниз
ПОКА слева свободно ДЕЛАТЬ влево
ПОКА сверху свободно ДЕЛАТЬ вверх
КОНЕЦ
```

В ответе запишите число – количество таких клеток, а далее, через запятые, их адреса (сначала идет латинская буква столбца, а затем цифра строки). Например, нижний левый угол лабиринта имеет адрес *A*1.

16 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 соот-

ветственно 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 свободн

Цикл

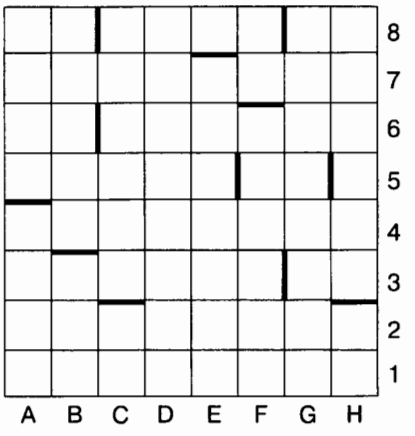
ПОКА < условие > команда

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

17 Дана система команд исполнителя Робот, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх
вниз
влево
вправо



При выполнении любой из этих команд Робот перемещается на соответствующую клетку.
Команды проверки истинности условия на наличие стены у той клетки, где он находится:

сверху свободно
снизу свободно
слева свободно
справа свободн

Если Робот начнет движение в сторону стены, то он разрушится.
Сколько клеток данного лабиринта соответствуют требованию, что выполнив предложенную программу, Робот остановится в той же клетке, с которой он начал движение?

НАЧАЛО
ПОКА снизу свободно ДЕЛАТЬ вниз
ПОКА справа свободно ДЕЛАТЬ вправо
ПОКА сверху свободно ДЕЛАТЬ вверх
ПОКА слева свободно ДЕЛАТЬ влево
КОНЕЦ

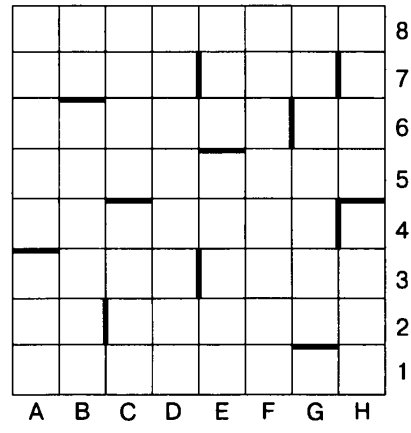
В ответе запишите число – количество таких клеток, а далее, через запятые, их адреса (сначала идет латинская буква столбца, а затем цифра строки).

Например, нижний левый угол лабиринта имеет адрес *A*1.

18 Дана 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 свободн

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 < слева свободно > влево
ПОКА < сверху свободно > вверх
ПОКА < справа свободно > вправо
ПОКА < снизу свободно > вниз
КОНЕЦ

Кузнечик

19 Имеется исполнитель Кузнечик, который живет на числовой оси. Система команд Кузнечика: "Вперед *N*" (Кузнечик прыгает вперед 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 51 команды, в которой команд "Назад 2" в 2 раза 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: Определим, сколько каких команд выполнил кузнечик. Получаем: 51 / 3 =17. Команд "Вперед 3" было 17, а "Назад 2" было 34. Кузнечик прыгнул вперед на 17 × 3 = 51 шаг, а назад на 34 × 2 = 68 шагов. Тем самым он оказался на 17 шагов назад от первоначальной точки. Последовательность команд в алгоритме в данном случае не имеет значения.

Ответ: Назад 17.

20 Имеется исполнитель Кузнечик, который живет на числовой оси. Система команд Кузнечика: "Вперед *N*" (Кузнечик прыгает вперед 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 52 команд, в которой команд "Назад 2" на 10 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?

Бусины

21 Цепочка из трех бусин формируется по следующему правилу: На первом месте в цепочке стоит одна из бусин М, Н, О. На втором – одна из бусин Л, М, О. На третьем месте – одна из бусин Л, М, Н, не стоящая в цепочке на первом или втором месте.
Какая из следующих цепочек создана по этому правилу:

1) НОН 2) НОМ 3) МНЛ 4) МНО

Полное решение.

На первом месте в цепочке стоит одна из бусин М, Н, О.
На втором шаге к первой бусине можно добавить одну из трех бусин Л, М, О. Получаем варианты:

МЛ, ММ, МО,
НЛ, НМ, НО,
ОЛ, ОМ, ОО.

На третьем шаге добавляем одну из бусин Л, М, Н с учетом правила построения цепочки. В результате складываются цепочки:

МЛН,
ММЛ, ММН,
МОЛ, МОН,
НЛМ,
НМЛ,
НОЛ, НОМ,
ОЛМ, ОЛН,
ОМЛ, ОМН,
ООЛ, ООМ, ООН.

Итого: 16 цепочек, из четырех предложенных подходит только "НОМ".

Краткое решение.

В цепочке "НОН" нарушено правило третьей бусины.
В цепочке "НОМ" все правила соблюдаются.
В цепочке "МНЛ" нарушено правило второй бусины.
В цепочке "МНО" нарушено правило третьей бусины.

Ответ: 2.

22 Для составления цепочек используются бусины, помеченные буквами: A, B, C, D, E. I. На первом месте в цепочке стоит одна из бусин, помеченных гласной буквой. На втором – любая согласная. На третьем месте – одна из бусин B, C, D, не стоящая в цепочке на втором месте. На четвертом – любая гласная, не стоящая в цепочке на первом месте.
Сколько существует вариантов цепочек, построенных по этому правилу?

1) 81 2) 54 3) 36 4) 24

23 Цепочка из трех бусин формируется по следующему правилу. На третьем месте в цепочке стоит одна из бусин А, Б, В. На втором – одна из бусин Б, В, Г. На первом месте – одна из бусин А, В, Г, не стоящая в цепочке на втором или третьем месте.
Какая из следующих цепочек создана по этому правилу?

1) БГВ 2) АВБ 3)ГАВ 4) АБА

24 Цепочка из трех бусин формируется по следующему правилу. На первом месте в цепочке стоит одна из бусин А, В, Г. На втором – одна из бусин

А, Б, В. На третьем месте – одна из бусин Б, В, Г, не стоящая в цепочке на первом или втором месте.

Какая из следующих цепочек создана по этому правилу?

1) ГВБ 2) БАГ 3) АББ 4) ВГБ

25 В понедельник в одном из классов должно быть проведено 4 урока – по химии, физике, информатике и литературе. Учителя высказали свои пожелания для составления расписания. Учитель химии хочет иметь третий или четвертый урок, учитель физики – первый или четвертый урок, учитель информатики – первый или второй, учитель литературы – второй или третий. Какой вариант расписания устроит всех учителей школы? (Обозначения: Х – химия, Ф – физика, И – информатика, Л – литература)

1) ИЛХФ 2) ИХЛФ 3) ФИХЛ 4) ИФЛХ

26 Для составления цепочек разрешается использовать бусины 5 типов, обозначаемых буквами А, Б, В, Г, Е. Каждая цепочка должна состоять из трех бусин, при этом должны соблюдаться следующие правила:

1) на первом месте стоит одна из букв: А, Г, Е;
2) после гласной буквы в цепочке не может снова идти гласная, а после согласной – согласная;
3) последней буквой не может быть В.

Какая из цепочек построена по этим правилам?

1) АЕБ 2) ГВА 3) ГЕБ 4) БАВ

27 Для составления цепочек используются бусины, помеченные буквами A, B, C, D, E. На первом месте стоит одна из букв: A, B, D. На втором – любая гласная, если первая буква согласная, и любая согласная, если первая гласная. На третьем месте – одна из бусин A, C, D, не стоящая на первом или втором месте.

Какая из перечисленных цепочек создана по этому правилу?

1) BCA 2) AEC 3) BAD 4) DED

28 Для составления цепочек используются бусины, помеченные буквами: M, N, O, P, U. В середине цепочки стоит одна из бусин N, O, P. На третьем – любая гласная, если первая буква гласная, и любая согласная, если первая согласная. На первом месте – одна из бусин M, N, O, не стоящая в цепочке в середине.

Какая из перечисленных цепочек создана по этому правилу?

1) NPO 2) MUN 3) ONP 4) NON

29 Для составления цепочек используются бусины, помеченные буквами A, B, C, D, E, G. Цепочка строится по следующему правилу. На первом месте в цепочке стоит одна из бусин C, D, E, G. На втором – любая согласная буква, если первая буква гласная, или любая гласная, если первая согласная. На третьем месте одна из бусин A, B, E, G, не стоящая в цепочке на первом или втором месте. На четвертом месте – любая гласная буква, не стоящая на втором или третьем месте.

Какая из перечисленных цепочек создана по этому правилу?

1) GCDA 2) CEBA 3) DADE 4) EBAA

Цепочки

30 Цепочки символов (строки) 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 цепочка цифр (одна за другой, подряд), а в конец приписывается еще 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) 1121123112112 34

Сколько раз в общей сложности встречаются в седьмой строке нечетные цифры (1, 3, 5, 7, 9)?

Решение: В первой строке встречается только цифра 1 – 1 раз. Во второй строке цифра 1 встречается 2 раза. В каждой следующей строке количество цифр 1 будет удваиваться. В строке с номером k цифра 1 встретится 2^{k-1} раз.

Аналогичные рассуждения можно провести для всех других нечетных цифр. Нечетная цифра n встречается 1 раз в строке с номером n. В строке с номером k ($k \geqslant n$) эта цифра встретится 2^{k-n} раз.

Подсчитаем, сколько раз в общей сложности встречаются в седьмой строке нечетные цифры:

$$2^{7-1} + 2^{7-3} + 2^{7-5} + 2^{7-7} = 2^6 + 2^4 + 2^2 + 2^0 = 64 + 16 + 4 + 1 = 85.$$

Следовательно, в седьмой строке будет 85 нечетных цифр.

Ответ: 85.

31 Цепочки символов (строки) создаются по следующему правилу.

Первая строка состоит из одного символа – цифры «1».

Каждая из последующих цепочек создается такими действиями: в очередную строку дважды записывается цепочка цифр из предыдущей строки (одна за другой, подряд), а в конец приписывается еще 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

Какая цифра стоит в восьмой строке на 250-м месте (считая слева направо)?

Решение: Найдем длину восьмой строки. По условию длина каждой последующей строки увеличивается в 2 раза по сравнению с предыдущей плюс еще один символ – цифра, обозначающая порядковый номер самой строки.

Получается, что длина строк составит:
1) 1 элемент в строке;
2) $1 \times 2 + 1 = 3$ элемента в строке;
3) $3 \times 2 + 1 = 7$;
4) $7 \times 2 + 1 = 15$;
5) $15 \times 2 + 1 = 31$;
6) $31 \times 2 + 1 = 63$;
7) $63 \times 2 + 1 = 127$;
8) $127 \times 2 + 1 = 255$ элементов в строке.

Требуется найти 250-й элемент в строке длиной 255 элементов. Это означает, что нам нужен шестой элемент с конца. Поскольку в конце строки на каждом шаге добавляется его номер (совпадающий с номером формируемой строки), последние восемь символов 8-й строки будут 12345678. А 6-м символом с конца будет 3.

Ответ: 3.

32 Строки (цепочки символов латинских букв) создаются по следующему правилу.
Первая строка состоит из одного символа – латинской буквы «A». Каждая из последующих цепочек создается такими действиями: в начало очередной строки записывается буква, чей порядковый номер в алфавите соответствует номеру строки (на *i*-м шаге пишется «*i*»-я буква алфавита). К этой букве справа дважды подряд приписывается предыдущая строка.
Вот первые 4 строки, созданные по этому правилу:

1) A
2) BAA
3) CBAABAA
4) DCBAABAACBAABAA

Латинский алфавит (для справки): *ABCDEFGHIJKLMNOPQRSTUVWXYZ*.
Запишите восемь символов подряд, стоящих в шестой строке с 55-го по 62-е место (считая слева направо).

33 Цепочки символов (строки) 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 цепочка цифр (одна за другой, подряд), а в конец приписывается еще 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) 1121123112112 34

Сколько раз в общей сложности встречаются в седьмой строке четные цифры (2, 4, 6, 8)?

34 Цепочки символов (строки) 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 цепочка цифр (одна за другой, подряд), а в конец приписывается еще 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) 1121123112112 34

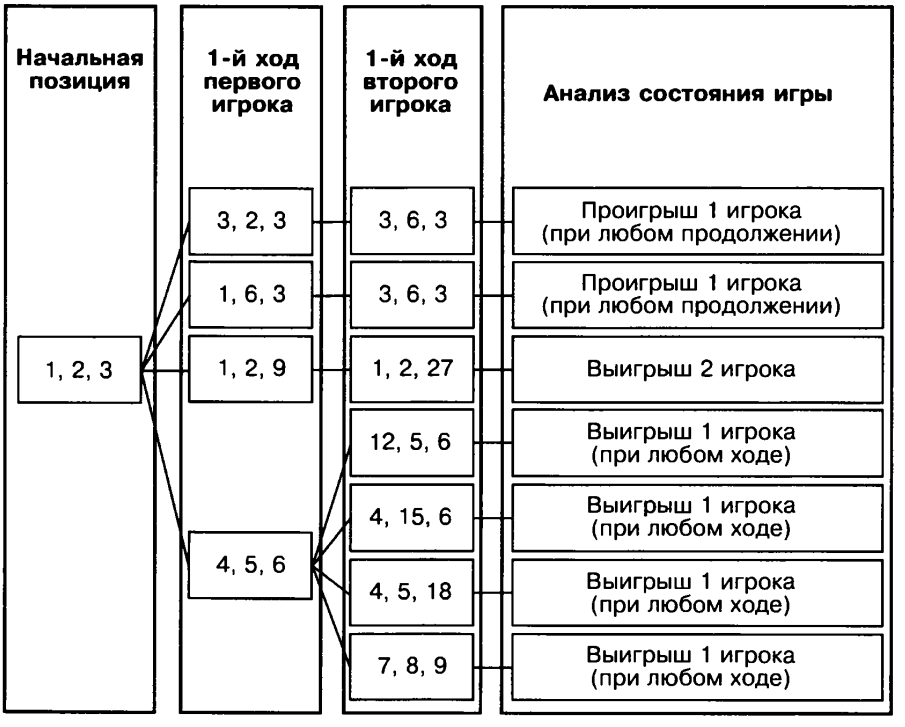
Сколько раз в общей сложности встречаются в десятой строке четные цифры (2, 4, 6, 8)?

Камешки

35 Два игрока играют в следующую игру.
Имеются три кучи камней, содержащих соответственно 1, 2, 3 камня. За один ход разрешается или утроить количество камней в какой-нибудь куче, или добавить по 3 камня в каждую из трех куч. Предполагается, что у каждого игрока имеется неограниченный запас камней.
Выигрывает тот игрок, после хода которого в какой-нибудь куче становится больше 20 камней или во всех трех кучах суммарно становится не менее 30 камней.

Игроки ходят по очереди. Выяснить, кто выигрывает при правильной игре, – первый или второй игрок.

Решение: Для решения задачи составим неполное дерево игры (дерево развития игры при различных продолжениях). Вершиной дерева игры будет начальное состояние игры. На уровне 1 дерева показаны все 4 возможные состояния игры после 1-го хода 1-го игрока; на уровне 2 из 16 возможных состояний игры после 1-го хода 2-го игрока показаны только те, которые существенно влияют на продолжение игры; далее дерево игры не ведется, а проводится анализ уже рассчитанных состояний игры.



Если 1-й игрок сделает свой первый ход 3, 2, 3 или 1, 6, 3, то 2-й игрок при правильной игре сделает ход 3, 6, 3, что приведет к проигрышу 1-го игрока (так как из состояния (3, 6, 3) 1-й игрок может своим ходом перевести игру в одно из четырех состояний – (9, 6, 3), (3, 18, 3), (3, 6, 9), (6, 9, 6), а для любого из этих состояний найдется ход 2-го игрока, дающий ему выигрыш).
Если 1-й игрок сделает свой первый ход 1, 2, 9, то он проиграет, так как 2-й игрок, сделав ход 1, 2, 27, добьется выигрыша.

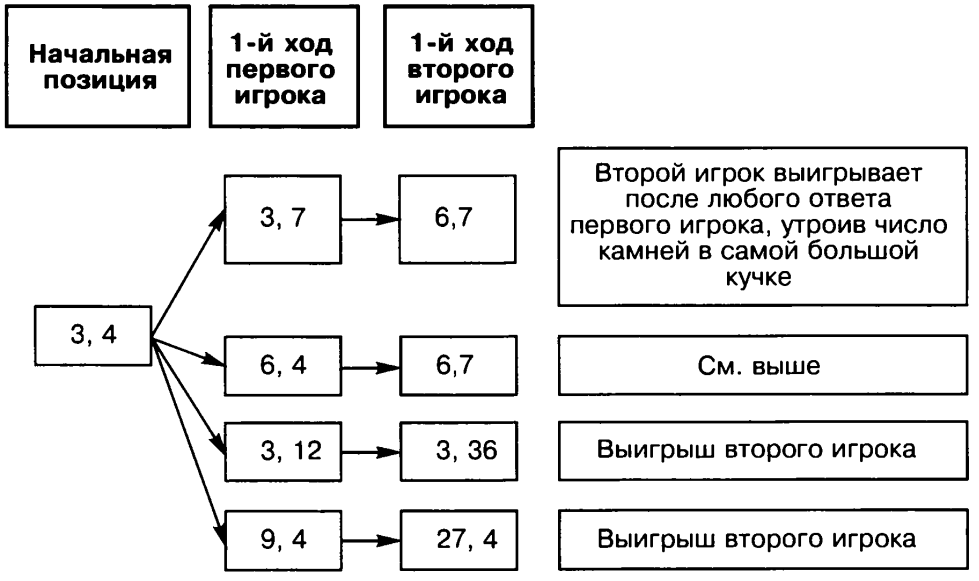
Наконец, если 1-й игрок сделает свой первый ход 4, 5, 6, то он выигрывает игру, так как на любой из четырех возможных ответов 2-го игрока (уровень 2 дерева) любой ход 1-го игрока при-

водит к победе. Таким образом, окончательный ответ к данной задаче: при правильной игре выигрывает 1-й игрок, при этом его первый ход должен быть 1, 2, 3 → 4, 5, 6.

Ответ: 1-й игрок.

36 Два игрока играют в следующую игру.
Перед ними лежат две кучки камней, в первой из которых 4, а во второй – 3 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 3 раза число камней в какой-то кучке, или добавляет 3 камня в какую-то кучку. Выигрывает игрок, после хода которого в одной из кучек становится не менее 24 камней. Кто выигрывает при безошибочной игре: игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

Решение: Выигрывает 2-й игрок. Для доказательства рассмотрим неполное дерево игры.



Из дерева видно, что при любом первом ходе 1-го игрока у 2-го игрока имеется выигрышный ход.

Ответ: 2-й игрок.

37 Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 3, а во второй – 2 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 2 раза число камней в какой-то кучке, или до-

бавляет 3 камня в какую-то кучку. Выигрывает игрок, после хода которого в одной из кучек становится не менее 18 камней. Кто выигрывает при безошибочной игре – игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

38 Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 3, а во второй – 4 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 3 раза число камней в какой-то кучке, или добавляет по 2 камня в каждую кучку. Выигрывает игрок, после хода которого в одной из кучек становится не менее 27 камней. Кто выигрывает при безошибочной игре – игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

39 Два игрока играют в следующую игру.
Перед ними лежат две кучки камней, в первой из которых 1 камень, а во второй – 2 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 2 раза число камней в какой-то кучке, или добавляет 2 камня в какую-то кучку. Выигрывает игрок, после хода которого общее число камней в двух кучках становится не менее 12 камней. Кто выигрывает при правильной игре – игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

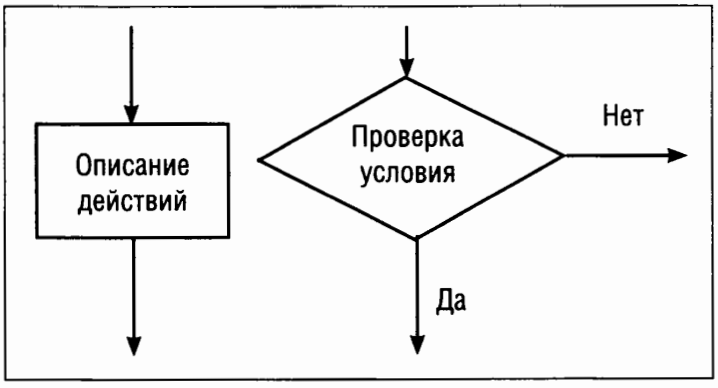
40 (*Демо-вариант 2011 г., задача С3*).
Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 6, а во второй – 5 камней. У каждого игрока неограниченное количество камней. Игроки ходят по очереди. Ход состоит в том, что игрок увеличивает или в 2, или в 3 раза число камней в какой-то куче. Выигрывает игрок, после хода которого общее число камней в двух кучах становится не менее 48. Кто выигрывает при безошибочной игре обоих игроков – игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

Представление алгоритмов

Алгоритмом называют последовательность точных и понятных исполнителю команд, которая определяет порядок конечного числа действий, необходимых для решения поставленной задачи, а также их содержание.

Графический способ представления алгоритмов является более наглядным по сравнению со словесным описанием. В графическом представлении алгоритм изображается в виде последовательности функциональных блоков, соединенных между собой ветвями, в каждом из которых требуется выполнить одно или несколько действий. Такое графическое представление алгоритма называется блок-схемой. Каждому типу действий (вводу исходных данных, проверке выполнимости условий, выполнению операции присваивания и т. д.) в блок-схеме соответствует своя геометрическая фигура, представленная в виде блочного символа. С помощью ветвей, соединяющих блочные символы, задается очередность выполнения действий.

В приведенных ниже задачах используются базовые алгоритмические структуры – «следование» и «ветвление».

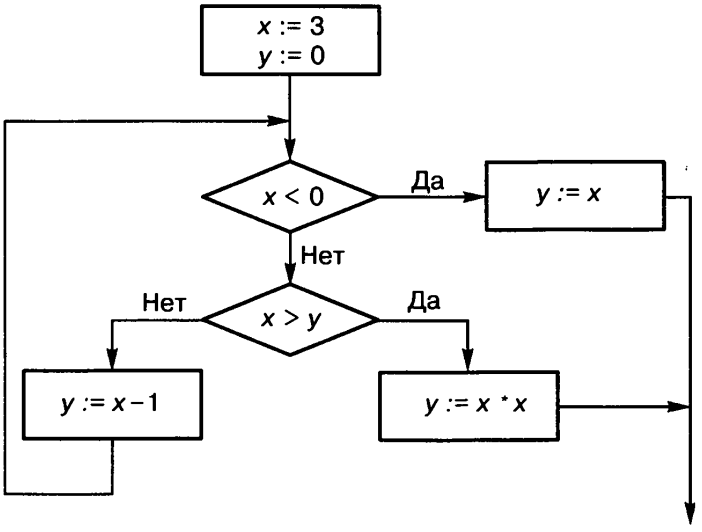


Структура «следование» определяет последовательность действий, которые требуется выполнить одно за другим. В блок-схемах этому соответствует прямоугольник, внутри которого написано одно или несколько действий, выполняемых по порядку. Дальнейший переход осуществляется по ветвям в направлении, указанном стрелками.

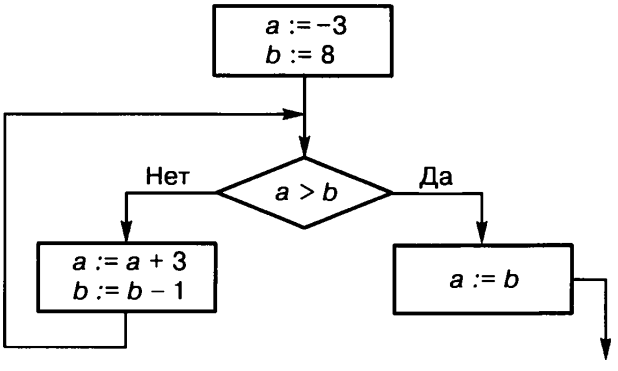
Структура «ветвление» обеспечивает переход по одной из ветвей алгоритма в зависимости от выполнения или невыполнения условия. В блок – схемах этой структуре соответствует геометрическая фигура ромб, внутри которого написано условие, проверку которого нужно осуществить. Если условие выполняется, то осуществляется переход по ветви ДА, если же не выполняется, то осуществляется переход по ветви НЕТ.

Задачи к разделу «Представление алгоритмов»

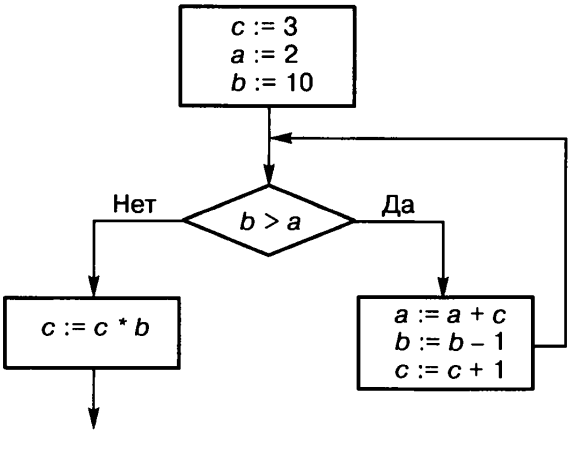
1 Определите значение переменной y после выполнения фрагмента программы:



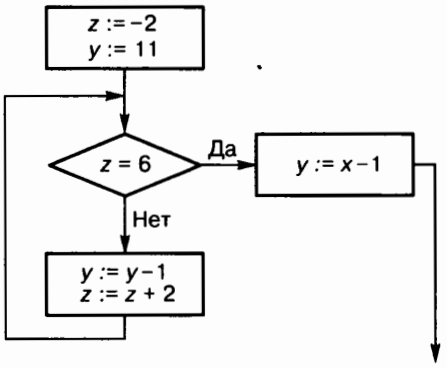
2 Определите значение переменных *a* и *b* после выполнения фрагмента программы:



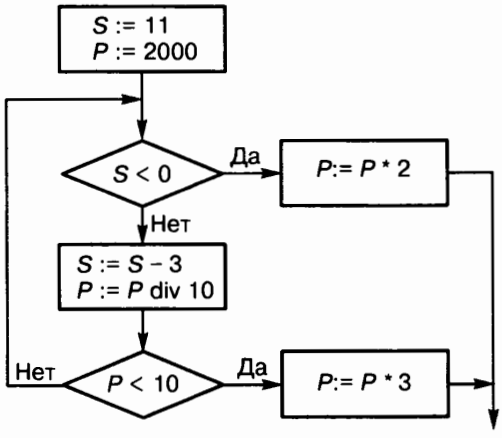
3 Определите значение переменной *c* после выполнения фрагмента программы:



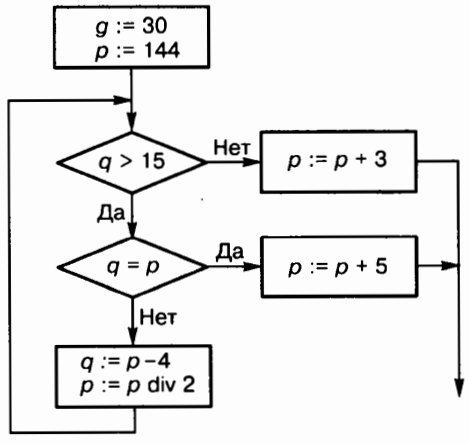
4 Определите значение переменной y после выполнения фрагмента алгоритма:



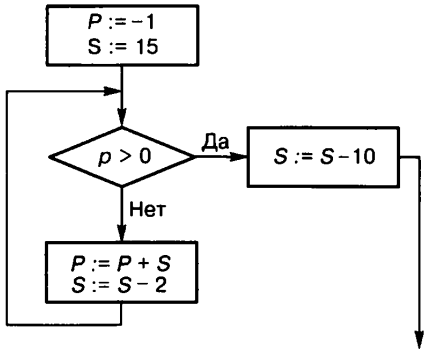
5 Определите значение целочисленных переменных S и P после выполнения фрагмента алгоритма:



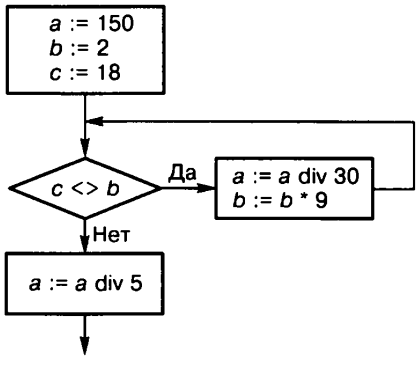
6 Определите значение целочисленной переменной p после выполнения фрагмента алгоритма:



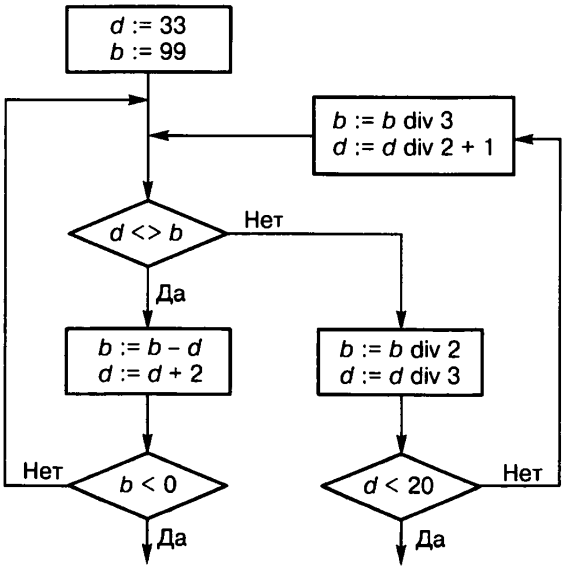
7 Определите значение переменной S после выполнения фрагмента алгоритма:



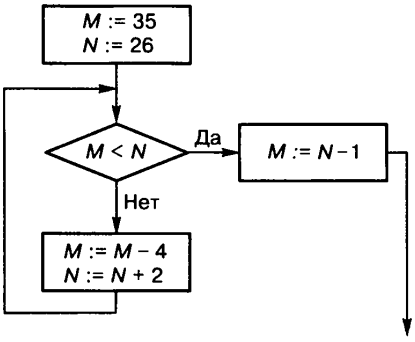
8 Определите значение переменных a после выполнения фрагмента алгоритма:



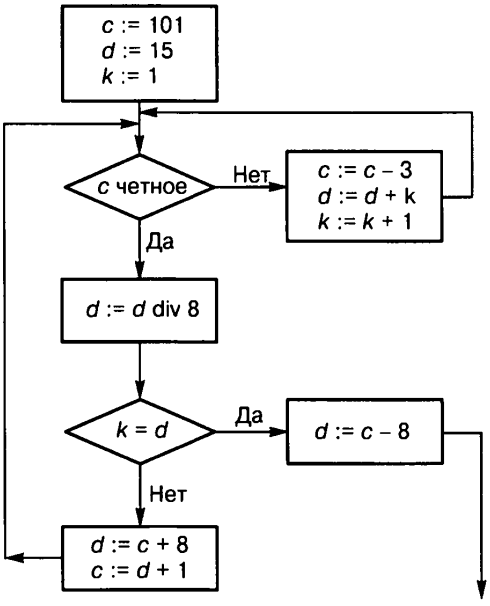
9 Определите значение целочисленных переменных b и d после выполнения фрагмента алгоритма:



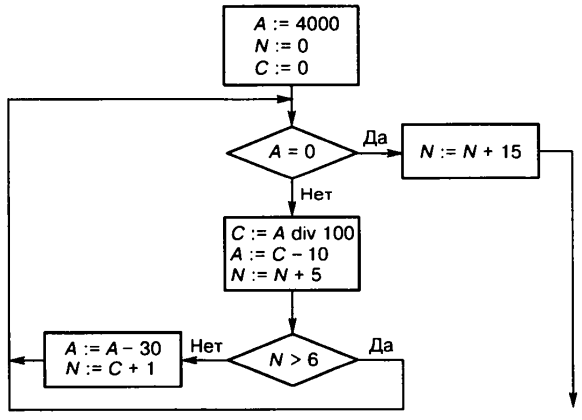
10 Определите значение целочисленной переменной M после выполнения фрагмента алгоритма:



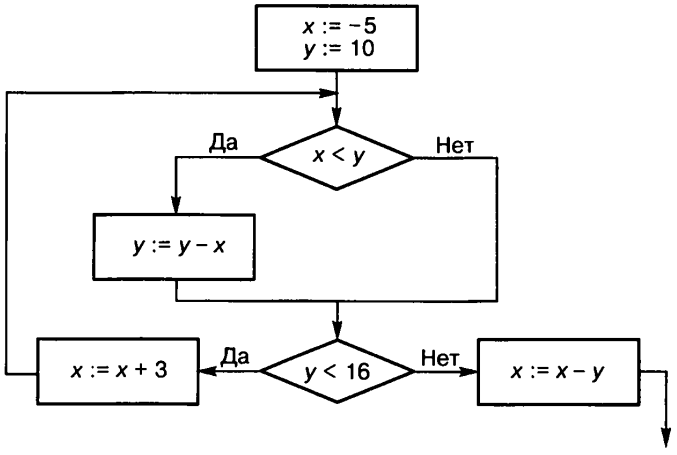
11 Определите значение целочисленной переменной d после выполнения фрагмента алгоритма:



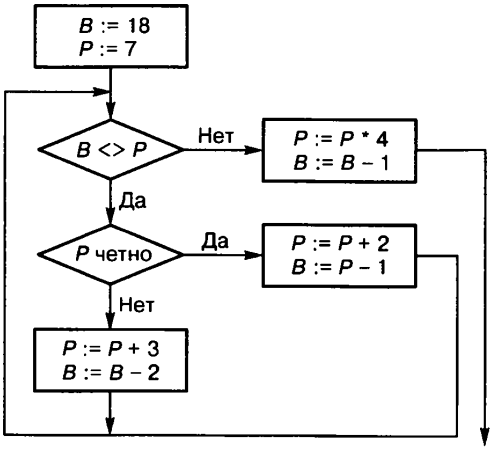
12 Определите значение переменной N после выполнения фрагмента алгоритма:



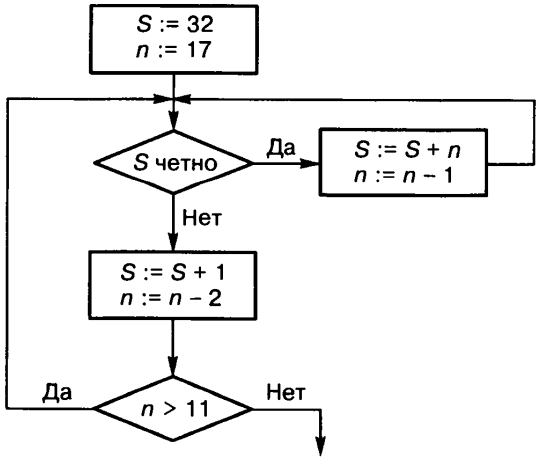
13 Определите значение переменной x после выполнения фрагмента алгоритма:



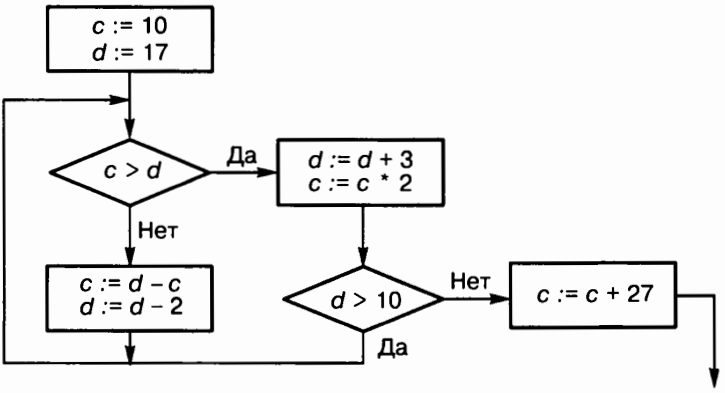
14 Определите значение целочисленных переменных B и P после выполнения фрагмента алгоритма:



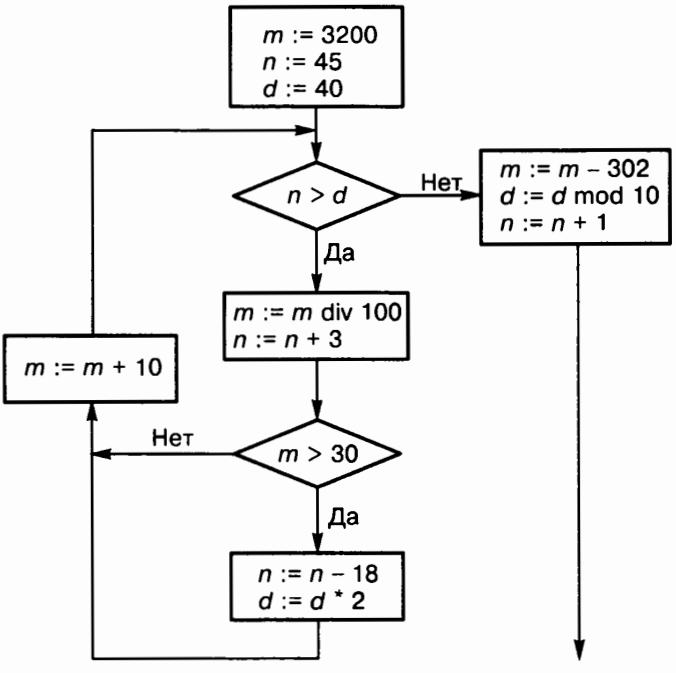
15 Определите значение переменной S после выполнения фрагмента алгоритма:



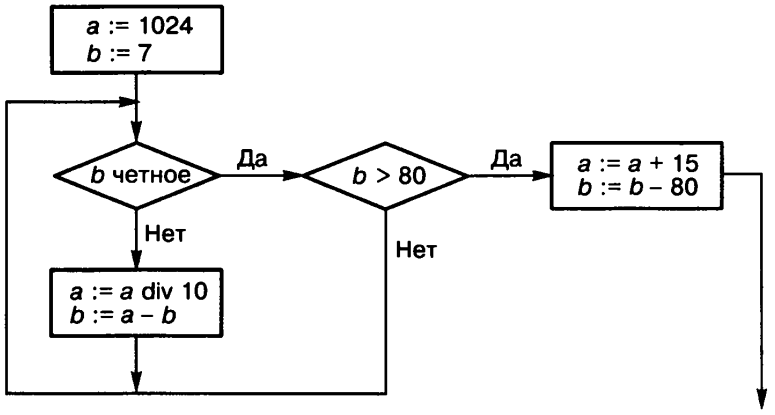
16 Определите значение переменных *c* и *d* после выполнения фрагмента алгоритма:



17 Определите значение переменных *m*, *n*, *d* после выполнения фрагмента алгоритма:



18 Определите значение переменных *a* и *b* после выполнения фрагмента алгоритма:



Задачи, представленные далее, содержат в себе описание фрагментов программ на языке программирования: Паскаль.

19 Определите значения *a* и *b* после выполнения фрагмента программы:

```
a := 51;
b := 17;
c := a div b;
a := a + c;
b := b - c;
```

20 Определите значения *x* и *y* после выполнения фрагмента программы:

```
x := 1000;
y := 3;
x := x div y;
y := y * x;
```

21 Определите значение *d* после выполнения фрагмента программы:

```
d := 3255;
f := d div 15 - 17;
z := d div (f * 5);
d := d mod (z + 1);
```

22 Определите значения *t*, *x* и *y* после выполнения фрагмента программы:

```
t := 169;
x := (t mod 15) * 100;
y := x div 110;
x := (x*2) div y;
x := x div 22;
t := x - 9;
```

23 Определите значения *a* и *b* после выполнения фрагмента программы:

```
a := 3789;
b := (a mod 1000) + 11;
c := b div 100;
a := (a div 1000) +b;
b := a + c;
```

24 Определите значения *t*, *x* и *y* после выполнения фрагмента программы:

```
x := 4;
y := 132;
t := x mod y;
x := x + t +15;
y := y div x;
t := t + y;
```

25 Определите значения *c* и *d* после выполнения фрагмента программы:

```
c := 21;
d := 8;
c := c div d;
d := c * d;
c := d mod c;
```

26 Определите значения *g* после выполнения фрагмента программы:

```
f := 5;
g := 21 - f;
t := g div 3;
f := (2*f) mod t;
g := g + f + t;
```

27 Значения элементов массивов *A*[1..10] и *B*[1..10] задаются с помощью операторов цикла. Какими будут значения *A*[2] и *B*[9] после выполнения фрагмента программы?

```
for n := 1 to 10 do
   A[n] := n - 8;
for m := 1 to 10 do
   B[m] := A[m] + m;
```

28 Значения элементов массивов *A*[1..100] и *B*[1..100] задаются с помощью операторов цикла. Сколько элементов массива *B* будут отрицательными после выполнения фрагмента программы?

```
for m := 1 to 100 do
   A[m] := 300 - m*100;
for n := 1 to 100 do
   B[n] := A[n] + 200;
```

29 Значения элементов двумерного массива *A* размерностью 6 × 6 задаются с помощью вложенного оператора цикла. Сколько элементов массива *A* будут неположительными после выполнения фрагмента программы?

```
for n := 1 to 6 do
   for m := 1 to 6 do
       A[n, m] := m + n - 3;
```

30 Все элементы двумерного массива *B* размерностью 7 × 7 первоначально равны 2. Чему будет равно значение *B*[5,2] после выполнения фрагмента программы?

```
for m := 1 to 7 do
  B[m, 2] := B[m, 2] * 2;
for n := 1 to 7 do
  B[5, n] := B[5, n] + 3*n;
```

31 Все элементы массива *C* [1..10] первоначально равны 1. Чему будет равно значение элемента *C* [7] после выполнения фрагмента программы?

```
for m := 1 to 10 do
     C[m] := C[m] + m;
for n := 1 to 10 do
     C[n] := C[n] + 9;
```

32 Сколько положительных элементов будет в массиве *C*[1..100] после выполнения фрагмента программы?

```
for i := 1 to 100 do
     C[101-i] := - i - 98;
for j := 1 to 100 do
     C[j] := C[j] + 150;
```

33 Все элементы двумерного массива *Z* размерностью 10 × 10 первоначально равны 0, затем их значения меняются с помощью вложенного оператора цикла. Сколько элементов будет равно 5 после выполнения фрагмента программы?

```
for n:= 1 to 10 do
   for m := n to 10 do
      Z[n,m] := 5 ;
```

34 Сколько элементов массива *C* будут иметь отрицательные значения после выполнения фрагмента программы, если значения элементов двух массивов *C*[1..50] и *B*[1..50] изменяются с помощью операторов цикла?

```
for n:= 1 to 50 do
     B[n] := - n + 10;
for m := 1 to 50 do
     C[m] := B[m] * 2 - 5;
```

35 Значения элементов двумерного массива A размерностью 80 × 80 изменяются фрагментом программы. Что делает данный алгоритм?

```
n := 3;
for j:= 1 to 80 do
begin
      z := B[j,j];
      B[j,j] := B[j,n];
      B[j,n] := z;
end;
```

36 Значения элементов двумерного массива D размерностью 20 × 20 задаются с помощью вложенного оператора цикла. Чему будет равна сумма элементов последней строки квадратной таблицы размерностью 20 × 20, в виде которой можно представить данный массив, после выполнения фрагмента программы?

```
for i := 1 to 20 do
    for j:= 1 to 20 do
      begin
       D[i,j] := j - i + 8;
      end;
```

Глава 4

Моделирование и компьютерный эксперимент

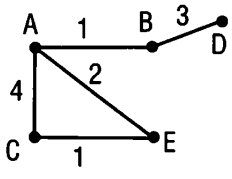
В задачах данного раздела используется табличный и графический способы представления информации.

Задачи могут быть связаны с переводом представления данных из одного вида в другой.

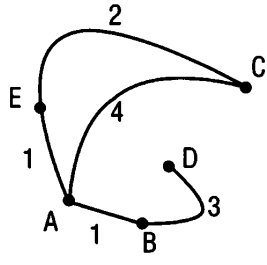
1 (*Демо-вариант 2006 г., задача А12*)

В таблице приведена стоимость перевозок между соседними станциями. Укажите схему, соответствующую таблице.

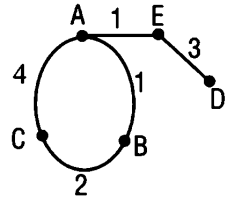
	A	B	C	D	E
A		1	4		1
B	1			3	
C	4				2
D		3			
E	1		2		



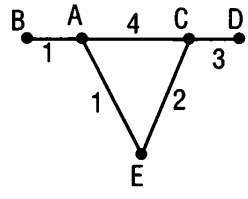
1)



2)



3)



4)

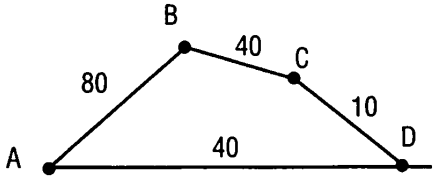
Решение. Таблицу надо преобразовать в схему. Для этого каждому названию строки/столбца в таблице надо поставить в соответствие одну точку на плоскости. Если в ячейке на пересечении строки и столбца стоит число, то такие точки надо соединить линией, рядом с которой написать это число.

Вообще во многих задачах найти правильный ответ легче всего, если представить описание задачи в графической форме.

Ответ: 2.

2 Грунтовая дорога проходит последовательно через населенные пункты A, B, C и D. При этом длина дороги между A и B равна 80 км, между B и C – 40 км, между C и D – 10 км.
Между A и D построили новое асфальтовое шоссе длиной 40 км. Оцените минимально возможное время движения велосипедиста из пункта A в пункт B, если его скорость по грунтовой дороге – 20 км/час, по шоссе – 40 км/час.

Решение. Основываясь на условии задачи, строим схему расположения дорог:



1. Рассчитаем время движения велосипедиста по грунтовой дороге по маршруту A → B:

$$t = 80 / 20 = 4 \text{ часа.}$$

2. Рассчитаем время движения по маршруту A → D → C → B.

A → D – шоссе: $t_1 = 40 / 40 = 1$ час;

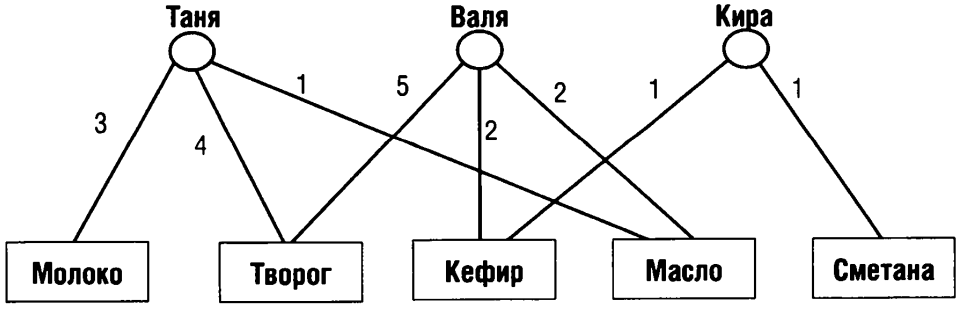
D → C → B – грунтовая дорога: $t_2 = (40 + 10) / 20 = 2{,}5$ часа;

$t = t_1 + t_2 = 1 + 2{,}5 = 3{,}5$ часа.

Ответ: 3,5 часа.

Задачи к разделу «Моделирование и компьютерный эксперимент»

1 Три девочки – Таня, Валя и Кира – пошли в магазин. На схеме изображено, что и в каком количестве купила каждая девочка. Выберите таблицу, соответствующую приведенной схеме.



1)

	Молоко	Творог	Кефир	Масло	Сметана
Таня		4		1	1
Валя			5		
Кира	2		2		2

2)

	Молоко	Творог	Кефир	Масло	Сметана
Таня	5		3		1
Валя	2	4			1
Кира	3		1		

3)

	Молоко	Творог	Кефир	Масло	Сметана
Таня		2	2		3
Валя	5			1	1
Кира			4		

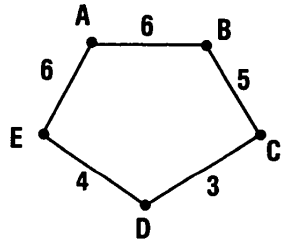
4)

	Молоко	Творог	Кефир	Масло	Сметана
Таня	3	4		1	
Валя		5	2	2	
Кира			1		1

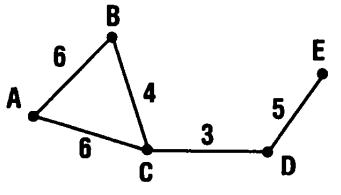
2 В таблице приведена стоимость железнодорожных перевозок между пятью станциями. Укажите схему, соответствующую таблице.

	A	B	C	D	E
A		6	6		
B	6		4		
C	6	4		3	
D			3		5
E				5	

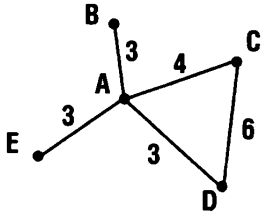
1)



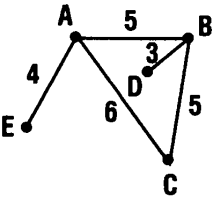
2)



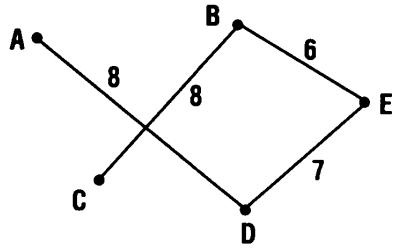
3)



4)



3 На рисунке представлена схема автомобильных перевозок между соседними городами. Укажите таблицу, соответствующую этой схеме.



	A	B	C	D	E
A		8			
B	8		8		
C		8		6	
D			6		7
E				7	

1)

	A	B	C	D	E
A					8
B					6
C					7
D					6
E	8	6	7	6	

2)

	A	B	C	D	E
A				8	
B			8		6
C		8			
D	8				7
E		6		7	

3)

	A	B	C	D	E
A		6		6	
B	6		7		
C		7			8
D	6				
E			8		

4)

4 Пятеро мальчиков живут в одном доме: Петя, Саша, Коля, Иван, Гриша. Петя любит кататься с Колей на велосипеде, а в кино ходит вместе с Гришей. Иван и Саша обмениваются музыкальными дисками. Коля ездил в турпоездку вместе с Сашей и Иваном. Саша играет в футбол с Иваном и Петей. Гриша ходит в компьютерный клуб с Сашей и Колей. Кто из мальчиков не поддерживает между собой никаких отношений?

Рекомендации. Задача решается с помощью графа.

---

5 Шесть девочек учатся в одном классе: Оля, Наташа, Надя, Вика, Лена, Света. Оля, Надя и Света живут в одном доме и вместе ходят в школу. Вика и Света занимаются танцами. Наташа вместе с Леной любят математику и ходят на математический кружок. Лена, Света и Надя вместе ходили в турпоход. Наташа и Оля любят музыку и часто обмениваются дисками. У кого из девочек меньше всего друзей?

Рекомендации. Задача решается с помощью графа.

---

6 Восемь девочек учатся в одном классе: Оля, Наташа, Надя, Вика, Лена, Света, Галя и Лариса. Вика и Света занимаются танцами. Наташа вместе с Леной любят математику и ходят на математический кружок. Лена, Света и Надя вместе ходили в турпоход. Наташа и Оля любят музыку и часто обмениваются дисками. Галя, Лариса и Света учатся в одном классе и вместе ходят в школу. У кого из девочек больше всего друзей?

Рекомендации. Задача решается с помощью графа.

---

7 Для переправы на другой берег реки используют 2 лодки:
1-я берет на борт 10 человек и тратит на путь туда и обратно 12 минут;
2-я берет на борт 3 человека и тратит 3 минуты.
Для переправы можно воспользоваться только одним типом лодки. Какую лодку следует выбрать, чтобы как можно быстрее перевезти 18 человек на другой берег?

8 Для переправы на другой берег реки используют 2 лодки:
1-я берет на борт 10 человек и тратит на путь туда и обратно 12 минут;
2-я берет на борт 3 человека и тратит 4 минуты.
Для переправы можно воспользоваться только одним типом лодки. Какую лодку следует выбрать, чтобы как можно быстрее перевезти 18 человек на другой берег?

9 В высотном здании несколько лифтов, поднимающихся на разные этажи. Лифт, который идет до 28-го этажа, сломался. Теперь туда можно добраться двумя способами:

1) подняться на лифте до 10-го этажа, там сделать пересадку на другой лифт, идущий без остановок до 28-го этажа;

2) подняться на лифте до 25-го этажа, потом идти пешком до 28-го этажа.

Два друга поспорили: кто быстрее доберется до 28-го этажа высотного здания? Сергей поднимается первым способом, а Вячеслав – вторым. Ответьте, чем закончится спор, если известно, что:

- лифту требуется для подъема на 1-й этаж 30 секунд, если он останавливается на этаже, и 8 секунд, если не останавливается;
- подъем пешком на 1-й этаж занимает 1 минуту;
- лифт, идущий до 10-го этажа, останавливался 5 раз, идущий до 25-го этажа – ни разу, идущий от 10-го до 28-го этажа – ни разу.

1) Сергей поднялся быстрее; 2) Вячеслав поднялся быстрее;
3) друзья добрались одновременно.

10 Из города Медвежий в город Озерный надо перевезти 20 человек. Маршрут проходит через поселок Брусничный. Перевозка производится в 2 этапа. Транспорт из Медвежьего в Брусничный перевозит за 1 раз 5 человек, затрачивает на это 30 минут и отправляется из Медвежьего каждые 30 минут. Транспорт из Брусничного в Озерный перевозит за 1 раз 10 человек, затрачивает на это 30 минут, отправляясь из Брусничного в момент, когда наберется ровно 10 человек. Через какое время все 20 человек окажутся в Озерном?

Рекомендации. Чтобы эффективно решить задачу, для каждого из трех пунктов надо нарисовать шкалу времени. Эти шкалы расположить точно друг под другом, расставить на них деления по 30 минут каждое и через каждые 30 минут отметить на шкале времени количество пассажиров на каждом пункте.

	0	**30**	**60**	**90**	**120**	**150**
Пункт 1						
Пункт 2						
Пункт 3						

11 Мальчика Лешу пригласили на день рожденья в поселок Дачи к 14:00. Из города до поселка Дачи можно добраться двумя способами:
1) на электричке до станции Яблочное и далее на автобусе № 10;
2) на электричке до станции Грушевое и далее на автобусе № 10.
Автобус № 10 следует от станции Яблочное до поселка Дачи, проезжая станцию Грушевое.
Мальчик Леша приехал на вокзал в городе в 11:50. До какой станции и на какой поезд он должен взять билет, чтобы не опоздать на день рожденья? В таблицах приведено расписание отправления электричек из города и автобуса № 10.

Расписание электричек

Город	**Яблочное**	**Грушевое**	**Озерки**
11:55	Не останавливается	13:45	14:30
12:00	Не останавливается	Не останавливается	14:50
12:10	13:00	13:30	
12:22	13:03		
12:30	13:15	13:45	

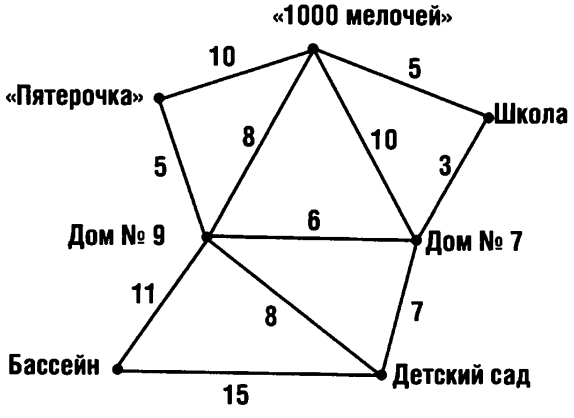
Расписание автобуса № 10

Яблочное	Грушевое	Дачи
12:25	12:59	13:25
12:40	13:20	13:50
12:58	13:35	14:00
13:25	13:55	14:15
13:46	14:38	15:00

12 Таблица, в которой сравнивается возраст трех человек, устроена следующим образом. Числа, стоящие на пересечении строк и столбцов, означают разницу в возрасте между людьми. Укажите абсолютную величину числа, которое должно стоять в клетке со знаком «?».

	Гриша	Артем	Олег
Гриша		-5	1
Артем	5		?
Олег	-1	?	

13 В новом микрорайоне проложили удобные дорожки между основными объектами. Узлы схемы – основные объекты, числа – время перемещения между объектами по проложенным дорожкам.



Какая из приведенных ниже таблиц соответствует схеме? Таблицы устроены следующим образом: числа, стоящие на пересечении строк и столбцов, означают время, которое затрачивает пешеход на перемещение между соответствующими объектами. Если пересечение пусто, то дорожка между объектами не проложена.

1)

	«Пятерочка»	«1000 мелочей»	Школа	Дом №9	Дом №7	Бассейн	Детский сад
«Пятерочка»		10		5			6
«1000 мелочей»	10		5			7	
Школа		5			3		
Дом №9	5					15	8
Дом №7			3				
Бассейн		7		15			11
Детский сад	6			8		11	

2)

	«Пятерочка»	«1000 мелочей»	Школа	Дом №9	Дом №7	Бассейн	Детский сад
«Пятерочка»			10	8		11	
«1000 мелочей»				8	6		
Школа	10				3		
Дом №9	8	8			6		
Дом №7		6	3	6		5	7
Бассейн	3				5		
Детский сад			11		7		

3)

	«Пятерочка»	«1000 мелочей»	Школа	Дом №9	Дом №7	Бассейн	Детский сад
«Пятерочка»		15		5	10		
«1000 мелочей»	15		7	8			
Школа		7				3	
Дом №9	5	8			6	11	8
Дом №7	10			6			5
Бассейн			3	11			10
Детский сад				8	5	10	

4)

	«Пятерочка»	«1000 мелочей»	Школа	Дом №9	Дом №7	Бассейн	Детский сад
«Пятерочка»		10		5			
«1000 мелочей»	10		5	8	10		
Школа		5			3		
Дом №9	5	8			6	11	8
Дом №7		10	3	6			7
Бассейн				11			15
Детский сад				8	7	15	

14 В новом микрорайоне проложили удобные дорожки между основными объектами. Время, затрачиваемое пешеходом на переход между объектами, представлено в виде таблицы.

	Поликлиника	Станция метро	Школа	Дом №18	Дом №20	Дом №22	Бассейн	Детский сад
Поликлиника				5	6			
Станция метро				8	3	5	12	
Школа					10	10		
Дом №18	5	8			6		11	
Дом №20	6	3	10	6				7
Дом №22		5	10					5
Бассейн		12		11				
Детский сад					7	5		

Таблица устроена следующим образом: числа, стоящие на пересечении строк и столбцов, означают время, которое затрачивает пешеход на перемещение между соответствующими объектами. Если пересечение пусто, то дорожка между объектами не проложена.

Сколько минут нужно пешеходу, чтобы дойти от школы до бассейна по самому короткому пути?

15 Из города А в город Б можно проехать тремя дорогами:

1) напрямую до города Б по грунтовой дороге, ее длина 90 км;
2) через поселок Рыбачий по асфальтовой дороге; от А до Рыбачьего 90 км, а от Рыбачьего до Б – 30 км;
3) через поселок Лесной; до поселка Лесной проложено шоссе длиной 90 км, а от Лесного до города Б дорога длиной 10 км совсем разбита и требует ремонта.

По какой дороге быстрее всего добраться из А в Б, если:

- скорость передвижения по шоссе – 90 км/час;
- скорость передвижения по грунтовой дороге – 60 км/час;
- скорость передвижения по дороге, требующей ремонта, – 10 км/час.

16 Борис и Вера договорились пойти вместе в кино. Борис добирается до киноцентра на маршрутке №5, поездка занимает 10 минут. Вера доезжает до киноцентра за 20 минут на маршрутке №7.

Борис пришел на остановку маршрутки в 11:00 и оказался в очереди 32-м. Вера была на остановке в 11:10, перед ней оказалось 23 человека.

Кто приедет на встречу раньше и сколько времени будет ждать, если известно, что маршрутка вмещает в себя 10 человек и отправляется каждые 10 минут?

Рекомендации. Для решения задачи использовать шкалу времени.

Глава 5

Информационные и коммуникационные технологии

Программные средства информационных и коммуникационных технологий

Основные понятия

Файл – это упорядоченный набор данных, содержащий определенное количество информации (программу или данные), имеющий имя и хранящийся в памяти компьютера (как правило, долговременной). В качестве долговременной памяти могут использоваться различные носители: дисковые устройства, ленточные устройства, устройства на основе флэш-памяти, CD-ROM, дискеты. Имя накопителя состоит из двух символов: буквы английского алфавита и символа «:». Например, «C:» – имя жесткого диска. Подключаемым устройствам даются аналогичные имена.

Жесткий диск логически может быть поделен на разделы, которые воспринимаются как самостоятельные устройства. Имена разделам даются по тем же правилам.

Файловая система – способ организации хранения системной и пользовательской информации на носителях.

Программные средства для организации работы с файлами входят в состав операционной системы

Имя файла задается пользователем при создании или переименовании файла, состоит из двух частей – собственно имени файла и его расширения, соответствующего типу файла. Имя файла и расширение разделяются точкой.

Имя файла не может содержать следующие символы:

/ \ : * ? “ < > |

Тип файла задается создающей его программой автоматически при создании файла. Тип файла определяет способ хранения информации внутри файла.

Расширение имени файла записывается после имени файла и отделяется от него точкой. Оно непосредственно связано с типом файла.

Примеры расширений	Описание
EXE, COM	Расширения для исполняемых файлов (программ)
SYS	Расширение файла системной программы или драйвера
BAT	Пакетный файл с командами для операционной системы
OBJ	Откомпилированный файл («полуфабрикат», является результатом компиляции текстового файла программы для будущей сборки из нескольких модулей OBJ в исполняемую программу)
BAK	Резервная копия файла
DAT	Файл данных со служебной информацией
TXT	Текстовый файл простой структуры с разбивкой текста по строкам
DOC	Текстовый файл в формате текстового редактора Microsoft Office
BAS	Файл с текстом программы на языке программирования BASIC
PAS	Файл с текстом программы на языке программирования PASCAL
JPG, BMP, TIFF, CDR, PNG	Графические файлы в различных форматах хранения

Размер файла – количество байтов в файле. В размер файла не включаются служебные байты для поддержки его физической организации.

Список прав доступа к файлу для конкретного пользователя или системной службы – права на чтение, запись, удаление, изменение файла, полный доступ к файлу, разрешение исполнять файл, чтение каталога, чтение и запись основных и дополнительных атрибутов файла, чтение прав доступа, смена владельца файла.

Прикладные программы работают с содержимым файла, добавляя, удаляя и изменяя в нем информацию.

Организация хранения информации на дисках компьютера

В памяти компьютера организовано структурированное хранение информации. Суть структуризации заключается в объединении объектов в отдельные группы. Каждая такая группа хранится в отдельном каталоге, имеющем имя. В операционной системе MS Windows каталог называется папкой. Имена папкам даются по тем же правилам, что и имена файлам. В отличие от имен файлов имена папок не имеют расширений.

Файловая система в операционных системах MS Windows и MS-DOS представлена многоуровневой иерархической системой папок (каталогов) и файлов.

Вершиной иерархии объектов является корневая папка (каталог) диска, который можно сравнить со стволом дерева: на нем растут ветки – папки (под-

каталоги), а на ветках располагаются листья – файлы. На каждом диске имеется ровно одна корневая папка (каталог). В ней регистрируются файлы и папки первого уровня. В папке первого уровня регистрируются файлы и папки (каталоги) второго уровня и т. д.

Текущая папка – папка, в которой пользователь находится в данный момент.

Путь к файлу – последовательность имен папок (каталогов) от корневой до текущей папки (в ней записан файл), разделенных символами, специфичными для конкретной операционной системы. В MS Windows и MS-DOS это знак \, в ОС UNIX это знак /. Пример пути к файлу: **C:\школа\Класс**.

Полное имя файла – это имя файла, дополненное путем к файлу. Его часто называют длинным именем. Например, **C:\Школа\Класс\Иванов.doc**.

Основные операции работы с файлами и папками: копирование и перемещение, удаление, переименование, создание папки. Следует помнить, что:

- при удалении файла «теряется» лишь запись о файле в каталоге;
- при копировании информация физически записывается в указанный каталог;
- при перемещении файл остается там, где он был; в другой каталог перемещается лишь его название.

Поиск файлов и папок

В команде поиска потерянных файлов и папок можно указывать как полное имя объекта (если вы его помните), так и *маску имени файла* (шаблон имени). Маска отличается от имени тем, что содержит знаки **?** и ***:

- ? означает ровно один произвольный символ;
- * означает любую последовательность символов произвольной длины, в том числе и пустую.

Примеры

Отч*.doc – все файлы типа **doc**, начинающиеся с букв «Отч», регистр значения не имеет;

??рестор*.ht* – все файлы, где первые два символа в имени любые, далее символы «рестор», заканчивается имя файла любыми символами, расширение файла начинается с «ht» и заканчивается любыми символами.

Задачи к разделу «Программные средства информационных и коммуникационных технологий»

1 Могут ли в памяти компьютера одновременно находиться
- папки с одинаковыми именами?
- файлы с одинаковыми именами и расширениями?

2 Могут ли в одной папке находится два файла : **Tasks.txt** и **Tasks.tif**?

3 Вы только что подготовили документ в системе Word. Какие из приведенных ниже расширений имен файлов можно выбрать при сохранении этого файла?
txt gif doc rtf exe bmp xls

4 В папке «Доклады» записано несколько файлов.

заголовок	14 КБ	CorelDRAW 11.0 Gr...	30.10.2006 8:07
застаека	1 197 КБ	Adobe Photoshop I...	05.10.2005 8:21
история	958 КБ	Документ Microsof...	18.08.2006 15:48
класс 230	24 КБ	Документ Microsof...	22.12.2005 18:20
литература	30 КБ	Документ Microsof...	29.12.2004 18:27
математика	1 217 КБ	Документ Microsof...	18.08.2006 15:45
мгу_черн	27 КБ	Рисунок JPEG	06.05.2005 10:39
мгу_черн	15 КБ	Файл Microsoft Offi...	06.05.2005 10:38
общая_осень_2005	288 КБ	PageMaker Publication	28.08.2005 6:02
перед_занятием	25 КБ	Документ Microsof...	05.03.2002 10:30
план	16 КБ	CorelDRAW 11.0 Gr...	08.10.2008 11:39
план	55 КБ	Рисунок JPEG	08.10.2008 11:40
презентация_день_науки2	4 775 КБ	Презентация Micro...	10.10.2008 20:32
программа_DELPHI	23 КБ	Документ Microsof...	15.06.2006 12:03
физика	1 128 КБ	Документ Microsof...	18.08.2006 15:30
химия	141 КБ	Документ Microsof...	10.06.2002 13:11

а) Файлы «физика», «химия» и «история» надо переписать на дискету 3,5 дюйма объемом 1,44 Мб. Поместятся ли они на эту дискету?

б) Файлы «литература» и «история» надо переписать на дискету 3,5 дюйма. Поместятся ли они на эту дискету, если на ней осталось 1100 Кбайт свободного пространства?

5 Брат и сестра пользуются одним компьютером, но на этом компьютере на одном диске у них разные папки. У брата в его папке находится 2000

файлов большого объема с фотографиями летнего путешествия. Сестра хочет иметь эти файлы в своей папке, а брату эти файлы не нужны. Какой операцией – «копировать» или «переместить» следует воспользоваться, чтобы файлы появились в папке сестры как можно быстрее?

6 У Пети на компьютере записано 3000 музыкальных файлов. Коля хочет, чтобы у него были такие же файлы, и просит переписать их ему на flash-носитель. Какой операцией – «копировать» или «переместить» – следует воспользоваться, чтобы файлы появились как можно быстрее?

7 Из текущей папки, содержащей большой объем данных, Степа хочет удалить 23 файла. Чтобы выполнить удаление как можно быстрее, он выделяет все эти файлы и применяет операцию удаления ко всем файлам одновременно. Какой клавишей на клавиатуре воспользовался Степа для одновременного выделения нескольких файлов, если известно, что файлы разбросаны по всему списку?
Shift; Alt; Ctrl; Enter; Пробел; Tab; Esc; Delete; Insert.

8 Из текущей папки, содержащей большой объем данных, Гриша хочет скопировать 10 файлов в другую папку. Чтобы выполнить копирование как можно быстрее, он выделяет эти файлы и применяет операцию копирования ко всем файлам одновременно. Какой клавишей на клавиатуре воспользовался Гриша для одновременного выделения нескольких файлов, если известно, что в списке объектов эти файлы располагаются подряд, а не разбросаны по всему списку?
Shift; Alt; Ctrl; Enter; Пробел; Tab; Esc; Delete; Insert.

9 Ученица несколько месяцев назад делала доклад по биологии про жизнь белых медведей. Какой шаблон (маска) отвечает имени файла с докладом, если она помнит только, что подготовила его с помощью редактора MS Word (версия 2003) и в названии была фраза со словом «медведь» в каком-либо падеже (например, он мог называться «Белый медведь» или «Жизнь медведей»)?

10 Пусть некоторый цифровой фотоаппарат сохраняет фотографии в формате JPG и задает им имена из 12 символов по следующим правилам:
- первые четыре символа – буквы FFTT;
- 5 и 6 символы – номер месяца, когда произведена съемка;
- 7 и 8 символы – номер дня в месяце;
- остальные 4 символа – номер фотографии в этот день (каждый день нумерация начинается с нуля).

Фотографии были переписаны на компьютер. Какой надо задать шаблон (маску), чтобы найти:

А) все фотографии, сделанные этим фотоаппаратом?
Б) все фотографии, сделанные в марте?
В) несколько первых фотографий, сделанных 7 ноября?
Г) фотографии, сделанные в ноябре и декабре?
Д) фотографии, сделанные в новогоднюю ночь?
Е) верно ли, что с помощью шаблона **FFTT*10*.JPG** мы получим все фотографии, сделанные 10 числа каждого месяца?

11 Какой ответ верен для утверждения: в группу файлов с маской **?обед*.tx*** входят файлы:
1) победа.txt 2) обед.tx 3) побед.text 4) беда.txt

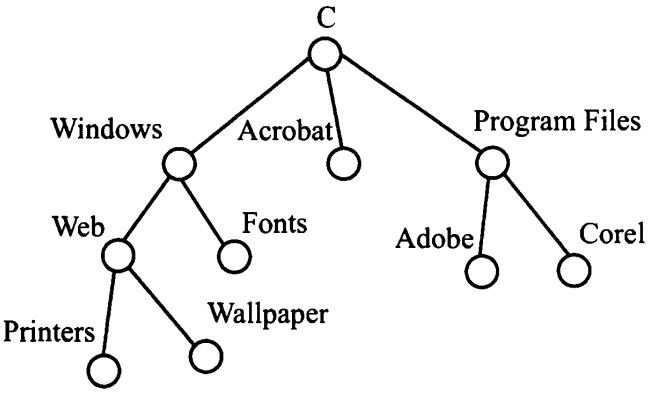
12 Наташе для работы на компьютере выделили папку с именем «Ната». Девочка создала в этой папке свои папки: «Кино», «Музыка», «Фото», «Уроки». В папку «Фото» Наташа скопировала четыре папки с летними путешествиями. В папке «Уроки» она создала две папки: «Доклады» и «Домашние задания». Все свои фильмы Наташа разделила на группы: «Комедии», «Детективы» и «Романтические». Папка «Музыка» пока не заполнена. Сколько всего папок у Наташи внутри папки «Ната»?
1) 4 2) 9 3) 7 4) 8 5) 13 6) 14

13 В семье Ромашковых трое детей школьного возраста, бабушка, дедушка, папа и мама. Все пользуются компьютером. Чтобы не мешать друг другу, для каждого на диске С: создана своя папка. Плюс есть общая для всех папка, куда записываются семейные расходы и ведется учет покупок. У мамы и у папы в их папках есть два раздела, относящихся к домашним делам и к их рабочим проблемам. Сколько всего папок было заведено на компьютере семьи Ромашковых?

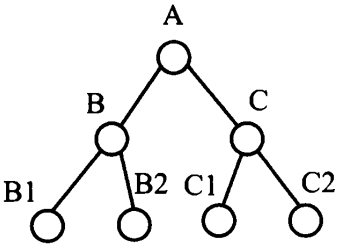
14 Файл с полным именем **C:\ОБУЧЕНИЕ\СТУДЕНТ\диплом.doc** скопировали в папку **ВЫПУСКНИКИ** на диске **D:**. Каково полное имя скопированного файла?

15 Файл с полным именем **C:\ОБУЧЕНИЕ\СТУДЕНТ\диплом.doc** скопировали в папку **РАБОТЫ**, находящуюся в той же папке **ОБУЧЕНИЕ**. Каково полное имя скопированного файла?

16 Запишите длинное имя файла **Ветер.bmp**, который хранится в папке **Wallpaper**.



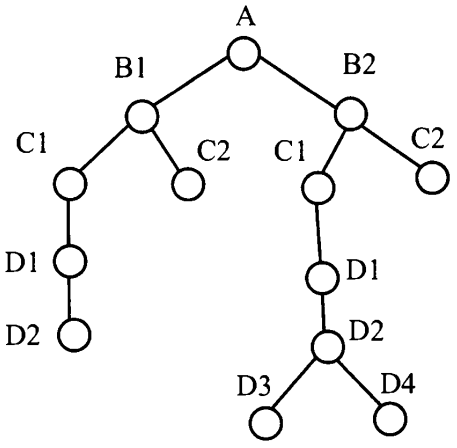
17 В папке **C2** записан файл **Text.txt**. Папку **C2** скопировали в папку **B**. Запишите новое длинное имя файла **Text.txt** в скопированной папке.



18 В некоторой папке был записан файл **letter.doc**. В этой же папке создали новую папку и скопировали в нее файл **letter.doc**. Полное имя скопированного файла **D:\ОБЩИЕ\УЧЕНИКИ\УСПЕВАЕМОСТЬ\ИВАНОВ\letter.doc**. Каково полное имя папки, в которой хранился файл до копирования?

19 Программист перемещался из папки в папку, либо спускаясь в папку на уровень ниже, либо поднимаясь на уровень выше. Он побывал последовательно в следующих папках: **P**, **T**, **C:**, **FF**, **K**. Каково полное имя папки, в которой оказался программист?

20 Программист находится в папке с именем **A:\B2\C2** (см. рис. ниже). Он покидает эту папку и переходит на один уровень выше. Затем программист опускается на два уровня ниже. Каково полное имя папки, в которой он оказался?



21 Пользователь перемещался из папки в папку, либо спускаясь в папку на уровень ниже, либо поднимаясь на уровень выше. Он побывал последовательно в следующих папках: **Flash**, **Студент**, **Обучение**, **D:**, **Проекты**, **Курс3**, **Группа10**. Каково полное имя папки, в которой программист находился изначально?

22 Файл с полным именем **C:\Обучение\Студент\Photoshop\Name.txt** переместили в папку **Проекты** диска **D:**. Каково полное имя перемещенного файла?

Технология обработки информации в электронных таблицах MS Excel

Типы данных

В ячейках электронной таблицы могут храниться данные, которые подразделяются на *константы* и *формулы*. Различают четыре типа констант.

1. **Числа.** Число может состоять из цифр (от 0 до 9) и некоторых специальных символов: + - ( ) : , / р . Е е %. В зависимости от формата вывода числа на экране могут представлять собой:
   - *десятичные числа* (1,5);
   - десятичные числа *в экспоненциальной форме* (1,5E3). Буква Е означает, что число слева от буквы Е надо умножить на 10 в степени, указанной справа от буквы Е. В приведенном примере: – $1{,}5 \times 0^3 = 1500$. Такие числа занимают меньше места на экране;
   - *проценты* (12 %; равно числу 0,12);
   - *даты* (05.02.09, 28/12/2009, Март 2009);

- *время* (11:05);
- *денежные значения* (5,1 р.; равно числу 5,1);
- *дроби* (1/2; равно числу 0,5).

2. **Логические значения**: ИСТИНА и ЛОЖЬ.
3. **Значения ошибок**.
4. **Текст** – любая последовательность символов, которая не начинается со знака «=» и не является числом, логическим значением или значением ошибки.

Параметры ячеек таблицы

Каждая ячейка характеризуется несколькими параметрами:

1) адрес ячейки;
2) содержимое ячейки (в частности, это может быть формула);
3) изображение – форматированный результат вычисления по формуле; по умолчанию на экране в ячейке отображается не формула, а ее результат;
4) формат – форма представления информации на экране. Состоит из:
   - формата представления чисел;
   - способа выравнивания текста в ячейке;
   - шрифта (характеризуется названием, размером, стилем, эффектами);
   - вида рамки ячейки;
   - вида заливки ячейки;
   - способа защиты данных в ячейке от изменений и просмотра;
5) имя, которое можно присвоить ячейке или диапазону ячеек;
6) примечание – текст или звук, которые можно поставить в соответствие ячейке.

Правила ввода чисел

1. При вводе *десятичных* чисел:
   - запятая отделяет целую часть числа от дробной;
   - между тысячами можно для удобства указывать пробел, например, 1 000 000;
   - введенный перед числом знак «плюс» игнорируется;
   - перед отрицательным числом нужно ввести знак «минус» или заключить число в круглые скобки.
2. При вводе дат день, месяц и год отделяются точками, знаком «минус» или символом «/», например: 01.10.94. Даты можно складывать и вычитать, при этом в формулах даты надо заключать в кавычки: “01.10.94“.

3. При вводе времени часы, минуты и секунды разделяются двоеточием, например 11:50:28 (секунды можно не указывать). В формулах время надо заключать в кавычки: “11:50:28”.
4. При вводе чисел в формате процентов после числа надо указать символ «%», например 120% отображает число 1,2 на экране в виде процентов.
5. При вводе денежных единиц можно указать символы «р.». Например, если в ячейку ввести денежное значение 40200р., то число 40200 будет отображаться на экране с символами «р.», а ячейке автоматически присвоится денежный тип данных.
6. Для ввода простой дроби надо набрать 0, затем пробел и саму дробь, например, 0 1/3. На экране отобразится 1/3, а в памяти будет храниться число 0,3333333.

Операции и порядок их выполнения

Далее перечислены операции, которые используются в выражениях, в порядке убывания приоритета их выполнения:

1) операции в скобках ( );
2) вычисление значения функции;
3) унарные, т. е. применяющиеся к одному операнду; знаки + и – (например, –8);
4) возведение в степень (знак ^);
5) умножение и деление – выполняются в порядке их следования в выражении слева направо (знаки * и /);
6) сложение и вычитание – выполняются в порядке их следования в выражении слева направо (знаки * и –);
7) операции над текстом – конкатенация (слияние) текста (знак &); например, в результате операции “кар”&”тон” получается слово «картон»;
8) операции отношения в логических выражениях (результат – логическая константа ИСТИНА или ЛОЖЬ); знаки: =, < (меньше), > (больше), <= (меньше либо равно), >= (больше либо равно), <> (не равно).

Относительные и абсолютные ссылки

Большинство формул содержит в себе адреса ячеек. При вводе формулы адрес ячейки можно ввести непосредственно с клавиатуры, но удобнее щелкнуть мышью по ячейке с нужным адресом.

Очень часто пользователю приходится создавать большие таблицы, ячейки которых содержат одинаковые по структуре формулы. Поскольку формула

является одним из типов данных, электронные таблицы позволяют их копировать.

Для правильного копирования были введены два вида ссылок. Ссылки вида *В6 – относительные*. Относительная ссылка представляет собой смещение от активной ячейки (той, в которую вводится формула) до ячейки, из которой требуется взять данное. Ссылка отображается в формуле как адрес нужной ячейки. При копировании формулы относительные ссылки автоматически корректируются, чтобы положение ячейки, на которую ссылается формула, относительно скопированной клетки было таким же, как и относительно копируемой.

Однако на практике в таблице могут быть ячейки, значения которых служат источником данных для многих однотипных формул, получаемых путем копирования, например, если в ячейке хранится курс рубля по отношению к доллару. При копировании формул ссылки на такие ячейки не должны изменяться. Чтобы при копировании формул ссылки не корректировались, их нужно зафиксировать. Для этого перед частью ссылки, которая должна остаться неизменной, ставится символ «$». Такие ссылки называются *абсолютными.* Например, ссылка **$H$3** будет неизменной при копировании, **H$3** будет корректироваться только при изменении столбца (абсолютная строка), а **$H3** – только при изменении строки (абсолютный столбец).

Знак «$» можно вставить:

- с клавиатуры;
- нажимая клавишу F4 сразу после вставки ссылки в формулу до получения требуемой комбинации символов «$»;
- выделив необходимую ссылку в формуле мышью и нажимая клавишу F4 в режиме правки.

Встроенные функции MS Excel

При работе с электронными таблицами достаточно часто приходится использовать встроенные функции. Для вставки наиболее часто употребляемых функций предусмотрена кнопка со списком , располагающаяся в одной из верхних панелей. Если щелкнуть по треугольничку, то список раскроется и из него можно будет выбрать следующие функции:

- суммировать – функция **СУММ()**: вычисляет сумму значений диапазона ячеек; диапазон можно задать вручную или выделить на рабочем листе мышью;
- среднее – функция **СРЗНАЧ()**: вычисляет среднее арифметическое диапазона ячеек;
- число – функция **СЧЕТ()**; подсчитывает количество числовых значений в выделенном диапазоне;

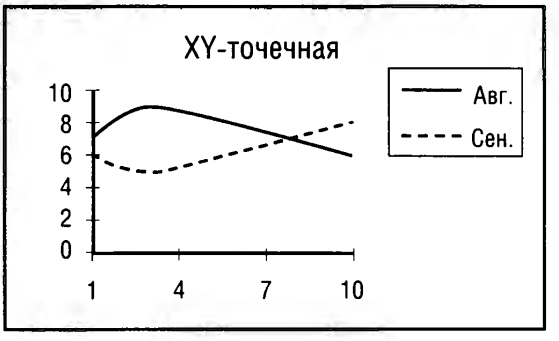
- максимум – функция **МАКС()**: определяет максимальное значение в диапазоне;
- минимум – функция **МИН()**: определяет минимальное значение в диапазоне.

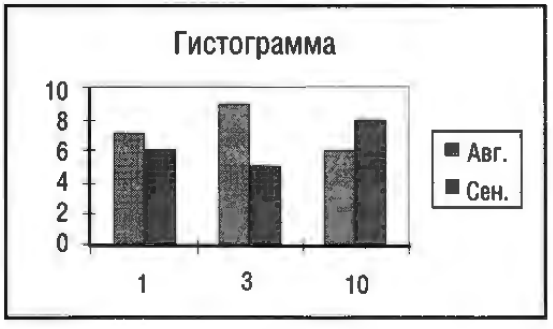
Если среди перечисленных в списке функций требуемой не оказалось, следует выбрать пункт списка «Другие функции…», вызвав тем самым мастер функций. Также мастер функций можно вызвать с помощью кнопки fx; как правило, на панели она располагается рядом с уже упомянутой кнопкой.

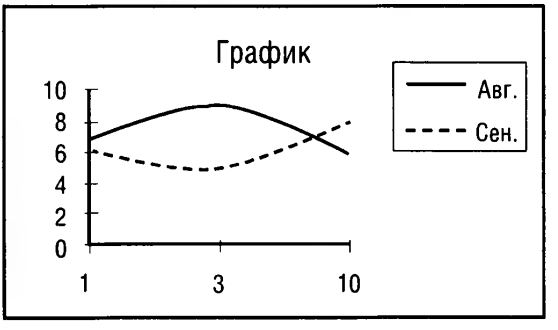
Тогда, например, можно воспользоваться функцией **СРГЕОМ()**, которая находится в категории «Статистические» и вычисляет среднее геометрическое значений диапазона ячеек.

Создание диаграмм

Диаграмма представляет собой графическое представление данных таблицы. Диаграммы облегчают восприятие и интерпретацию данных. Они могут помочь, например, при анализе и сравнении результатов расчетов.







Электронные таблицы предлагают несколько видов диаграмм. Среди них есть наиболее часто используемые: гистограммы, круговые диаграммы (используются для иллюстрации значений одного столбца или одной строки таблицы), графики, XY-точечные (для представления графиков) – и совсем экзотические.

Ячейки, значения которых отображаются на диаграмме, называются *элементами данных*. Эти значения представляются на диаграмме в виде полос, линий, столбиков, секторов, точек и иных фигур, которые называются *маркерами данных*.

Задачи к разделу «Технология обработки информации в электронных таблицах MS Excel»

1 В электронной таблице формулу =**D2+E$1**, записанную в ячейке **F2**, скопировали в ячейку **G3**. Какой вид приобретет формула?

1) =**E2+F$1** 2) =**E3+F$2** 3) =**E3+F$1** 4) =**D2+E$1**

2 В электронной таблице формулу =**C2+$E$1**, записанную в ячейке **D2**, скопировали в ячейку G3. Какой вид приобретет формула?

1) =**F2+$I$1** 2) =**F3+$E$1** 3) =**F3+$I$2** 4) =**C2+$E$1**

3 В электронной таблице формулу =**B2+$D$1**, записанную в ячейке **E3**, скопировали в ячейку **F3**. Какой вид приобретет формула?

1) =**B2+$D$1** 2) =**C3+$E$2** 3) =**C2+$E$2** 4) =**C2+$D$1**

4 В электронной таблице формулу =**D2+$E3**, записанную в ячейке **G2**, скопировали в ячейку **F3**. Какой вид приобретет формула?

1) =**C2+$E3** 2) =**E2+$E3** 3) =**E2+$F4** 4) =**C3+$E4**

5 В электронной таблице формулу =**C2+$D3**, записанную в ячейке **E3**, скопировали в ячейку **F4**. Какой вид приобретет формула?

1) =**D3+$E3** 2) =**D3+$D3** 3) =**D3+$D4** 4) =**D3+$E4**

6 В электронной таблице формулу =**D2+$E3**, записанную в ячейке **G2**, скопировали в ячейку **G4**. Какой вид приобретет формула?

1) =**D2+$G4** 2) =**D4+$E5** 3) =**D4+$G5** 4) =**C3+$E4**

7 В электронной таблице формулу =**D4+$E5**, записанную в ячейке **G4**, скопировали в ячейку **E2**. Какой вид приобретет формула?

1) =**B2+$E3** 2) =**D2+$E3** 3) =**D4+$E3** 4) =**B2+$E5**

8 В электронной таблице формулу =**D$4+E$5**, записанную в ячейке **G4**, скопировали в ячейку **E2**. Какой вид приобретет формула?

1) =**D$4+C$5** 2) =**B$4+C$5** 3) =**D$2+E$2** 4) =**B$4+E$2**

9 В электронной таблице формулу **=$D$4+E5**, записанную в ячейке **G4**, скопировали в ячейку **E2**. Какой вид приобретет формула?

1) **=$D$4+C3** 2) **=$D$4+E3** 3) **=$D$2+E5** 4) **=$B$2+E5**

10 В электронной таблице формулу **=D4+E5**, записанную в ячейке **G4**, скопировали в ячейку **E3**. Какой вид приобретет формула?

1) **=E3+F5** 2) **=B2+C3** 3) **=B3+C4** 4) **=D4+C5**

11 На рисунке приведен фрагмент электронной таблицы.

	A	B	C	D	E	F
1	3	4	5	7	1	2
2	7	5	7	4	0	9
3	3	6	4	8	6	5
4	4	5	8	9	6	0

Основываясь на данных фрагмента таблицы, укажите, чему будет равно значение формулы

=(A3+B2)*C4*(F1-E1)

1) 84 2) 64 3) 123 4) 40

12 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы
=КОРЕНЬ($B$1-A1)+C4-E3/E4

1) 9,67 2) 7 3) 8 4) 6

13 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы
=C3*E1+B3+F2*$F$1

1) 28 2) 64 3) 60 4) 40

14 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы
=C3*(E1+F3)*($A4-A3)

1) 42 2) 24 3) -7 4) 6

15 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы **=$A$2+ (D1+C1)*E1+B1**, которую скопировали из ячейки **G1** в ячейку **G2**.

1) 8 2) 12 3) 15 4) 23

16 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы **=$A$2+ (D1+C1)*E1-B1**, которую переместили из ячейки **G1** в ячейку **G2**.

1) 8 2) 2 3) 15 4) 23

17 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы **=A1+ B1*E$1**, которую скопировали из ячейки **G2** в ячейку **H2**.

1) 7 2) 84 3) 15 4) 14

18 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы **=A1+ B$1*E1**, которую скопировали из ячейки **G1** в ячейку **H2**.

1) 50 2) 14 3) 12 4) 7

19 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы **=$A$1+ B1*E1**, которую переместили из ячейки **G1** в ячейку **G2**.

1) 11 2) 7 3) 13 4) 15

20 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы **=СУММ(A2:B2;E2:F2)**.

1) 21 2) 32 3) 16 4) 5

21 Основываясь на данных фрагмента таблицы к задаче 11, укажите, чему будет равно значение формулы **=СУММ($A$2:$B$2;E2:F2)**, которую скопировали из ячейки **G2** в ячейку **G3**.

1) 41 2) 18 3) 20 4) 23

22 В ячейку **A1** введена дата 01.01.2008. Чему будет равен результат вычисления формулы **=A1+30**?

23 В ячейку **F1** введена дата 20.04.2008. Чему будет равен результат вычисления формулы **=F1+21**?

24 В ячейку **D2** введена дата 15.03.08. Чему будет равен результат вычисления формулы =**D2+45**?

25 В ячейку **B5** введена дата 10.05.08. Чему будет равен результат вычисления формулы =**B5-20**?

26 В ячейку **C3** введена дата 15.03.08. Чему будет равен результат вычисления формулы =**C3-17**?

27 В ячейку **D4** введена дата 17.09.2008. Чему будет равен результат вычисления формулы =**D4-21**?

28 В ячейку **B2** введена дата 29.08.08. Чему будет равен результат вычисления формулы =**B2+46**?

29 В ячейку **B2** введено время 8:30, а в ячейку **C2** введено время 4:30. Чему будет равен результат вычисления формулы =**B2+C2**?

30 В одну ячейку ввели время 8:30, а в другую – время 5:45. Чему будет равна сумма значений этих ячеек?

31 В ячейку **A2** ввели время 13:15, а в ячейку **G2** – время 0:40. Чему будет равен результат вычисления формулы =**A2-G2**?

32 В ячейку **C1** ввели время 0:40. Чему будет равен результат вычисления формулы =**C1*2**?

33 В ячейку **C1** ввели время 2:35. Чему будет равен результат вычисления формулы =**C1*2**?

34 В ячейку **C1** ввели время 1:45. Чему будет равен результат вычисления формулы =**C1*2**?

35 Какую формулу нужно выбрать, чтобы определить максимальное из значений в диапазоне ячеек с **B3** по **B10**?

1) =**СУММ(B3;B10)**
2) =**МАКС(B3;B10)**
3) =**СЧЕТ(B3:B10)**
4) =**МАКС(B3:B10)**

36 Какую формулу нужно выбрать, чтобы определить минимальное из значений в диапазоне ячеек с **C5** по **C20**?

1) =**СУММ(C5;C20)** 3) =**МИН(C5:C20)**
2) =**МИН(C5;C20)** 4) = **СЧЕТ(C5:C20)**

37 Какую формулу нужно выбрать, чтобы определить среднее арифметическое значений в диапазоне ячеек с C5 по C20?

1) =**СРГЕОМ(C5;C20)** 3) =**СУММ(C5:C20)**
2) =**СРЗНАЧ(C5:C20)** 4) = **СУММЕСЛИ(C5:C20)**

38 Основываясь на данных фрагмента таблицы, укажите, чему будет равно значение формулы =**СУММ(B6:E6)**

	A	B	C	D	E
1		Сводка оценок по информатике			
2		10 “А”	10 “Б”	11 “А”	11 “Б”
3	Отлично	4	6	9	7
4	Хорошо	10	8	7	13
5	Удовлетворительно	8	13	6	6
6	Неудовлетворительно	2	1	3	0

1) 2 2) 6 3) 7 4) 4

39 Основываясь на данных фрагмента таблицы к задаче 38, укажите, какому классу будет соответствовать значение формулы =**МИН(B4:E4)**.

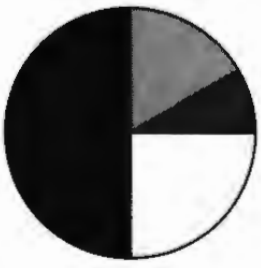
1) 10 “А” 2) 10 “Б” 3) 11 “А” 4) 11 “Б”

40 Основываясь на данных фрагмента таблицы к задаче 38, укажите, какому классу будет соответствовать значение формулы =**МАКС(B3:E3)**.

1) 10 “А” 2) 10 “Б” 3) 11 “А” 4) 11 “Б”

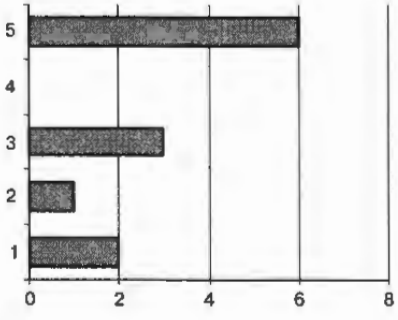
41 Основываясь на данных таблицы, определите, какая из диаграмм показывает распределение по классам количества учеников, получивших оценку «неудовлетворительно».

	A	B	C	D	E	F
1		Сводка оценок по информатике				
2		10 “А”	10 “Б”	11 “А”	11 “Б”	Итого
3	Отлично	4	6	9	7	26
4	Хорошо	10	8	7	13	38
5	Удовлетворительно	8	13	6	6	33
6	Неудовлетворительно	2	1	3	0	6

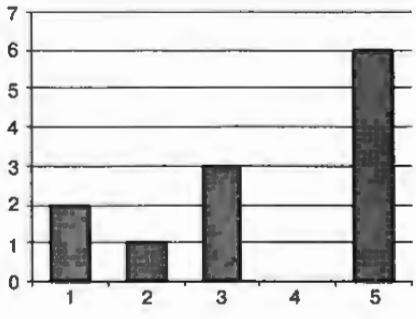


1)

2)

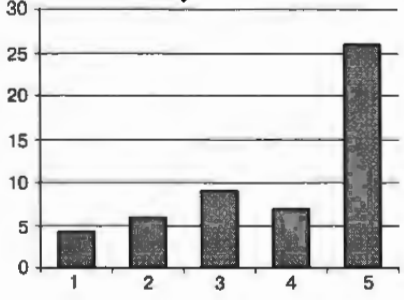


3)

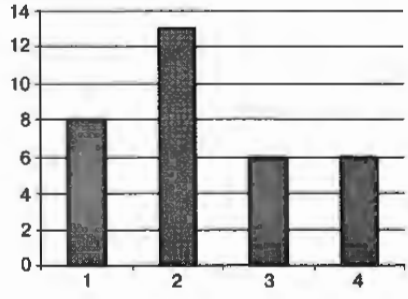


4)

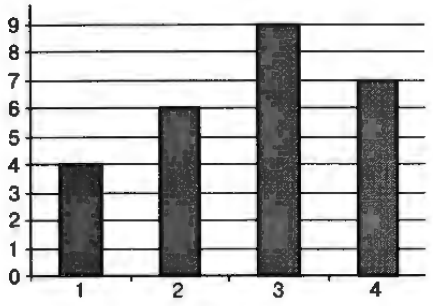
42 Основываясь на данных таблицы к задаче 41, определите, какая из гистограмм показывает распределение по классам количества учеников, получивших оценку «отлично».



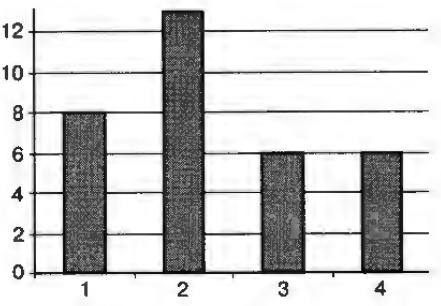
1)



2)

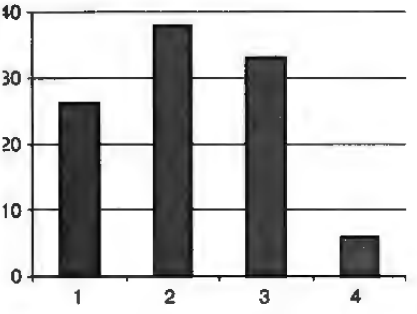


3)

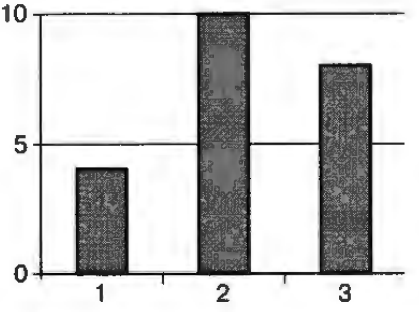


4)

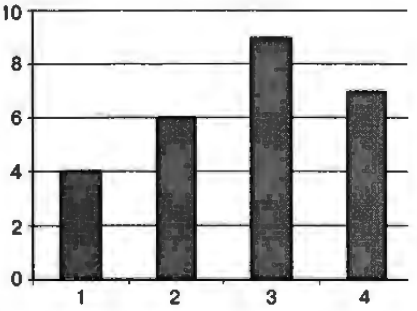
43 Основываясь на данных таблицы к задаче 41, определите, какая диаграмма и для какого класса отображает распределение полученных оценок.



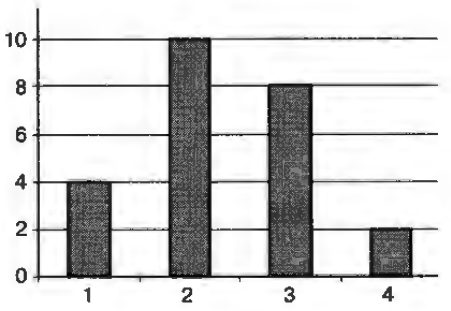
1)



2)



3)



4)

Технология хранения, поиска и сортировки информации в базах данных

База данных – разновидность информационной системы, в которой реализованы функции централизованного хранения и накопления обрабатываемой информации.

Базы данных реализуются с помощью *Систем управления базами данных* (*СУБД*) – наборов программ, позволяющих создать базу данных и обеспечить доступ к ней.

Хранимые в базе данные имеют определенную логическую структуру – *модель данных*.

К числу классических моделей данных относятся:

- иерархическая,
- сетевая,
- реляционная.

Иерархическая модель

Связи между данными можно описать с помощью упорядоченного графа (или дерева).

Достоинства: эффективное использование памяти, хорошие показатели по времени выполнения основных операций над данными.

Недостатки: громоздкость для обработки информации с достаточно сложными логическими связями, сложность понимания для обычного пользователя.

Сетевая модель

Связи между элементами данных отображаются в виде произвольного графа.

Достоинства: эффективное использование памяти, хорошие показатели по времени выполнения основных операций над данными.

Недостатки: высокая сложность схемы данных, сложность понимания для обычного пользователя.

Реляционная модель

Элементы данных представлены в виде таблиц. Является наиболее распространенной моделью.

Достоинства: простота, понятность и удобство реализации на ЭВМ.

Основным элементом реляционной модели данных является таблица. Каждая таблица имеет имя – идентификатор, по которому на нее можно сослаться. Столбцы таблицы соответствуют тем или иным характеристикам объектов и называются *полями*. Каждое поле характеризуется именем и типом хранящихся данных. Каждая строка таблицы соответствует одному из объектов. Она называется *записью* и содержит значения всех полей, характеризующих данный объект.

При построении таблиц базы данных важно обеспечить их целостность. Это делается введением *ключевых полей*, которые обеспечивают уникальность каждой записи. *Первичный ключ* – поле (или несколько полей) таблицы, однозначно идентифицирующие каждую запись. Если ключ состоит из одного поля, то он называет-

ся *простым*, а если из нескольких полей – *составным*. В таблице не должно быть двух и более записей с одинаковыми значениями ключа. Если условия задачи не дают возможности выбрать первичный ключ, удовлетворяющий этому условию, то допустимо добавление поля для организации первичного ключа.

База данных, как правило, состоит из нескольких таблиц, которые логически связаны между собой. Для этого две таблицы должны иметь набор одинаковых полей. *Внешним ключом* является одно или несколько полей таблицы (она называется *подчиненной*), которые в другой таблице (*главной*) являются первичным ключом.

По заданным полям таблицы может быть построен *индекс*. Он во многом сходен с первичным ключом, однако допускает повторение значений входящих в него полей. Индекс, как и первичный ключ, играет роль своеобразного оглавления таблицы, просмотр которого предшествует обращению к записям. Поля, входящие в индекс, называются *индексными полями*, а процесс создания индекса – *индексированием таблицы*. Индексы могут храниться в отдельных файлах или же вместе с данными.

Использование индексов позволяет сократить время поиска данных в таблице и, как следствие, длительность других операций, использующих поиск, например удаления и редактирования.

Выборка информации из базы данных проводится с помощью *запросов* – логических выражений, реализованных, как правило, на различных модификациях специального языка *SQL* (*Structure Query Language* – структурированный язык запросов).

Запросы позволяют проводить выборку, добавление, удаление и редактирование записей по заданным критериям. В критериях запроса можно формировать как простые логические выражения с помощью отношений (>, <, = и т. д.), так и составные с помощью логических операций (И, ИЛИ, НЕ и т. д.). Приоритет выполнения логических операций при вычислении значения выражения в условии совпадает с приоритетом, принятым в математической логике.

При отборе записей запросы могут содержать вычисляемые поля. Значения таких полей рассчитываются на основании значений полей таблицы, указанных в запросе.

Для удобства просмотра информации может применяться *сортировка* – расположение записей таблицы в порядке возрастания или убывания значений указанных полей.

При сортировке по нескольким полям важен указанный порядок полей. Предположим, что поле, указанное в списке полей сортировки первым, имеет несколько одинаковых значений. Тогда порядок записей будут определять значения поля, указанного в списке следующим. Например, пусть в таблице, содержащей поля «Фамилия», «Имя» и «Отчество», хранится информация о следующих людях: Иванов Сергей Михайлович, Иванов Антон Михайлович и Ива-

нов Сергей Алексеевич. База данных сортируется по возрастанию набора полей «Фамилия» + «Имя» + «Отчество». Тогда записи расположатся в порядке:

Иванов Антон Михайлович;
Иванов Сергей Алексеевич;
Иванов Сергей Михайлович.

Задачи к разделу «Технология хранения, поиска и сортировки информации в базах данных»

1 Приведен фрагмент таблицы базы данных, содержащей поля «Автор», «Название», «Год издания» и «Общее количество»:

	Автор	Название	Год издания	Общее количество
1	Пушкин	Капитанская дочка	2000	5
2	Перышкин	Физика	1990	25
3	Пушкин	Капитанская дочка	1989	7
4	Перышкин	Физика	1998	20
5	Пушкин	Дубровский	2004	6

Какие из полей для таблицы базы данных целесообразно выбрать в качестве первичного ключа?

1) Автор
2) «Автор» + «Общее количество»
3) «Автор» + «Название» + «Год издания»
4) «Автор» + «Название» + «Общее количество»

2 Фрагмент базы данных к задаче 1 отсортировали по полям «Автор», «Название», «Год издания». Укажите через запятую, в каком порядке будут располагаться записи.

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

3 Приведен фрагмент таблицы базы данных:

Фамилия	Математика	Рус. язык	Ин. язык
Андреева	4	3	5
Баранкин	4	4	4
Волин	5	5	5
Данилов	5	3	5
Иванова	3	5	4
Ломов	3	3	3

Сколько записей во фрагменте таблицы удовлетворяют условию **(«Математика = 4») или («Ин. язык = 4»)**?

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

4 Сколько записей во фрагменте таблицы к задаче 3 удовлетворяют условию **(«Математика» = 4 и «Рус. язык» = 3)**?

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

5 Сколько записей во фрагменте таблицы к задаче 3 удовлетворяют условию **(«Ин. язык» = 4 и «Рус. язык» = 3)**?

1) 1 2) 2 3) 3 4) ни одной

6 Какое из условий позволит отобрать в таблице к задаче 3 записи со значениями «Математика» не равна 5 и «Рус. язык» = 4?

1) **(«Математика» < 5) и («Рус. язык» = 4)**
2) **(«Математика» = 4) и («Рус. язык» = 4)**
3) **(«Математика» < 5) или («Рус. язык» = 4)**
4) **(«Математика» = 3) и («Рус. язык» = 4)**

7 В табличной форме приведен фрагмент таблицы базы данных:

Город	Население в 1979 г. (тыс. чел.)	Население в 2002 г. (тыс. чел.)	Население в 2006 г. (тыс. чел.)
Волгоград	926	1013	1025
Москва	8057	10358	10425
Нижний Новгород	1342	1311	1284
Ростов-на-Дону	925	1070	1055
Самара	1192	1158	1143

Какое из условий позволит отобрать в таблице города, в которых увеличивалась численность населения с 1979 по 2006 год?

1) **(«Население в 2002 г. (тыс. чел.)» > «Население в 1979 г. (тыс. чел.)»)**
2) **(«Население в 2006 г. (тыс. чел.)» > «Население в 2002 г. (тыс. чел.)»)**
3) **(«Население в 2002 г. (тыс. чел.)» > «Население в 1979 г. (тыс. чел.)») и («Население в 2006 г. (тыс. чел.)» > «Население в 2002 г. (тыс. чел.)»)**
4) **(«Население в 2002 г. (тыс. чел.)» < «Население в 1979 г. (тыс. чел.)») и («Население в 2006 г. (тыс. чел.)» < «Население в 2002 г. (тыс. чел.)»)**

8 Какое из условий позволит отобрать в таблице к задаче 7 города, в которых до 2002 года численность населения увеличивалась, а потом начала уменьшаться?

1) **(«Население в 2002 г. (тыс. чел.)» > «Население в 1979 г. (тыс. чел.)») и («Население в 2006 г. (тыс. чел.)» > «Население в 2002 г. (тыс. чел.)»)**
2) **(«Население в 2002 г. (тыс. чел.)» > «Население в 1979 г. (тыс. чел.)») и («Население в 2006 г. (тыс. чел.)» < «Население в 2002 г. (тыс. чел.)»)**
3) **(«Население в 2006 г. (тыс. чел.)» > «Население в 2002 г. (тыс. чел.)»)**
4) **(«Население в 2006 г. (тыс. чел.)» < «Население в 2002 г. (тыс. чел.)»)**

Телекоммуникационные технологии

Компьютерной вычислительной сетью называют несколько компьютеров и сетевых устройств, соединенных между собой какой-либо *средой передачи данных* (*media*) для обмена информацией между компьютерами. Под средой передачи данных понимается любая используемая физическая среда, способная переносить сигналы: кабельная (несколько разновидностей медных кабелей или разновидности оптико-волоконных кабелей) или беспроводная.

Различают:

- локальные вычислительные сети или ЛВС (LAN, local area network) – сети, имеющие географически небольшие размеры (комната, этаж здания, здание или несколько расположенных рядом зданий). В качестве среды передачи данных в ЛВС используют, как правило, кабель, однако в последнее время набирают популярность беспроводные сети. Близкое расположение компьютеров продиктовано физическими законами передачи сигналов по используемым в ЛВС кабелям или мощностью передатчика беспроводных сигналов. ЛВС могут объединять от нескольких единиц до нескольких сотен компьютеров. Простейшая ЛВС может состоять из двух ПК, связанных кабелем или беспроводными адаптерами;
- интерсети, или сетевые комплексы – две и более ЛВС, объединенные специальными устройствами для поддержки больших ЛВС. Являются, по сути, сетями сетей;
- глобальные сети – (WAN, Wide Area Network) ЛВС, соединенные средствами удаленной передачи данных;
- корпоративные сети – глобальные сети, находящиеся в ведении одной организации.

С точки зрения логической организации сети бывают *одноранговые* и *иерархические*.

В одноранговых сетях все компьютеры равноправны с точки зрения распределения и использования ресурсов сети. Примером одноранговой сети может служить рабочая группа в операционной системе Microsoft Windows. В таких сетях каждый пользователь отвечает за состояние и за доступность данных для других пользователей сети и не предусмотрено выделение компьютеров-серверов.

Сервером называют компьютер сети, который предоставляет свои ресурсы (данные или периферийные устройства) другим компьютерам сети. С этой точки зрения все компьютеры в одноранговой сети могут быть одновременно и серверами, и *клиентами*, т.е. компьютерами, пользующимися ресурсами сервера.

Кроме аппаратного понятия сервера существует также программное. *Сервером* называют часть программного обеспечения, реализующую запросы от клиентской части, с которой работает пользователь. *Программный клиент* реализует возможности организации запроса к серверу и отображение его результатов.

В *иерархических* сетях существуют специальные компьютеры-серверы, ресурсы которых могут быть доступны всем пользователям сети и за работой которых следит администратор сети.

Для работы компьютера в сети на нем должна быть установлена *сетевая операционная система* (*NOS, Network Operating System*). Причем различаются операционные системы для сервера и для рабочей станции.

Протоколы

Взаимодействие компьютеров в сети регламентируется протоколами, т.е. наборами формальных правил и кодировок, определяющих, как устройства в сети обмениваются данными. Эти протоколы описывают любой момент взаимодействия – от характеристик сигналов, передаваемых через среду передачи данных, до языков запросов, позволяющих обмениваться сообщениями приложениям, исполняемым на разных компьютерах. Компьютеры сети используют множество протоколов, которое называют *стеком* и которое охватывает всё от пользовательского интерфейса приложения до физического интерфейса сети.

Стек протоколов TCP/IP (*Transmission Control Protocol/ Internet Protocol*) разрабатывался специально для создания глобальной сети Интернет. Основным требованием к этому стеку было его умение общаться с различными ЛВС, работающими с разными стеками протоколов. Именно эта универсальность и простота настроек стали одними из главных причин, по которым TCP/IP стали применять и в ЛВС.

Среди протоколов, составляющих стек TCP/IP, непосвященному человеку наиболее известны два протокола – TCP и IP. Из них протокол IP организует в сети работу с логической адресацией, а протокол TCP относится к группе про-

токолов, занимающихся подготовкой данных к передаче. В ряде случаев вместо TCP используется другой протокол стека – *UDP (User Datagram Protocol)*. В ведении этих двух протоколов находится разделение всего объема данных на мелкие сегменты, чтобы предотвратить монопольный захват сети двумя рабочими станциями, тип передачи (надежный или ненадежный) и в случае надежной передачи восстановление полного объема данных из сегментов.

Протокол UDP является протоколом ненадежной передачи данных и в его задачу входит только подготовка сегментов данных для отправки в сеть. Поэтому он, как правило, используется для работы с мультимедийным трафиком (голос, видео).

Протокол TCP обеспечивает надежную доставку данных, требует подтверждения получения данных принимающей стороной, позволяет передающей и принимающей сторонам подобрать оптимальную скорость передачи данных, меняющуюся в зависимости от ситуации в сети (управление потоком) и имеет ряд других полезных функций. Используется для передачи файлов (FTP), писем электронной почты, в работе с WWW.

Протокол IP. Для идентификации компьютеров в TCP/IP применяется IP-адресация. *IP-адрес* однозначно определяет сеть и узел сети (сетевой интерфейс), куда должен быть доставлен пакет данных, – эта информация содержится в каждом пакете, который передается по сети. IP-адрес представляет собой двоичное число, имеет длину 32 бита и изображается в виде четырех десятичных чисел – октетов, или квадрантов, разделенных точкой. Десятичное значение октета может изменяться в пределах от 0 до 255.

IP-адрес состоит из двух частей: первая часть битов отвечает за идентификацию сети, а ставшаяся часть – за идентификацию узла (например, компьютера). Первая часть адреса требуется для передачи пакетов в сеть назначения: маршрутизаторы, которые обеспечивают передачу пакетов, вычисляют адрес сети и на его основании определяют, как доставить пакет в эту сеть, следовательно, у устройств в одной сети первая часть IP-адреса должна совпадать. Вторая часть назначается произвольно и отдельно не используется как таковая. После того как пакет попал в сеть назначения, для определения получателя в сети адрес обрабатывается целиком. За выделение частей адреса отвечает *маска подсети* (маска сети или сетевая маска). Маска подсети – это тоже 32-битное двоичное число, которое, аналогично IP-адресу, записывается четырьмя десятичными числами, разделенными точками. В маске запрещено чередование 0 и 1: значение ведущих битов всегда 1, а замыкающих – 0. Биты IP-адреса, которым соответствуют биты маски со значением 1, определяют адрес сети; остальные – адрес узла (хост-биты). От количества единиц в маске зависит количество устройств, которые можно адресовать в сети.

Операция выделения частей IP-адреса основана на логическом побитовом умножении IP-адреса на маску и иначе называется наложением маски на адрес.

Отсюда следует, что при умножении IP-адреса на маску левые биты IP-адреса, которым соответствуют единичные биты маски, сохранят свои значения, а оставшиеся биты умножаются на 0 и поэтому в результате получат значение 0. Следовательно, какие бы значения ни принимали хост-биты IP-адреса, при умножении их на маску произведение будет одинаковым; результат произведения и есть адрес сети. Из-за того что обработка проводится в двоичном виде, по десятичному представлению IP-адреса его части увидеть не всегда просто. Удобнее перевести адрес и маску в двоичный вид.

Пример 1.

Пусть есть IP-адрес 143.18.26.100 и этот адрес назначен устройству в некоторой сети с маской 255.255.252.0. Требуется определить адрес сети. Переведем заданные IP-адрес и маску в двоичный вид. Если для удобства чтения сохранить точки, разделяющие октеты, то двоичный вид IP-адреса будет выглядеть следующим образом:

10001111.00010010.00011010.01100100

Двоичный вид маски:

11111111.11111111.11111100.00000000

Выпишем их друг под другом и проведем операцию логического умножения:

10001111.00010010.00011010.01100100
11111111.11111111.11111100.00000000
10001111.00010010.00011000.00000000

В десятичном представлении результат произведения запишется как 143.18.26.0. Значит, наш адрес принадлежит IP-сети 143.18.26.0 с маской 255.255.252.0. Длина единичной части этой маски составлет 22 бита. Для сокращения записи маски может быть указано количество единичных битов через символ «/» после адреса: 143.18.26.100/22

Зная сетевую маску, можно легко посчитать, сколько в ней нулевых битов, а из этого значения определяется диапазон IP-адресов сети и количество адресуемых устройств. Следует помнить, что количество адресов в сети всегда на два больше возможного количества хостов. Количество адресов определяется соотношением $2n$, где n – количество нулевых битов маски; следовательно, возможное количество хостов – $2n$-2. Разница объясняется тем, что два адреса диапазона, а именно первый и последний, запрещено назначать устройствам сети. Первый из этих адресов является адресом сети и в этом адресе все хост-биты имеют значение 0. Последний адрес диапазона – адрес широковещательной рассылки (broadcast), в нем все хост-биты имеют значение 1. Этот вид рассылки используется в основном для служебных целей и означает, что пакет, отправленный с таким адресом назначения, получат *все* сетевые устройства данной сети, т.е. один компьютер отправляет в сеть один пакет данных, а принимают этот пакет и обрабатывают его все компьютеры и сетевые устройства.

Очевидно, что в пределах одной сети маска подсети обязана совпадать на всех узлах. В таком случае результат произведения любого адреса на маску в рамках одной сети одинаков; он называется IP-сетью. Только если у двух устройств сети совпадают адреса сети, эти устройства будут обмениваться данными. В противном случае, если адреса сетей не совпадают, то обмен данными будет возможен, только если между устройствами установлен *маршрутизатор*.

Пример 2.

Пусть в некоторой сети двум компьютерам назначили адреса 192.168.10.2 и 192.168.10. 65 с маской 26 бит (255.255.255.192). Почему при таких настройках адресов компьютеры друг друга «не видят»?

Определим адрес сети первого из компьютеров (192.168.10.2/26):

11000000.10101000.00001010.00000010
11111111.11111111.11111111.11000000
11000000.10101000.00001010.00000000

Следовательно, адрес сети в десятичном виде 192.168.10.0.

Теперь проведем такую же операцию с адресом второго компьютера (192.168.10.65/26):

11000000.10101000.00001010.01000001
11111111.11111111.11111111.11000000
11000000.10101000.00001010.01000000

Во втором случае адрес сети 192.168.10.64. Так как адреса сетей не совпадают, обмен данными между этими компьютерами возможен только при наличии маршрутизатора.

Интернет

Интернет – это глобальная телекоммуникационная сеть, объединяющая многие локальные, региональные и корпоративные сети.

С точки зрения реализации глобальные сети отличаются от локальных каналами связи, позволяющими передавать данные на значительно большие расстояния и, как следствие изменения каналов связи, протоколами, учитывающими особенности работы этих каналов. Объединение множества сетей стало возможно с появлением стека протоколов TCP/IP и IP-адресации. На стыках сетей могут стоять программные или аппаратные маршрутизаторы, реализующие принцип маршрутизации пакетов данных на основании IP-адреса назначения. Маршрутизатор принимает из непосредственно подключенной к нему сети пакет и автоматически пересылает его в другой непосредственно подключенный сегмент сети. Таким образом, после пересылки пакет может попасть или в сеть назначения, или на соседний маршрутизатор, который также перешлет его дальше.

Пакет данных в протоколе IP не учитывает каналы связи, т. е. является канально-независимым. Благодаря этой концепции, положенной в основу разработки протокола IP, появилась возможность передавать пакет по разным физическим каналам связи.

Однако в быту Интернет чаще всего ассоциируется не с самой физической сетью, а с набором служб, которые в нем реализованы. Самой известной из них является WWW («Всемирная паутина»), которую многие путают с самим Интернетом.

Службы (сервисы) Интернет

WWW (World Wide Web)

WWW («Всемирная паутина») – гипертекстовая информационная система поиска ресурсов Интернет и доступа к ним.

Гипертекст – информационная структура, позволяющая устанавливать смысловые связи между элементами текста на экране компьютера таким образом, чтобы можно было легко осуществлять переход от одного элемента к другому. Все содержимое системы WWW состоит из web-страниц, объединенных в сайты. Создаются они с помощью *языка разметки гипертекста HTML* (*HyperText Markup Language*). Для работы с WWW применяются программы-*браузеры* (например, Internet Explorer).

Найти страницу в Интернет или сделать на нее ссылку можно с помощью *универсального указателя ресурсов* (*адреса страницы*). Универсальный указатель ресурсов (*URL – Universal Resource Locator*) включает в себя способ доступа к документу, имя сервера, на котором находится документ, а также путь к файлу (документу).

Для доступа к документу требуется указать используемый протокол передачи данных и поставить цепочку символов "://". Далее указывается сервер и путь к документу. Например, для доступа к web-страницам используется протокол передачи гипертекста – *HTTP* (*HyperText Transfer Protocol*), входящий в стек протоколов TCP/IP:

http:// home.microsoft.com/intl/ru/

Доменная система имен

Понятие URL тесно связано с системой доменных имен.

Доменная система имен (*DNS – Domain Name System*) служит для организации однозначного соответствия IP-адреса и доменного имени. Вся работа в сети Интернет идет на основе IP-адресов, что представляет для большого количества пользователей серьезные неудобства. Обычному человеку проще общаться с буквенными строками, чем с абстрактным набором чисел IP-адреса. Эта возможность обеспечивается службой DNS. При обращении пользователя к ресурсу по его доменному имени следует запрос к DNS-серверу, который в ответ возвращает IP-адрес ресурса, и далее весь обмен данными уже идет по IP-адресу. Время отклика DNS-сервера на запрос очень мало и практически не влияет на скорость обращения к ресурсу.

Сама система представляет собой распределенную базу данных, хранящуюся на множестве DNS-серверов и периодически обновляемую. Поэтому при появлении нового ресурса с новым доменным именем он какое-то время может быть недоступен по доменному имени, но сразу доступен по IP-адресу.

Доменные имена регистрируются в Центре сетевой информации Интернет (InterNIC).

Крайняя справа группа букв в URL обозначает *домен верхнего уровня*. Домены верхнего уровня бывают двух типов: географические (двухбуквенные) и административные (трехбуквенные).

Административные		**Географические**	
com	Коммерческая организация	de	Германия
edu	Образовательная организация	ru	Россия
int	Международная организация	su	Бывший СССР
net	Сетевая организация	uk	Великобритания
org	Некоммерческая организация	us	США

Доменные имена второго уровня географического типа регистрируют национальные центры.

Электронная почта (e-mail)

Электронная почта (*electronic mail, E-mail*) – это одна из первых и наиболее распространенных услуг сети Интернет, обеспечивающая пересылку сообщений между пользователями сети (абонентами). Также может использоваться в качестве своеобразной базы данных документов, записей назначенных встреч, новостей и т. д.

Для использования этой услуги пользователь должен иметь *электронный адрес* и знать электронный адрес получателя.

Адрес электронной почты (*электронный адрес*) – адрес абонента, который может быть получен от:

- провайдера при заключении с ним договора, оговаривающего создание "почтового ящика";
- администратора сети организации, в которой работает пользователь или сформирован самостоятельно на любом из почтовых серверов, предлагающих эту услугу бесплатно.

Почтовый ящик – выделяемая пользователю часть дискового пространства на *почтовом сервере* для хранения сообщений электронной почты. Для доступа к своему почтовому ящику пользователь должен указать его имя и пароль.

Почтовый сервер – компьютер в сети, отправляющий, принимающий и хранящий сообщения электронной почты.

Электронный адрес состоит из двух частей, разделенных символом @ («коммерческое at»):

username@hostname.domain,

где *username* – это сетевой псевдоним пользователя (логин). В качестве него может использоваться одна или несколько цепочек символов, каждая из которых должна состоять из английских букв и арабских цифр. Цепочки разделяются символами «точка», «тире» и символом подчеркивания. После имени пользователя ставится разделитель @;

hostname – имя хоста (почтового сервера);

domain – доменный адрес.

Задачи к разделу «Телекоммуникационные технологии»

1 Укажите, что в адресе электронной почты **kot-matroskin@prostokvashino.net** является псевдонимом пользователя?
1) prostokvashino 2) kot-matroskin 3) net 4) kot

2 Укажите, что в адресе электронной почты **kot-matroskin@prostokvashino.net** является именем почтового сервера?
1) prostokvashino.net 2) matroskin 3) prostokvashino 4) net

3 В адресе электронной почты **Karlson@krysha.org** часть **Karlson** является:
1) доменом верхнего уровня;
2) сетевым псевдонимом пользователя;
3) именем почтового сервера;
4) паролем к почтовому ящику.

4 В адресе электронной почты **Krolik@nora.org** часть **org** является:
1) доменом верхнего уровня;
2) сетевым псевдонимом пользователя;
3) именем почтового сервера;
4) паролем к почтовому ящику.

5 В адресе электронной почты **Postoronnim_v@pyatachok.org** часть **Postoronnim_v** является:

1) именем почтового сервера;
2) доменом верхнего уровня;
3) паролем к почтовому ящику;
4) сетевым псевдонимом пользователя.

6 Какие из приведенных ниже цепочек символов могут быть правильным электронным адресом?
1) abc@abc.org
2) //abc@abc.org
3) @abc.net
4) org.abc@abc

7 Какие из приведенных ниже цепочек символов могут быть правильным электронным адресом?
1) abc+def@klm.ru
2) !tom@xyz.gb
3) abc.ru @Ivan
4) bill@lmn.net

8 Какие из приведенных ниже цепочек символов могут быть правильным электронным адресом?
1) abc@def@klm.ru
2) linda@xyz.com
3) abc/ru/ @Ivan
4) bill??@lmn.net

9 Что из перечисленного ниже является правильным IP-адресом?
1) 2.2.2.2
2) 22.22.22.22.22
3) 192.168.257.24
4) все правильные

10 Что из перечисленного ниже является неправильным IP-адресом?
1) 224.0.0.2
2) 11.12.22.32
3) 172.16.24.264
4) все правильные

11 Что из перечисленного ниже является неправильным IP-адресом?

1) 169,0,12,2
2) 1.12.23.34
3) 172.16.124.24
4) все правильные

12 Восстановите из отдельных частей URL:

А) :
Б) ftp.
В) rar
Г) http
Д) //narod.
Е) ru
Ж) /

13 Восстановите из отдельных частей URL:

А) raspisaniye
Б) ftp:
В) /
Г) //my_files.docs.
Д) .docx
Е) ru

14 Восстановите из отдельных частей URL:

А) catalog/
Б) www
В) midi.
Г) ru/
Д) mazurka
Е) .mid
Ж) .

15 Восстановите из отдельных частей URL:

А) http
Б) .microsoft
В) ww
Г) /log
Д) in.php
Е) .com
Ж) ://w

16 Восстановите из отдельных частей URL:

А) //my_first_site.
Б) nar
В) /raz
Г) htt
Д) noye/
Е) txt
Ж) p:
З) od.ru
И) document.

17 Восстановите из отдельных частей URL:

А) g/games
Б) tris.zip
В) /www
Г) /te
Д) .files.
Е) http:/
Ж) or

18 Восстановите из отдельных частей URL:

А) films/kin
Б) ftp:
В) .org
Г) //my_ftp
Д) /downloads/
Е) zaza.avi

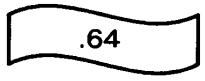
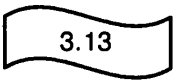
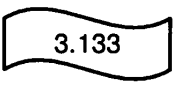
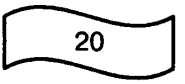
19 Восстановите из отдельных частей URL:

А) od.ru/
Б) chive
В) /page.
Г) www
Д) php
Е) .my_site.nar
Ж) news/ar

20 (*Демо-вариант, 2009 г.*)

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.

			
А	Б	В	Г

21 Сотруднику продиктовали по телефону IP-адрес: 215628319, он его записал, но не поставил разделительные точки. В ответе напишите этот IP адрес, поставив разделительные точки.

22 Сотруднику продиктовали по телефону IP-адрес: 25218327239, он его записал, но не поставил разделительные точки. В ответе напишите этот IP адрес, поставив разделительные точки.

23 Укажите число, которое не может быть использовано в IP адресе:
155, 271, 1, 205.

24 Укажите число, которое не может быть использовано в IP адресе:
231, 0, 217, 282.

25 Укажите число, которое не может быть использовано в IP адресе:
256, 10, 2, 0.

26 Сколько различных адресов может быть закодировано с помощью IP адреса?

А) 4 000 000 Б) $4*2^{30}$ В) 1024000 Д) 2^{31}

27 (*Демо-вариант 2008 г., задача В7*)
Доступ к файлу **http.txt,** находящемуся на сервере **mail.com**, осуществляется по протоколу **ftp**. В таблице фрагменты адреса файла закодированы буквами от А до Ж. Запишите последовательность этих букв, кодирующую адрес указанного файла в сети Интернет.

А	/
Б	.com
В	ftp
Г	mail
Д	.txt
Е	http
Ж	://

Решение. Адрес файла записывается следующим образом:

<протокол>:// <имя сервера>/ <имя файла>

Тогда искомая запись адреса:

ftp://mail.com/ http.txt

Ответ: ВЖГБАЕД.

28 (*Демо-вариант 2007 г., задача B7*)

Доступ к файлу **ftp.net**, находящемуся на сервере **txt.org** 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	.net
Б	ftp
В	://
Г	http
Д	/
Е	.org
Ж	txt

29 (*Демо-вариант 2006 г., задача B7*)

Доступ к файлу **www.txt**, находящемуся на сервере **ftp.net**, 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	.txt
Б	http
В	/
Г	://
Д	.net
Е	www
Ж	ftp

30 Запишите последовательность букв, кодирующую адрес сайта факультета ВМК МГУ.

А	ru	Домен России
Б	.	Разделительная точка
В	msu	Домен МГУ

Г	www	
Д	cmc	Домен факультета ВМК МГУ
Е	http	
Ж	://	

31 Запишите последовательность букв, кодирующую адрес сайта физического факультета МГУ.

А	ru	Домен России
Б	www	
В	http	
Г	msu	Домен МГУ
Д	.	Разделительная точка
Е	phys	Домен физического факультета МГУ
Ж	://	

32 Запишите последовательность букв, кодирующую адрес сайта министерства образования и науки.

А	ru	Домен России
Б	http	
В	mon	министерство образования и науки
Г	://	
Д	.	Разделительная точка
Е	www	
Ж	gov	Домен правительства РФ

33 Даны IP-адрес и маска: 213.121.29.100/28. Укажите адрес сети.

34 Даны IP-адрес и маска: 213.121.29.100/28. Укажите первый адрес в сети, допустимый для назначения интерфейсу.

35 Даны IP-адрес и маска: 213.121.29.100/28. Укажите последний адрес в сети, допустимый для назначения интерфейсу.

36 Даны IP-адрес и маска: 213.121.29.100/28. Укажите адрес широковещательной рассылки по сети.

37 Даны IP-адрес и маска: 213.121.29.100/23. Укажите адрес сети.

38 Даны IP-адрес и маска: 213.121.29.100/23. Укажите первый адрес в сети, допустимый для назначения интерфейсу.

39 Даны IP-адрес и маска: 213.121.29.100/23. Укажите последний адрес в сети, допустимый для назначения интерфейсу.

40 Даны IP-адрес и маска: 213.121.29.100/23. Укажите адрес широковещательной рассылки по сети.

41 Даны IP-адрес и маска: 193.11.200.120/25. Укажите адрес сети.

42 Даны IP-адрес и маска: 193.11.200.120/25. Укажите первый адрес в сети, допустимый для назначения интерфейсу.

43 Даны IP-адрес и маска: 193.11.200.120/25. Укажите последний адрес в сети, допустимый для назначения интерфейсу.

44 Даны IP-адрес и маска: 193.11.200.120/25. Укажите адрес широковещательной рассылки по сети.

45 Даны IP-адрес и маска: 144.200.31.80, 255.255.252.0. Укажите адрес сети.

46 Даны IP-адрес и маска: 144.200.31.80, 255.255.252.0. Укажите первый адрес в сети, допустимый для назначения интерфейсу.

47 Даны IP-адрес и маска: 144.200.31.80, 255.255.252.0. Укажите последний адрес в сети, допустимый для назначения интерфейсу.

48 Даны IP-адрес и маска: 144.200.31.80, 255.255.252.0. Укажите адрес широковещательной рассылки по сети.

49 Даны IP-адрес и маска: 150.255.97.200, 255.255.255.192. Укажите адрес сети.

50 Даны IP-адрес и маска: 150.255.97.200, 255.255.255.192. Укажите первый адрес в сети, допустимый для назначения интерфейсу.

51 Даны IP-адрес и маска: 150.255.97.200, 255.255.255.192. Укажите последний адрес в сети, допустимый для назначения интерфейсу.

52 Даны IP-адрес и маска: 150.255.97.200, 255.255.255.192. Укажите адрес широковещательной рассылки по сети.

53 Даны IP-адрес устройства 10.0.10.35 и IP-адрес сети 10.0.10.32. Укажите количество единиц маски.

54 Даны IP-адрес устройства 10.0.10.35 и IP-адрес сети 10.0.10.32. Укажите количество нулей маски.

55 Даны IP-адрес устройства 10.0.10.35 и IP-адрес сети 10.0.8.0. Укажите количество единиц маски.

56 Даны IP-адрес устройства 10.0.11.35 и IP-адрес сети 10.0.10.0. Укажите количество нулей маски.

57 Даны IP-адрес устройства 10.0.10.35 и IP-адрес сети 10.0.0.0. Укажите количество единиц маски.

58 Даны IP-адрес устройства 10.0.10.35 и IP-адрес сети 10.0.10.32. Укажите количество единиц маски.

59 Даны IP-адрес устройства 10.0.10.35 и IP-адрес сети 10.0.10.32. Укажите количество единиц маски.

60 К какому классу относится адрес 102.90.81.5?

61 К какому классу относится адрес 17.255.221.155?

62 К какому классу относится адрес 225.90.81.5?

63 К какому классу относится адрес 230.100.0.100?

64 К какому классу относится адрес 132.0.32.18?

65 К какому классу относится адрес 191.20.2.188?

66 К какому классу относится адрес 202.10.132.234?

67 К какому классу относится адрес 193.17.64.127?

Глава 6

Технология программирования

Ввод и вывод числовой информации. Выражения

Оператор ввода в Паскале – **Read**, вывода – **Write**. Версии этих операторов (точнее, это стандартные процедуры ввода-вывода) с буквами **ln** на конце после соответствующего действия переводят курсор на следующую строку: **Readln**, **Writeln**. В операторе ввода записывают через запятую имена переменных, значения которых надо ввести. (Мы рекомендуем не вводить с клавиатуры несколько переменных с помощью одного оператора, а использовать для каждого значения свой ввод.)

В операторе вывода в скобках допускается использование списка вывода любой длины. Элементы в нем разделяются запятой и могут быть разного типа (строки символов, которые записываются в кавычках, а также переменные, константы, выражения). Внимательно следите, чтобы элементы выводимой информации не сливались между собой: используйте пробелы и другие разделители, вовремя переходите на другую строку.

В сложных случаях для печати таблиц, а также чисел в столбик удобно пользоваться *форматным выводом*: для выводимого целого числа или символа надо указать, сколько позиций требуется под него отвести на экране, а для вещественного числа – еще и сколько цифр напечатать после запятой. Формат ставится через двоеточие непосредственно после каждого элемента вывода. Например:

Write('S=', S:8:2)

Здесь **S** – выводимая переменная, значение которой – вещественное число. Оно будет выведено с двумя знаками после запятой, а на все число на экране будет отведено 8 позиций.

В конце программы желательно переводить строку, чтобы результаты работы следующей программы печатались на другой строке. В Паскале это иногда делают с помощью **Readln**, тогда результаты можно будет видеть на экране, пока на клавиатуре не будет нажата клавиша ввода.

Выражения в языке Паскаль записываются в строчку, из-за этого часто возникают «дополнительные» скобки. Например, дробь $\frac{1}{2X}$ правильно записать 1/(2*X) или 1/2/X. Аргументы функций пишутся в скобках, например SIN(X).

Замечание. В этом разделе задачи с номером, помеченным звездочкой, относятся к числу задач повышенной сложности.

Задачи к разделу «Ввод и вывод числовой информации. Выражения»

1 Пользователь вводит длину стороны квадрата. Вывести на экран его площадь.

2 Пользователь вводит длины сторон прямоугольника. Вывести на экран его площадь.

3 Пользователь вводит два числа. Вывести на экран их среднее арифметическое.

4 Пользователь вводит два числа. Вывести на экран их среднее геометрическое.

5 Пользователь вводит длины катетов прямоугольного треугольника. Вывести на экран его гипотенузу и площадь.

6 Пользователь вводит значение угла в градусах. Вывести на экран его синус и косинус с точностью до 4-го знака после запятой. Сверить ответ с таблицами Брадиса.

Указание. В некоторых языках программирования аргументом функций *Sin* и *Cos* является угол в радианах, поэтому вам может понадобиться перевести вводимый угол из градусов в радианы, а только потом брать от него синус и косинус.

---

7 Пользователь вводит число N. Вывести на экран через запятую $(N\text{-}1)^2$, N^2 и $(N+1)^2$.

8 Для вводимых чисел A и B проверить справедливость формулы $(A+B)^4 = A^4+4A^3B+6A^2B^2+4AB^3+B^4$, т. е. вычислить по отдельности левую и правую части формулы и сравнить их.

9 Для вводимого угла α проверить справедливость формулы $\sin^2\alpha + \cos^2\alpha = 1$.

Указание. Поскольку на обычной клавиатуре нет греческих букв, для имени угла α можно использовать переменные *A*, *Alpha* или имеющие любое другое подходящее по смыслу имя.

10* Пользователь вводит коэффициенты a, b и c квадратного уравнения $ax^2+bx+c=0$. Вывести на экран корни этого уравнения.

Примечание

При написании программы можно считать, что корней у этого уравнения всегда два, и вводить такие коэффициенты, чтобы корней действительно было два или один (тогда считается, что корней два, но они совпали). Например, $a=2$, $b=8$, $c=8$; $a=1$, $b=0$, $c=-1$; $a=1$, $b=-3$, $c=2$.

Проверьте справедливость теоремы Виета для корней этого уравнения (а именно, выведите на экран $(x1+x2)a$ и $x1x2a$ и сравните эти числа со значениями переменной b и переменной c соответственно).

---

11 Поменять местами значения переменных A и B.

12* Поменять местами значения переменных A и B, не используя вспомогательную переменную.

Условный оператор

Условный оператор можно представить следующим образом:

If <условие> **Then** <группа операторов> **Else** <группа операторов>

Здесь <условие> – условное (логическое) выражение. Чаще всего в качестве условий фигурируют отношения (например, ***X* > 0**), которые бывают связаны между собой логическими операциями (**And**, **Or**). В Паскале в таком случае отношения надо брать в скобки;

<группа операторов> – один оператор или несколько. Если операторов больше одного, то в Паскале они заключаются в *операторные скобки* **Begin... End**. Напомним, что использовать **Goto** не рекомендуется, структура языка позволяет обходиться без таких переходов.

Часто применяется укороченный условный оператор – без второй части (**Else**).

Задачи к разделу «Условный оператор»

1 Пользователь вводит год. Вывести на экран ответ, является ли год високосным.

Указание. Обычно каждый год, который делится на 4, – это високосный год, кроме лет столетий (делящихся на 100). Но среди последних год, делящийся на 400, также считается високосным.

---

2 Пользователь вводит целое число. Если оно является четным и при этом не находится между 10 и 20, то вывести на экран его квадрат. В противном случае напечатать исходное число.

3 Пользователь вводит целое число. Программа должна ответить, четным или нечетным является это число.

4 Пользователь вводит целое число. Программа должна ответить, четным или нечетным является это число, делится ли оно на 3 и делится ли оно на 6.

5* Пользователь вводит коэффициенты *a, b* и *c* квадратного уравнения $ax2+bx+c=0$. Вывести на экран все корни этого уравнения или сообщение о том, что их нет. Отдельно рассмотреть случай, когда корни совпадают.

6 Пользователь вводит два числа. Если первое число меньше второго, то вычислить их сумму, иначе – разность первого и второго.

7 В компьютер вводится число (например, дальность выстрела). Если оно находится в интервале от 28 до 30, то напечатать текст ПОПАЛ, иначе – НЕ ПОПАЛ.

8 В компьютер вводится число (дальность выстрела). Если оно находится в интервале от 28 до 30, то напечатать текст ПОПАЛ; если оно больше или равно 30, то напечатать ПЕРЕЛЕТ; если оно больше 0, но меньше или равно 28, то НЕДОЛЕТ; если меньше или равно 0 – НЕ БЕЙ ПО СВОИМ.

9 **Калькулятор v.1.** Пользователь вводит число, операцию (сложение, вычитание, умножение или деление) и второе число. Распечатать результат.

Указание. Используйте для ввода чисел и операции три разных оператора **Readln**. Знак операции должен иметь тип **Char**.

Циклы

Циклы бывают трех видов: с предусловием, с постусловием и с известным числом повторов. В теле цикла может содержаться один или несколько операторов, которые и следует повторять.

В *цикле с предусловием*, как следует из его названия, сначала проверяется некоторое условие и в зависимости от его истинности либо выполняется тело цикла (после чего опять проверяется условие), либо тело пропускается и выполняется оператор, следующий за оператором цикла. Таким образом, возможна ситуация, когда тело такого цикла не выполнится ни разу. При использовании этого вида цикла надо следить, чтобы переменные, задействованные в условии, были определены (введены или им присвоены значения) до входа в цикл.

Цикл с предусловием реализуется оператором

While <условие> **Do** <оператор>

Обратите внимание, что в цикле **While** после **Do** допускается только один (!) оператор. Если в этом месте надо поставить несколько операторов, то надо превратить их в один, заключив в операторные скобки **Begin...End**.

В *цикле с постусловием* условие выписывается и проверяется после выполнения тела цикла. В зависимости от его истинности цикл либо повторяется еще раз, либо завершается. Такой цикл всегда выполняется хотя бы один раз, а переменные, задействованные в условии, могут определяться в теле цикла.

Для цикла с постусловием предназначен оператор

Repeat <операторы> **Until** <условие>

Конечно, эти циклы можно реализовать с помощью оператора перехода **Goto.** Но в Паскале использование этого оператора настолько не приветствуется, что такие примеры мы разбирать не будем.

Операторы цикла, в зависимости от условия, могут повторяться сколько угодно раз, в том числе и бесконечное количество раз, т.е. вызывать «зацикливание» программы. Как этого избежать?

При работе над алгоритмом следите, чтобы переменным, которые задействованы в <условии>, в <теле> присваивались новые значения. Например, если условие цикла (***X*<0**), а в теле цикла переменной ***X*** вообще нет, такой цикл если начнет выполняться, то никогда не закончится. Но обычно зацикливание происходит в более сложных ситуациях.

При наличии компьютера для борьбы с зацикливанием надо применять пошаговую отладку. Она может быть встроена в среду программирования, которой вы пользуетесь, или ее можно реализовать самостоятельно, вставив в тело цикла дополнительные (отладочные) операторы ввода и вывода (вывода – чтобы напечатать значения переменных, от которых может зависеть

зацикливание, ввода – чтобы приостановить на время выполнение программы, иначе при зациклившейся программе результаты будут бесконечно «бежать» по экрану, вы не успеете их разглядеть). Перед запуском на выполнение программы, в которой есть цикл, надо сохранить ее на диске, так как может создаться ситуация, при которой вам из-за зацикливания придется завершать работу с потерей программы.

Однако на ЕГЭ у вас не будет компьютера и никто не поможет вам не только исправить, а даже обнаружить зацикливание в вашей программе. Как быть? Спасает только прогон программы вручную с построением таблиц значений переменных на каждом шаге цикла. Зато при такой работе вы не только «поймаете» зацикливание, но и в большинстве случаев определите его причину. Программы с циклами мы настоятельно рекомендуем проверять особенно тщательно.

Цикл с известным числом повторений при правильном его использовании зациклиться не может. А правильное использование как раз и состоит в том, чтобы писать его только в тех случаях, когда количество повторений определено до начала цикла (например, если надо обработать все компоненты массива).

В общем виде оператор выглядит так:

For <переменная цикла>: = <выражение1>
To (или **downto**) <выражение2> **Do**
<1 оператор>

Повторим, что если вместо одного оператора надо поставить несколько, то используют операторные скобки.

Иногда из такого цикла требуется досрочно выйти (не понадобилось делать все шаги, решение уже известно). В Паскале для этого используется оператор **Break**.

Оператор **For** имеет несколько особенностей, которые надо знать, чтобы безошибочно им пользоваться.

1. Конечное значение переменной цикла, при котором его выполнение прекратится, вычисляется при входе в цикл, поэтому изменение значения <выражения2> в теле цикла не влечет за собой изменения числа повторений цикла. Например, цикл

```
N:=5;
For I:=1 To N Do
    N:=2;
```

выполнится 5 раз, а не 2.

2. В Паскале параметр цикла не может быть вещественным – только целым (либо символьным). Шаг в Паскале может быть равен только 1 или –1.

3. В Паскале менять значение переменной цикла в теле цикла нельзя. Последствия могут быть самыми разными: от игнорирования компьютером этих действий до зацикливания. После нормального выхода из цикла значение переменной цикла считается неопределенным (поскольку в зависимости от версии языка может быть разным).

Задачи к разделу «Циклы»

1 Распечатать первые 16 степеней числа 2.

Указание. Для хранения переменной, являющейся степенью двойки, используйте тип **Longint**.

---

2 Распечатать первые 20 чисел Фибоначчи.

Примечание. Числа Фибоначчи получаются следующим образом: первое и второе числа последовательности равны единице, а каждое последующее число равно сумме двух предыдущих. Таким образом, $F1=F2=1$, $F3=F1+F2=1+1=2$, $F4=F2+F3=1+2=3$, $F5=F4+F3=2+3=5$ и т. д.

---

3 Пользователь вводит 10 целых чисел. Подсчитать, сколько среди них четных чисел.

4 Пользователь вводит 10 целых чисел. Подсчитать, сколько среди них положительных, сколько отрицательных чисел и сколько нулей.

5 Пользователь вводит целые числа, заканчивая ввод числом 0. Подсчитать, сколько среди них нечетных чисел, но при этом не делящихся на 5.

Указание. Если точное число проходов цикла еще не определено в момент начала его выполнения, то удобнее использовать цикл **Repeat** или **While**.

---

6 Пользователь вводит целые числа, заканчивая ввод числом 0. Подсчитать количество и произведение тех из них, которые больше 0 и меньше 10.

7 Найти разницу между наибольшим и наименьшим числами из введенных пользователем.

8 Пользователь вводит с клавиатуры целые положительные числа (их количество не ограничено). Остановить его, сказав «Достаточно», как только сумма введенных чисел станет больше 100. Вывести, сколько чисел введено.

9 Пользователь вводит с клавиатуры символы (их количество не ограничено). Прекратить работу, как только он в сумме введет 5 раз символ *«F»*.

10* Условие то же, что и в предыдущей задаче, но прекратить работу в том случае, когда пользователь введет символ *«F»* 5 раз подряд.

11 «Нарисовать» на экране прямоугольник размера *M*N* (значения вводятся с клавиатуры), т.е. на экране должно быть *M* строк по *N* звездочек в каждой.

Указание. Обратите внимание: если циклы вложены друг в друга, то переменные цикла должны быть различными; если же циклы следуют друг за другом, то можно пользоваться одной и той же переменной для обоих циклов.

---

12 «Нарисовать» на экране прямоугольный равносторонний треугольник с катетом *N* (значение вводится с клавиатуры):

А) прямой угол «смотрит» влево вверх;

Б) прямой угол «смотрит» влево вниз;

В) прямой угол «смотрит» вправо вверх;

Примеры рисунков для *N*=3:

А)	Б)	В)
***	*	***
**	**	**
*	***	*

13 Вывести на экран все двузначные числа по 10 в каждой строке.

14 Посчитать сумму и произведение цифр введенного целого положительного числа *N*. Учесть значения $N \leqslant 1\,000\,000$.

Массивы

Массив – некоторое количество элементов, которое может храниться в памяти компьютера на протяжении работы программы. Все элементы массива имеют общее имя (имя массива) и номер в массиве (он записывается справа от имени в скобках).

Массивы обязательно надо описывать (объявлять) в начале программы, чтобы под них была отведена память. Например массив из 20 чисел в Паскале может быть описан так:

Var M: Array [1..20] Of Real;

Работают с массивами обычно поэлементно, как правило, используя оператор цикла **For,** так как ни ввести весь массив целиком, ни напечатать нельзя – надо это делать с каждым элементом по отдельности.

При работе с массивами надо следить, чтобы не выйти за границы массива (например, в массиве из 5 элементов нельзя обращаться к 7-му). В некоторых случаях система программирования отслеживает такие ошибки (например, Турбо Паскаль можно специальным образом настроить) и информирует пользователя о них. В большинстве же случаев компьютер при таком обращении работает с некоторыми данными из памяти, никакого отношения к вашему массиву не имеющими. Программа с такой ошибкой может давать временами правильные, а временами неправильные результаты. На ЕГЭ эта оплошность всегда будет считаться ошибкой.

В реальных, встречающихся в жизни задачах программист обычно сам решает, нужен ли ему для хранения данных массив или какая-то иная структура. В задачах, предлагаемых на ЕГЭ, в большинстве случаев явно указывается, как задаются и хранятся данные. Поэтому если таких указаний нет, то ученик должен пользоваться массивом только в том случае, если без хранения данных в массиве задачу решить невозможно. Если же в задаче явно указано, что данные хранятся в массиве, то ее надо решать, обязательно используя массив (и, наоборот, если последовательность данных вводится и обрабатывается, но нигде не хранится, надо решать задачу без использования массива).

Большинство задач, предлагаемых ниже, можно решить и без использования массива; они приведены здесь для изучения методов работы с этой структурой данных. Заметим также, что в большинстве случаев перед решением задачи, поставленной в условии, массив надо ввести. В Паскале часто бывает удобно оформить это действие в виде процедуры.

Указание. Во всех задачах в данном разделе, если иное не указано в явном виде, следует считать массив (массивы) в условии задачи состоящим из 10 целочисленных элементов.

Задачи к разделу «Массивы»

1 Пользователь вводит массив A из 10 целых чисел. Сформировать массив B, в котором все элементы следуют в обратном порядке (первый элемент в B является последним в A и т. д.).

Замечание. Эта задача существенно отличается от задания «Напечатать элементы массива в обратном порядке», так как для выполнения последнего достаточно лишь цикла с заголовком **for J:=N downto 1 do**. Позиции элементов массива при этом не меняются. В данной же задаче надо именно сформировать новый массив, чтобы при прямой печати они выводились в обратном порядке.

---

2 Подсчитать разницу между суммой всех четных и суммой всех нечетных элементов массива целых чисел.

3 Подсчитать разницу между суммой всех элементов с четными индексами и суммой всех элементов с нечетными индексами.

4 Проверить, является ли введенный массив симметричным (т. е. первый элемент массива равен последнему, второй – предпоследнему и т. д.).

Указание. При решении подобных задач, подразумевающих ответ «ДА» или «НЕТ», в Паскале удобно использовать переменные логического типа. Можно также применять целые переменные, принимающие два значения: 0 (событие не произошло, правило не выполнено, ответ «нет» и т. п.) и 1 (событие случилось, ответ «да» и т. п.). Такие переменные в программировании называются «флагами» (потому что имеют два значения – флаг может быть «поднят» или «опущен»).

---

5 Выяснить, все ли элементы введенного пользователем массива, различны.

Указание. Если вы сравнили 2-й элемент массива с 3-м, то после этого сравнивать 3-й со 2-м уже не нужно. За нерациональный алгоритм решения задачи может быть снижено количество баллов на ЕГЭ.

---

6 Подсчитать среднее арифметическое всех отрицательных элементов целочисленного массива.

7 Подсчитать количество чисел, равных максимальному элементу массива.

8 Определить номера первого и последнего элементов, которые совпадают с максимальным элементом массива.

9 Определить количество элементов массива, которые больше суммы своих «соседей» слева и справа (если элемент – первый или последний, то сравнивать его надо, соответственно, только с правым или с левым «соседом»).

10* Определить, сколько элементов одного знака находится в конце целочисленного массива и какого они знака.

Примечание. Считается, что число 0 не имеет знака, следовательно, оно не может быть одного знака ни с положительными, ни с отрицательными числами. В случае, если последним элементом массива является число 0, надо вывести соответствующее сообщение.

---

11 Пусть в массиве из 12 элементов хранится количество осадков в каждом месяце. Требуется напечатать таблицу из 3 столбцов, где для каждого месяца вывести его номер, количество осадков в нем и отклонение количества осадков в этом месяце от среднегодового.

Указание. Это классическая задача на использование массивов. Ее нельзя решить без хранения данных в массиве, так как придется просматривать их два раза: чтобы найти среднегодовое значение и чтобы впечатать в таблицу отклонение.

---

12 Является ли введенное с клавиатуры слово палиндромом (так называются слова, которые слева направо и справа налево читаются одинаково, например «шалаш», «доход»)? Слово может состоять из любого (разумного) количества букв и заканчивается точкой.

13 С клавиатуры вводится основание системы счисления *Sis* (*Sis* < 10) и некоторое число в этой системе счисления. Перевести введенное число в десятичную систему счисления.

Примечание. Надо аккуратно вводить число в заданной системе счисления, учитывая, что цифры должны быть меньше основания системы счисления.

---

14 Задано число (в десятичной системе счисления) и число Q ($1 < Q < 10$) – основание системы счисления. Напечатать заданное десятичное число в Q-ичной системе счисления.

15 Задан массив вещественных чисел. Создать из его элементов два массива, записав в первый положительные числа, а во второй – отрицательные (в том же порядке).

Указание. Так как мы не знаем количества положительных и отрицательных чисел, каждый из массивов придется определять того же размера, что и заданный, а вот использовать в каждом массиве надо будет только нужное количество первых позиций (в «хвосте» будет «мусор»). Это надо учитывать при распечатке созданных массивов: печатать их не полностью, а только значимую их часть.

---

16 Произвести циклический сдвиг массива на одну позицию влево (т. е. второй элемент переставить на первое место, третий – на второе и т. д., а первый элемент – на освободившееся последнее место).

17 Пусть номер некоторого билета записан в массив (каждая цифра – один элемент массива). Является ли билет счастливым (т.е. сумма элементов первой половины массива равна сумме элементов второй половины)?

Строки

В Паскале строковый тип данных **string** – это массив символов (**char**). Длина строки может быть от 0 до 255 символов. При этом к строке можно обращаться поэлементно, как к массиву, т.е. если *S* имеет тип данных **string**, то мы вправе написать *S*[5] – это будет 5-й символ строки.

В Паскале определены стандартные процедуры и функции для работы со строками. Задачи, представленные в этом разделе, в первую очередь на применение этих процедур и функций, а также на операции над перечислимыми типами данных.

Напомним, что *перечислимыми типами данных* являются те, элементы которых можно пронумеровать, поставив им в соответствие конечную последовательность целых чисел. Ярким примером перечислимого типа данных может служить тип **char**, поскольку все его элементы имеют свой порядковый номер – от 0 до 255. В свою очередь, тип данных **string** не является перечислимым (его элементы не имеют строгого порядка), поэтому над переменными типа **string** невозможны операции **ord** (взятие порядкового номера), **pred** (предыдущий элемент), **succ** (последующий элемент) и некоторые другие.

Задачи к разделу «Строки»

1 Подсчитать количество заглавных и строчных латинских букв “A” и “a” в тексте, введенном пользователем с клавиатуры.

2 Подсчитать количество арабских цифр в тексте, введенном пользователем с клавиатуры.

Указание. Чтобы избежать сложных логических конструкций при решении задач, в которых требуется проверить вхождение одного элемента в некое достаточно большое множество (например, в множество арабских цифр, состоящее из 10 элементов), мы рекомендуем использовать в Паскале операцию **in** (вхождение во

множество) и тип-диапазон, который можно указать для перечислимых типов данных. Если же эти конструкции языка являются для вас незнакомыми, то при решении таких задач вы можете пользоваться условным оператором, но код программы получится более громоздким.

---

3 Подсчитать количество заглавных и строчных латинских букв в тексте, введенном пользователем.

4 Убрать из строки, введенной пользователем, все пробелы и распечатать ее.

5 Убрать из введенной пользователем строки на латинице все «посторонние» символы (т.е. все символы, кроме заглавных и строчных латинских букв, пробелов и всех знаков препинания). Распечатать полученную строку.

6 Во введенном пользователем тексте сделать все заглавные латинские буквы строчными.

7 Распечатать таблицу ASCII в виде таблицы символов из 16 строк и 16 столбцов.

Примечание. При выводе на экран некоторых первых служебных символов динамик компьютера может издавать характерный писк.

---

8 Распечатать первое слово из строки, введенной пользователем с клавиатуры.

Примечание. В этой и последующей задачах *словом* мы будем называть последовательность символов, отделенную от остального текста любым количеством пробелов.

---

9 Распечатать последнее слово из строки, введенной пользователем с клавиатуры.

Указание. Решить эту задачу в Паскале можно двумя способами: в цикле идти с начала в конец, выискивая последний пробел в тексте, либо идти с конца в начало, выискивая первый пробел. Сложность первого способа в том, что если пробелы встретятся в самом конце текста, то последнее слово мы пропустим, поэтому просмотренные слова надо запоминать. Сложность второго состоит в том, что функцию **pos** при таком направлении цикла применить не удастся. В конце книги мы приводим второй из этих способов, сочтя его более простым в реализации.

---

10 Подсчитать количество слов в строке, введенной пользователем.

Примечание. Эта задача несколько сложнее предыдущих, и решать ее можно самыми разными способами. Далее в этой книге задача встретится вам еще раз с предложением решить ее с помощью рекурсии. В конце книги мы приводим оба решения под соответствующими номерами, и вы можете сравнить, насколько непохожими могут быть решения одной и той же задачи.

---

11 В строке, введенной пользователем, удвоить все заглавные латинские буквы (например, A заменить на AA).

Примечание. В этой задаче нужно именно изменить исходную строку, а не сформировать другую.

Указание. Цикл **For I:=1 to Length(S)** для решения этой задачи применять нельзя по следующим причинам. В процессе выполнения цикла мы будем выполнять вставку символов в строку, тем самым увеличивая ее длину. Следовательно, **Length(S)** будет меняться. Компилятор может работать по-разному: какие-то версии компилятора вычисляют границы выполнения цикла один раз при первом заходе в него, какие-то вычисляют их при каждом проходе цикла (именно такая работа компилятора нам была бы нужна, если бы мы использовали цикл **For**), какие-то являются настраиваемыми в этом смысле. Чтобы люди, проверяющие на ЕГЭ ваши работы, не задавались вопросом, какой компилятор языка вы имеете в виду в программе, написанной на листочке, не используйте динамические границы в цикле **For**, а также не изменяйте самостоятельно переменную цикла внутри самого цикла (именно так предусмотрено в стандарте языка Паскаль).

Данная задача легко решается с применением цикла **While** или **Repeat**.

Файлы

Работа с файлом в Паскале ведется при помощи *файловой переменной*, которая описывается как **file of** <тип компонент файла> или **text** (что, по сути, является файлом из строк). Чтобы ассоциировать файловую переменную с именем реаль-

ного файла на диске, используется оператор **Assign**. Обычно он пишется один раз в начале программы.

Существует два основных режима работы с файлами: режим записи (файл открывается с помощью оператора **ReWrite**) и режим чтения (открывается с помощью **Reset**). По окончании работы файл надо закрыть (это обязательное действие для полного сохранения информации, если файл был открыт в режиме записи). Эти процедуры могут быть применены к одному файлу несколько раз, при этом работа с файлом будет всегда начинаться с начала, а открытие для записи непустого файла приводит к удалению в нем всей информации. Одновременно писать и читать из одного и того же файла невозможно!

В файле всегда доступна только одна позиция (при открытии файла – начальная), с которой может вестись работа (чтение или запись). Чтобы прочитать, например, последний элемент, придется прочитать весь файл. Назад вернуться нельзя, если несколько компонент уже прочитано: чтобы опять прочитать первую, надо снова открыть файл.

В Паскале есть два способа работы с файлами. Одного из них (при помощи буферной переменной) мы здесь касаться не будем, так как он не реализован в системе Турбо-Паскаль.

Для работы с файлами мы будем использовать стандартные процедуры ввода-вывода **Read** и **Write**. Процедуры **Readln** и **Writeln** можно применять только для файлов типа **text**. Обратите внимание на то, что вводить и просматривать информацию в файле типа **text** можно с помощью Паскаль-редактора или блокнота (но не MS Word!). Файлы всех остальных типов можно заполнять и просматривать только с помощью соответствующих программ, так как при записи информации в них применяется другая кодировка.

Полезной функцией для работы с файлами является логическая функция **EoF**, которая равна ИСТИНЕ, если достигнут конец файла. Для файлов **text** можно применять функцию **EoLn**, которая равна ИСТИНЕ, если достигнут конец строки.

Задачи к разделу «Файлы»

1 Записать с клавиатуры числа в файл.

Указание. Здесь очень важно в конце работы не забыть закрыть файл. Информация, записанная в файл, сохранится после работы программы (в отличие от массива), ее не испортит завершение работы компьютера и т. п. С помощью этой программы (поменяв типы данных) можно подготавливать файлы для других задач.

---

2 Напечатать на экране содержимое файла.

3 Сравнить два файла (будем считать два файла равными, если в них одинаковое количество элементов и соответствующие элементы равны между собой).

4 Из файла вещественных чисел вывести на экран числа, стоящие на четных местах.

5 Вывести на экран из текстового файла строку с заданным номером.

6 В файле **Famil.txt** хранятся фамилии, по одной на каждой строке. Приписать к каждой фамилии имя.

Процедуры и функции

Как определить, что писать – процедуру или функцию? Какие переменные будут являться параметрами процедуры, а какие – локальными, внутренними? Совет такой. Выпишите, что в вашей задаче дано, а что надо найти (как это обычно пишется в задачах по физике или геометрии). Если найти надо что-то одно, то используется функция, причем ее типом будет тип этого значения. Если же найти не надо ничего (бывает и такое, если надо что-то, например, напечатать) или таких параметров много, то используется процедура. Все выписанные параметры будут параметрами процедуры (функции), причем перед параметрами, которые надо найти, необходимо поставить **Var.** Перед параметрами «дано» **Var** ставить необязательно и даже нельзя, если при вызове процедуры (функции) на их месте удобно будет писать не переменные, а значения или выражения. Так как массивы можно передавать только как переменные, перед именем массива часто пишут **Var**, даже если его не надо найти, для экономии памяти и времени выполнения программы. Если параметром процедуры является массив, файл или какое-то другое значение сложного типа, то этот тип надо описать заранее и дать ему имя с помощью **type**.

В теле процедур (функций) можно использовать другие, описанные ранее процедуры и функции.

В теле функции обязательно должен быть хотя бы один оператор присваивания, где в левой части стоит имя функции (т.е. в результате работы функции ей должно быть что-нибудь присвоено). В правой части этого оператора присваивания или в условии имя функции, которая описывается в настоящий момент, стоять не должно, – это будет воспринято как рекурсивный вызов.

Задачи к разделу «Процедуры и функции»

1 Вектор в пространстве задается своими координатами (*X*,*Y*,*Z*), которые хранятся в трехмерном массиве. Написать процедуры и функции для работы с векторами:

1) ввод координат вектора с клавиатуры и вывод на экран;
2) нахождение суммы двух векторов;
3) нахождение длины вектора;
4) нахождение скалярного произведения двух векторов;
5) нахождение угла между двумя векторами.

2 Если целое число очень велико, то его нельзя представить в памяти компьютера с помощью типа **Integer** (или даже **Longint**): оно просто не помещается в отведенные ему ячейки. В таком случае число можно представить в виде массива его цифр. Пусть в числах не более 20 цифр, при этом последняя цифра записывается на 20-е место в массиве, а остальные – на соответствующие рядом. Если в числе менее 20 цифр, то незанятые первые позиции в массиве будем дополнять нулями. Написать процедуры и функции для работы с числами, представленными в таком виде:

1) ввод числа (недостающие позиции дополняются нулями);
2) вывод числа (незначащие нули не печатаются);
3) определение, является ли число четным;
4) определение, делится ли число на 3;
5) подсчет суммы двух чисел.

3 Пользователь вводит с клавиатуры целое число *N*. Рекурсивно найти *N*! (факториал числа *N*).

Примечание. *Рекурсивной* называется процедура (функция), в теле которой имеется обращение к самой себе.

Задания, представленные в этой и в последующих задачах этого раздела, можно найти в других разделах, где они решены не рекурсивно.

Следует помнить, что хотя никакого «цикла» мы в рекурсивном решении не записываем, оно является одним из способов его организации. Все операторы, которые вы запишете в теле рекурсивной функции, могут выполниться несколько раз (именно

поэтому, например, нельзя в теле рекурсивной функции писать оператор открытия файла **Reset**: он будет выполняться при каждом обращении, и чтение файла все время будет начинаться с начала).

Для прекращения рекурсии в рекурсивной функции (процедуре) обязательно должна быть *база* – нерекурсивная ветвь, где дается решение задачи для какого-то начального случая.

---

4 Рекурсивно вычислить сумму цифр целого числа.

5 С клавиатуры вводится слово (признак конца ввода – точка). Напечатать его «наоборот».

Примечание. Заметим, что без рекурсии эту задачу можно решить, только применяя какую-то структуру данных для хранения введенной последовательности (например, массив).

---

6 Используя рекурсию, распечатать первые N чисел Фибоначчи.

Примечания

1. Описание алгоритма получения чисел Фибоначчи см. в задаче 2 раздела «Циклы».

2. Далеко не всегда алгоритм решения задачи при помощи рекурсии так же эффективен, как алгоритм, решающий ту же задачу при помощи цикла, и наоборот, некоторые алгоритмы, реализованные при помощи рекурсии, намного эффективней алгоритмов, решающих ту же задачу при помощи цикла. Тем не менее любая задача, решенная с использованием рекурсии, может быть решена при помощи цикла **While** (доказательство данного утверждения лежит за пределами школьного курса информатики и за рамками данной книги).
После написания этой задачи запустите поочередно ее и программу задачи 2 раздела «Циклы» на компьютере и сравните скорость расчета первых 40 чисел Фибоначчи в обоих случаях.

---

7 Рекурсивно вычислить сумму целых чисел, если эти числа:

1) вводятся с клавиатуры (признак конца последовательности – 0);
2) находятся в файле;
3) находятся в массиве (с индексами от 1 до N).

8* Написать процедуры и функции для работы со строками, не используя аналогичные стандартные процедуры и функции языка:

1) функцию **MyLength(var S: string): Byte** (длина строки *S*);
2) функцию MyConcat(var S1, S2: string): string (конкатенация, т. е. сцепление двух строк);
3) функцию **MyCopy(var S: string; N, M: Byte): string** (подстрока длиной *M* символов строки *S*, начиная с *N*-го символа);
4) процедуру **MyInsert(var S1, S: string; N: Byte)** (вставка подстроки *S1* в строку *S* начиная с *N*-го символа).

Указание. При написании процедур и функций для работы со строками необходимо использовать тот факт, что строка в Паскале – это массив символов, причем в нулевом элементе этой строки хранится символ с кодом длины этой строки.

Примечания

1. Необходимо предусмотреть граничные случаи: когда строки пустые; когда параметры процедур или функций превышают длину строк.

2. В конце книги мы приводим решения, использующие дополнительные локальные строковые переменные. Их применение существенно облегчает программный код, но дополнительно загружает память компьютера. Тем не менее, поскольку на ЕГЭ перед вами не будет стоять задача написать наиболее экономичную по ресурсам программу, такие решения вполне приемлемы.

Смешанные задачи

1* Рекурсивно подсчитать количество слов в тексте, введенном пользователем с клавиатуры.

Примечание. В этой задаче, как и в задачах из раздела «Строки», *словом* мы будем называть последовательность символов, отделенную от остального текста любым количеством пробелов.

---

2 Вводятся 3 числа. Определить, можно ли построить треугольник с такими сторонами. Если да, то определить его вид: равносторонний, равнобедренный, прямоугольный, тупоугольный, остроугольный.

3 В файле записаны слова по одному в каждой строке. Напечатать все слова максимальной длины.

4* В файле записаны слова по одному в каждой строке. Выписать сначала все слова из одной буквы, потом из двух и т. п., пока не будут выписаны все слова.

5* В массиве записаны числа. А) Определить, сколько в нем разных чисел. Б) Для каждого входящего в массив числа напечатать, сколько раз оно в него входит (если число встречается в массиве несколько раз, то печатать его надо ровно один раз).

6 Написать программу для следующей игры человека с компьютером. Человек загадывает некоторое число из диапазона $[N,M]$, а компьютер должен его отгадать за минимальное количество вопросов типа «да-нет» («Ваше число больше 25?», «Ваше число меньше 70?» и т.д.).

7 Дано целое число от 20 до 99. Вывести словесное обозначение этого числа. (Например, 45 – «сорок пять»).

8 Задана логическая функция $F(a,b) = \neg a \vee b$. Напечатать таблицу истинности $F(a,b)$.

9 Дано натуральное число N. Среди чисел $1 \ldots N$ найти все такие числа, запись которых совпадает с последними цифрами записи их квадрата (пример: $6^2=36$).

10 Напечатать все четырехзначные натуральные числа, в записи которых нет одинаковых цифр.

11 Создать двумерный массив (матрицу) чисел размером $M \times N$, так чтобы числа от 1 до $M \times N$ шли по возрастанию «ходом быка»: начало в левом верхнем углу, далее «бык» идет направо, потом делает шаг вниз и по второй строке идет справа налево, в третьей строке опять слева направо. Распечатать получившийся массив в виде таблицы.

12 Дан массив (файл, строка). Найти наибольшее количество одинаковых чисел, идущих в нем подряд.

13 С клавиатуры вводятся длины четырех сторон выпуклого четырехугольника и длина одной из диагоналей. Найти его площадь.

14 С клавиатуры вводятся четыре целых числа. Найти их наибольший общий делитель.

15 Для каждого из чисел из диапазона [2,*N*] напечатать, является ли оно простым.

16 Написать программу проверки существования «близнецов» (так называются простые числа, разность между которыми равна двум) среди чисел меньших заданного *K*.

Примечание. Кстати, до сих пор неизвестно, бесконечно ли множество пар близнецов.

---

17 Написать программу для проверки гипотезы Гольдбаха для заданного числа *K*.

Примечание. Эта гипотеза (на сегодня не опровергнутая и полностью не доказанная) заключается в том, что каждое четное число, большее двух, представляется в виде суммы двух простых чисел.

---

18 В текстовом файле находятся слова (каждое – на отдельной строке). Выписать те из них, в которых содержатся удвоенные буквы.

19 Нарисовать «пирамидку» из прямоугольников (нижние больше верхних). Каждый прямоугольник получается при печати *M* строк любых символов по *N* символов в строке.

20 С клавиатуры вводится последовательность вещественных чисел, заканчивающаяся нулем (нуль не является членом последовательности). Найти в ней максимум и подсчитать, сколько раз он встречается в этой последовательности.

21 В массиве найти минимум и «предминимум» (число, которое стояло бы вторым, сразу после минимального, если бы массив был упорядочен).

Сложные задачи

1 Подсчитать сумму всех положительных элементов массива из 10 целочисленных элементов и количество всех отрицательных его элементов.

2 Проверить упорядоченность введенного пользователем массива.

Примечание. Массив считается упорядоченным, если выполняется одно из условий: 1) каждый последующий его элемент меньше либо равен предыдущему (по убыванию); 2) если каждый последующий элемент больше, либо равен предыдущему (по возрастанию).

---

3 Пользователь вводит два заведомо упорядоченных по возрастанию массива. Сформировать третий упорядоченный по возрастанию массив, состоящий в точности из элементов массивов, введенных пользователем.

4 Отсортировать по возрастанию массив, введенный пользователем. Сортировку осуществлять методом поиска наименьшего элемента.

Примечание. *Метод сортировки поиском наименьшего элемента* является самым долгим (при большом количестве элементов массива), но самым простым в реализации. Суть его состоит в следующем. Пусть в массиве n элементов. За первый проход по массиву программа ищет наименьший элемент и меняет его местами с последним, n-м элементом.

За i-й проход по массиву программа ищет наименьший элемент из оставшихся n-i+1 (кроме самых последних, уже заведомо наименьших) и меняет местами с последним из оставшихся, т. е. с n-i+1 элементом массива.

После n-1 таких проходов по массиву он окажется отсортированным по возрастанию.

---

5 Найти номер первого символа во введенной строке символов, начиная с которого указанная подстрока (последовательность символов) входит в эту строку, либо вывести сообщение о том, что подстрока не содержится во введенной строке. Не использовать в решении соответствующую функцию языка программирования (в Паскале – **Pos**).

Указание. Рекомендуется использовать строковый тип данных и учитывать возможность посимвольного обращения к строкам в Паскале.

---

6 Методом половинного деления найти корень монотонной функции на указанном пользователем отрезке с заданной в программе точностью. Для проверки использовать функцию $F(x)=\mathrm{Cos}(x/2)$ на отрезке [1; 2].

Примечания

1. Метод нахождения корня половинным делением состоит в следующем. Пусть на отрезке [*a, b*] задана непрерывная монотонная функция $F(x)$ и известно, что на данном отрезке она имеет корень ($F(a)*F(b)<0$).

Тогда мы можем поделить отрезок [*a, b*] пополам и подсчитать значение функции в середине отрезка $(a+b)/2$. Если функция имеет разные знаки на левом конце отрезка и в его середине, то мы можем утверждать, что корень функции находится где-то между *a* и $(a+b)/2$. В противном случае корень расположен в правой половине отрезка, между $(a+b)/2$ и *b*. Таким образом, за один шаг мы вдвое уменьшили ширину отрезка, на котором находится искомый корень функции.

При каждом последующем шаге отрезок делится пополам и выбирается та из его половин, на концах которой функция имеет разные знаки, до тех пор, пока длина отрезка не станет меньше необходимой точности.

2. Саму функцию с клавиатуры вводить не нужно (это, вообще говоря, намного более сложная задача, чем поиск корня на отрезке).

Указание. Наиболее удобно при решении задачи использовать пользовательские функции языка программирования (*function*). Однако если вы с ними не знакомы, то можно обойтись и без функций, а формулу, задающую функцию аналитически, каждый раз указывать в тексте программы. Мы рекомендуем вам ознакомиться с решением, использующим пользовательские функции, приведенным в конце книги.

---

7 Используя рекурсию, методом половинного деления найти корень монотонной функции на указанном пользователем отрезке, с заданной в программе точностью. Для проверки использовать функцию $F(x)=\mathrm{Cos}(x/2)$ на отрезке [1; 2].

Примечание. Описание метода половинного деления см. в условии задачи 6.

---

8 Найти наименьший делитель целого числа.

9 Разложить целое число на множители.

Указание. Мы предлагаем вам оформить решение предыдущей задачи в виде процедуры или функции и использовать ее при написании данной задачи. Тогда текст программы станет более понятным как для вас, так и для человека, проверяющего вашу работу.

---

10 Пользователь вводит коэффициенты многочленов $A(x)$ и $B(x)$. Вывести на экран многочлен $C(x)=A(x)*B(x)$.

Указание. В данной задаче удобнее хранить коэффициенты всех трех многочленов в одномерных массивах целых (вещественных) чисел.

---

11 Отсортировать массив по возрастанию методом «пузырька».

Примечание. Суть этого метода состоит в том, что по очереди просматриваются пары соседних элементов, и если элементы в паре стоят не в порядке возрастания, то они меняются местами. В результате одного такого просмотра с перестановками максимальный элемент оказывается на своем окончательном месте – в конце. Далее нужно повторить указанные действия для всех элементов, кроме последнего. Таким образом, в результате каждого прохода уменьшается длина сортируемой последовательности, один из элементов занимает свое «законное» место, а остальные тоже переставляются. Если же в процессе очередного просмотра последовательности не было обнаружено ни одной неупорядоченной пары (не сделано ни одной перестановки), то просмотр следует прекратить: сортировка закончена.

---

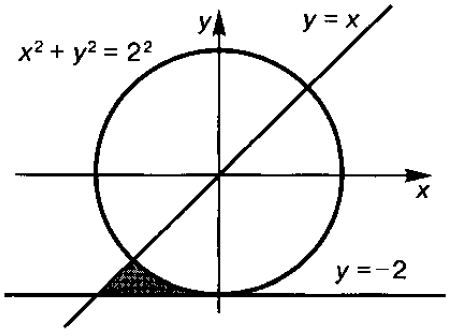
12 Отсортировать массив по возрастанию методом простых вставок.

Примечание. Описание алгоритма сортировки массива методом простых вставок см. в решении задачи в конце книги.

Типовые задачи по программированию части «С» ЕГЭ

1 (*Демо-вариант 2011 г.*)

Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (x, y – действительные числа) и определяет принадлежность точки заштрихованной области. Программист поторопился и написал программу неправильно.



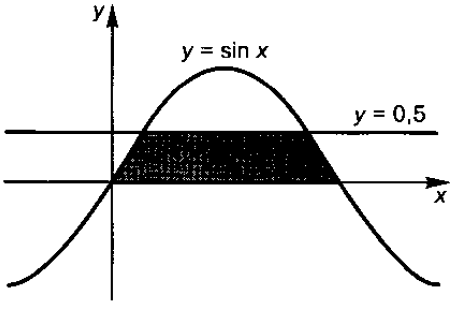

```
var x, y:real;
begin
     readln (x, y);
     if x*x + y*y >= 4 then
      if y >= -2 then
            if y <= x then
            write ('Принадлежит')
     else
      write ('Не принадлежит')
end.
```

Последовательно выполните следующее.

1. Приведите пример таких чисел x и y, 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.)

2 (*Демо-вариант 2011 г.*)

Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (x и y – действительные числа) и определяет принадлежность точки заштрихованной области. Программист торопился и написал программу неправильно.



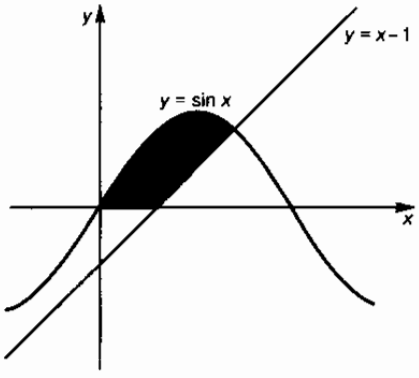

```
var x, y: real;
begin
     readln (x,y);
     if y<=sin(x) then
      if y<=0.5 then
            if y>=0 then
            write ('принадлежит')
     else
      write ('не принадлежит')
end.
```

Последовательно выполните следующее.

1. Приведите пример таких чисел x и y, 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 (*Демо-вариант 2011 г.*)

Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (x и y – действительные числа) и определяет принадлежность точки заштрихованной области. Программист торопился и написал программу неправильно.



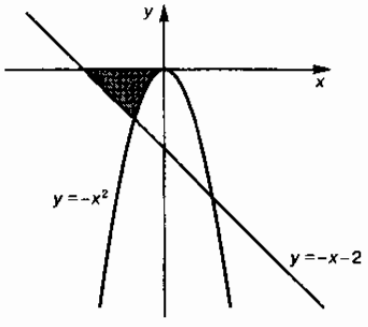

```
var x, y: real;
begin
   readln (x, y);
   if y<=sin(x) then
       if y>=x-1 then
           if y>=0 then
            write ('принадлежит')
   else write ('не принадлежит')
end.
```

Последовательно выполните следующее.

1. Приведите пример таких чисел x и y, 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.)

4 (*Демо-вариант 2011 г.*)

Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (x и y – действительные числа) и определяет принадлежность точки заштрихованной области. Программист торопился и написал программу неправильно.



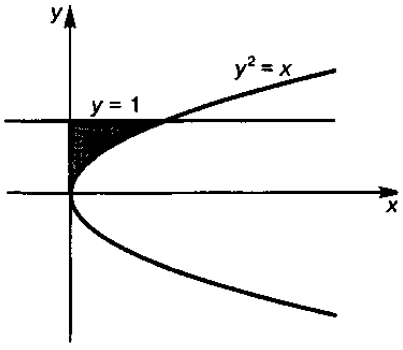

```
var x,y: real;
begin
    readln (x, y);
    if y>=-x*x then
        if y>=-x-2 then
            if y<=0 then
             write ('принадлежит')
    else write ('не принадлежит')
end.
```

Последовательно выполните следующее:

1. Приведите пример таких чисел x и y, 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.)

5 (*Демо-вариант 2011 г.*)

Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (x и y – действительные числа) и определяет принадлежность точки заштрихованной области. Программист торопился и написал программу неправильно.



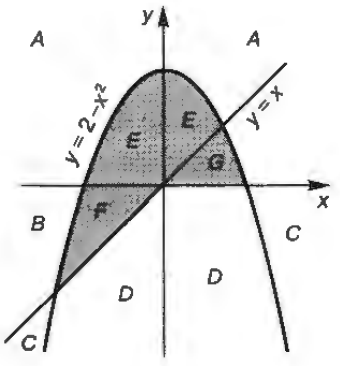

```
var x, y:real;
begin
    readln (x,y);
    if x<=y*y then
     if x>=0 then
          if y<=1 then
          writeln ('принадлежит')
    else
     writeln ('не принадлежит')
end.
```

Последовательно выполните следующее:

1. Приведите пример таких чисел x и y, 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.)

6 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.




```
var x,y: real;
begin
readln(x,y);
if y>=x then
if y>=0 then
if y<=2-x*x 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 и G).

Точки, лежащие на границах областей, отдельно не рассматривать.

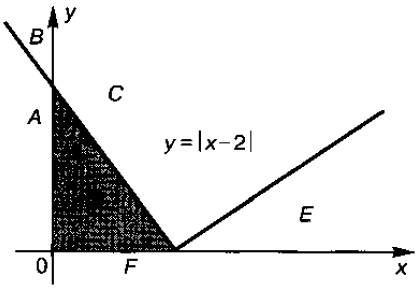
Область	$y>=x$	$y>=0$	$y<=2-x*x$	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					
G					

В столбцах условий укажите «да», если условие выполнится, «нет», если условие не выполнится, «–» (прочерк), если условие не будет проверяться, или «не изв.», если программа ведет себя по-разному для разных значений, принадлежащих данной области. В столбце «Программа выведет» укажите, что программа выведет на экран. Если программа ничего не выводит, напишите «–» (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тесты, напишите «не изв.». В последнем столбце укажите «да» или «нет».

2. Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы.

7 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.




```
var x, y:real;
begin
readln (x,y);
if y>=0 then
if x>=0 then
if y<=abs (x-2) then
writeln ('принадлежит')
else
writeln ('не принадлежит')
end.
```

В области A, B, C, E, F не включаются границы закрашенной области D. Последовательно выполните следующее.

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F).

Точки, лежащие на границах областей, отдельно не рассматривать.

Область	y>=0	x>=0	y<= abs(x-2)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					

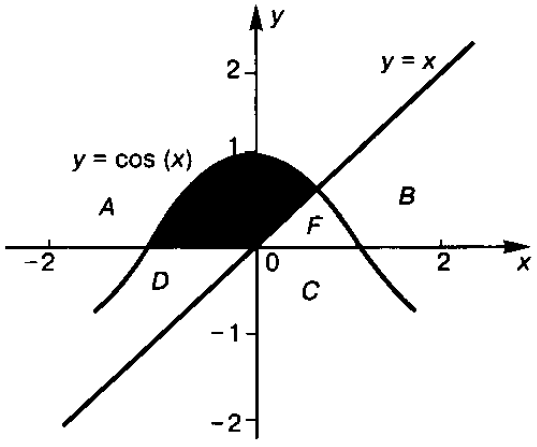
В столбцах условий укажите «да», если условие выполнится, «нет» если условие не выполнится, «–» (прочерк), если условие не будет проверяться, или «не изв.», если программа ведет себя по-разному для разных значений, принадлежащих данной области.

В столбце «Программа выведет» укажите, что программа выведет на экран. Если программа ничего не выводит, напишите «–» (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «не изв». В последнем столбце укажите «да» или «нет».

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

8 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.




```
var x, y: real;
begin
readln(x, y);
if y >= x then
if y>= 0 then
if y <= cos(x)  then
write('принадлежит')
else
write('не принадлежит');
end.
```

В области A, B, C, E, F не включаются границы закрашенной области D. Последовательно выполните следующее.

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F).

Точки, лежащие на границах областей, отдельно не рассматривать.

Область	$y>=x$	$y>=0$	$y <= \cos(x)$	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					
F					

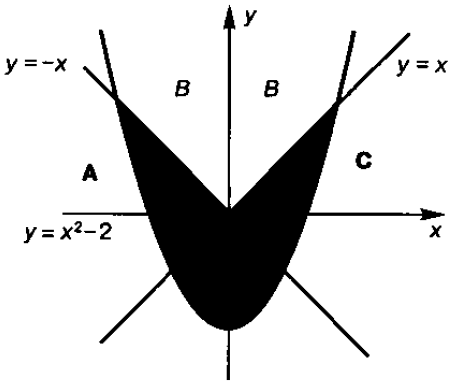
В столбцах условий укажите «да», если условие выполнится, «нет», если условие не выполнится, «–» (прочерк), если условие не будет проверяться, или «не изв.», если программа ведет себя по-разному для разных значений, принадлежащих данной области.

В столбце «Программа выведет» укажите, что программа выведет на экран. Если программа ничего не выводит, напишите «–» (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «не изв». В последнем столбце укажите «да» или «нет».

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

9 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.




```
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.
```

В области A, B, C, E, F не включаются границы закрашенной области D.

Последовательно выполните следующее.

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F).

Точки, лежащие на границах областей, отдельно не рассматривать.

Область	$y<=x$	$y<=-x$	$y >= x*x-2$	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E1					
E2					
F					

В столбцах условий укажите «да», если условие выполнится, «нет», если условие не выполнится, «–» (прочерк), если условие не будет проверяться, или «не изв.», если программа ведет себя по-разному для разных значений, принадлежащих данной области.

В столбце «Программа выведет» укажите, что программа выведет на экран. Если программа ничего не выводит, напишите «–» (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «не изв». В последнем столбце укажите «да» или «нет».

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

10 (*Демо-вариант 2011 г.*)
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 четное значение и делятся на три. Гарантируется, что в исходном массиве есть хотя бы один элемент, значение которого четно и не кратно трем.
Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но использовать все описанные переменные не обязательно.

```
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.
```

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).

11 (*Демо-вариант 2011 г.*)
Дан целочисленный массив из 30 элементов. Опишите на русском языке или на одном из языков программирования алгоритм подсчета суммы всех неотрицательных элементов данного целочисленного массива. Если отрицательных элементов нет вообще, сообщите об этом.

```
const N=30;
var
a: array [1..N] of integer;
i, m, s: integer;
begin
    for i:=1 to N do
```

```
    readln (a[i]);
...
end.
```

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 переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

12 (*Демо-вариант 2011 г.*)

Дан целочисленный массив из 28 элементов. Элементы массива могут принимать значения от 0 до 100 – процент выполнения учащимися домашних заданий по информатике. Для получения положительной оценки за год требовалось набрать не менее 40 баллов. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит минимальный балл среди учащихся, получивших за год положительную оценку. Гарантируется, что в классе хотя бы один учащийся получил за год положительную оценку.

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

```
const N=28;
var
   a: array [1..N] of integer;
   i, j, min: integer;
```

13 (*Демо-вариант 2011 г.*)

Дан вещественный массив из 50 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит наименьший номер отрицательного элемента массива или сообщение, что такого элемента нет.

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

```
const
    N=50;
var
    a: array [1..N] of real;
    i, j: integer;
begin
    for i:=1 to N do
        readln (a[i]);
        …
end.
```

В качестве ответа вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете также записать решение 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.

14 (*Демо-вариант 2011 г.*)

Дан вещественный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит минимальный положительный элемент массива или сообщение, что такого элемента нет.

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

```
const
  N=40;
var
  a: array [1..N] of real;
  i, j: integer;
  min: real;
begin
  for i:=1 to N do
    readln (a[i]);
  …
end.
```

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.

15 (*Демо-вариант 2011 г.*)

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать произвольные значения. С клавиатуры вводится целое число *X*. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит наименьший номер элемента массива, равного *X*, или сообщение, что такого элемента нет.

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

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

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.

16 (*Демо-вариант 2011 г.*)

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит значение второго максимума (элемента, который в отсортированном по невозрастанию массиве стоял бы вторым).

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

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, k, max, max2: integer;
begin
    for i:=1 to N do readln (a[i]);
    …
end.
```

В качестве ответа вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете также записать решение 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.

17 (*Демо-вариант 2011 г.*)

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит номер третьего положительного элемента массива (если из массива вычеркнуть все неположительные элементы, этот элемент стоял бы в получившемся массиве на третьем месте). Если в массиве меньше, чем три положительных элемента, вывести об этом сообщение.

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

```
const
    N=40;
```

```
var
    a: array [1..N] of integer;
    i, j, k: integer;
begin
    for i:=1 to N do
        readln (a[i]);
    …
end.
```

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.

18 (*Демо-вариант 2011 г.*)

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит сумму элементов наибольшей возрастающей последовательности подряд идущих элементов массива.

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

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, l, lmax, s, smax: integer;
begin
    for i:= 1 to N do readln (a[i]);
    …
end.
```

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 пере-

менным, используемым в алгоритме, записанном на естественном языке, с учетом синтаксиса и особенностей используемого вами языка программирования.

19 (*Демо-вариант 2011 г.*)

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит номер элемента массива, наименее отличающегося от среднего арифметического всех его элементов.

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

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, k: integer;
    min, s: real;
begin
    for i:=1 to N do readln (a[i]);
    …
end.
```

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.

20 (*Демо-вариант 2011 г.*)

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит номера двух элементов массива, сумма которых минимальна.

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

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, j, min, min2, s: integer;
begin
for i:=1 to N do readln (a[i]);
    …
end.
```

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.

21 (*Демо-вариант 2011 г.*)

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит номера двух элементов массива, наименее отличающихся друг от друга.

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

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, j, min, min2, s: integer;
begin
    for i:=1 to N do readln (a[i]);
    …
end.
```

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.

22 (*Демо-вариант 2011 г.*)

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько набрал баллов. Эта информация в том же виде была разослана в школы.

Завуч школы №50 решила наградить двух учащихся, которые лучше всех в школе сдали информатику. Программа должна вывести на экран фамилии и имена этих учеников.

Если наибольший балл набрало больше двух человек – вывести количество таких учеников.

Если наибольший балл набрал один человек, а следующий балл набрало несколько человек, – нужно вывести только фамилию и имя лучшего.

Напишите эффективную, в том числе и по используемой памяти, программу, которая должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников школы №50.

На вход программы сначала подается число учеников, сдавших экзамен. 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, причем ровно один между каждой парой (то есть всего по три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
Круглов Василий
Тарасова Дарья
```

Другой вариант выходных данных:

```
7
```

Третий вариант выходных данных:

```
Гусаров Илья
```

23 (*Демо-вариант 2012 г.*)

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 Интернета.

Перед текстом программы кратко опишите используемый вами алгоритм решения задачи.

На вход программе в первой строке подается 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 трех наиболее популярных задач с указанием количества запросов по ним. Если в запросах упоминаются менее трех задач, то выведите информацию об имеющихся задачах. Если несколько задач имеют ту же частоту встречаемости, что и третья по частоте встречаемости задача, то их тоже нужно вывести.

Пример выходных данных для приведенного выше примера входных данных:

```
A+B 2
Простой делитель 2
Крестики-Нолики 1
Прямоугольник 1
```

24 (*Демо-вариант 2011 г.*)

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько набрал баллов.

Районный методист решила выяснить номер школы, ученики которой набрали наибольший средний балл, с точностью до целых.

Программа должна вывести на экран номер такой школы и ее средний балл.

Если наибольший средний балл набрало больше одной школы – 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 должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников района. Также известно, что в районе школы с некоторыми номерами не существуют.

На вход программе сначала подается число учеников, сдававших экзамен. 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 в диапазоне от 1 до 99, <Количество баллов> – целое число в диапазоне от 1 до 100. Эти данные записаны через пробел, причем ровно один между каждой парой (т.е. всего по три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
50 74
```

Другой вариант выходных данных:

```
7
```

25 (*Демо-вариант 2011 г.*)

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько баллов набрал.

Районный методист решила выяснить номера школ, ученики которых набрали средний балл по школе, больший, чем районный средний балл (все средние баллы вычисляются с точностью до целых).

Программа должна вывести на экран номера таких школ, в любом порядке.

Если такая школа окажется только одна – вывести также средний балл по этой школе, с указанием, что это средний балл.

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников района. Также известно, что в районе школы с некоторыми номерами не существуют.

На вход программе сначала подается число учеников, сдававших экзамен.

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, причем ровно один между каждой парой (т.е. всего по три пробела в каждой строке).
Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
5 50 74 87
```

Другой вариант выходных данных:

```
7
Средний балл = 74
```

26 (*Демо-вариант 2011 г.*)
После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько набрал баллов.
Районный методист решила выяснить фамилии учеников, которые набрали наибольший балл, по каждой школе в отдельности, но только если из школы информатику сдавало не меньше трех человек. Если в школе информатику сдавало меньше трех человек, информацию по этой школе выводить не нужно. Если наибольший балл в какой-то школе набрали несколько человек, нужно вывести на экран их количество.
Программа должна вывести на экран информацию 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 должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников района. Также известно, что в районе школы с некоторыми номерами не существуют.
На вход программе сначала подается число учеников, сдававших экзамен. 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 0 до 100. Эти данные записаны

через пробел, причем ровно один между каждой парой (т.е. всего три пробела в каждой строке).
Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
5 Иванов
50 10
74 Сидоров
```

27 (*Демо-вариант 2011 г.*)
После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько баллов набрал.
В районе считается подозрительной ситуация, когда в школе более двух учащихся набирают одинаковый наибольший балл по школе. Районный методист решила выяснить номера таких школ.
Программа должна вывести номера этих школ в любом порядке.
Если такая школа окажется одна, нужно вывести наибольший балл в этой школе с указанием того, что это наибольший балл.
Если таких школ не окажется, нужно вывести об этом сообщение.
Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая должна вывести на экран требуемую информацию. Известно, что информатику сдавало больше пяти учеников района. Также известно, что в районе школы с некоторыми номерами не существуют.
На вход программе сначала подается число учеников, сдававших экзамен. 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 0 до 100. Эти данные записаны через пробел, причем ровно один между каждой парой (т.е. всего по три пробела в каждой строке).
Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
5 50 74 87
```

Другой вариант выходных данных:

```
7
Наибольший балл = 74
```

Третий вариант выходных данных:

```
Нет таких школ
```

29 (*Демо-вариант 2011 г.*)

При программировании школьной тестирующей системы по английскому языку выяснилось, что файлы с вопросами к тестам легкодоступны, и каждый может перед тестом открыть их и заранее узнать вопросы. Было решено закодировать файлы. Для этого придумали следующий алгоритм. Каждая строка файла кодируется отдельно.

В каждой строке ищутся отдельные слова, и все символы слова сдвигаются по алфавиту циклически вправо на длину слова.

Словом считается любая последовательность подряд идущих символов латинского алфавита, строчных и прописных.

Циклический сдвиг символа по алфавиту вправо на X – замена символа на символ, стоящий в алфавите на X позиций дальше. Если при этом происходит выход за пределы алфавита, счет начинается с начала алфавита.

Пример циклического сдвига символов на 3 позиции: буква «E» превращается в букву «H», буква «t» – в букву «w», буква «Y» – в букву «B».

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая должна закодировать строку по указанному алгоритму.

На вход программе подается строка, состоящая из не более чем 250 символов латинского алфавита, пробелов, знаков препинания, разного рода скобок, кавычек и других символов. Строка заканчивается символом «#». Других символов «#» в строке нет.

Программа должна вывести закодированную по указанному алгоритму строку.

Пример входных данных:

```
Day, mice. "Year" - a mistake#
```

Пример выходных данных:

```
Gdb, qmgi. "Giev" - b tpzahrl#
```

29 (*Демо-вариант 2011 г.*)

После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько баллов набрал. По положению об экзамене каждый район сам определяет, за какой балл нужно поставить какую оценку.

Районный методист решила, что оценку «отлично» должны получить 20% участников (целое число, с отбрасыванием дробной части). Для этого она должна определить, какой балл должен набрать ученик, чтобы получить «отлично».

Если невозможно определить такой балл, чтобы «отлично» получили ровно 20% участников, «отлично» должно получить меньше участников, чем 20%.
Если таких участников не окажется (наибольший балл набрали больше 20% участников) – эти и только эти ученики должны получить «отлично».
Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая должна вывести на экран наименьший балл, который набрали ученики, получившие «отлично». Известно, что информатику сдавало больше пяти учеников. Также известно, что есть такое количество баллов, которое не получил ни один участник.
На вход программе сначала подается число учеников, сдававших экзамен. 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, причем ровно один между каждой парой (т.е. всего три пробела в каждой строке).
Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
78
```

30 (*Демо-вариант 2011 г.*)
После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько баллов набрал. По положению об экзамене оценку «2» (неудовлетворительно) получают ученики, набравшие меньше 40 баллов. Оценку «3» (удовлетворительно) получают 30% учеников среди оставшихся, за исключением тех, кто набрал больше 60 баллов.
Если количество «троечников» оказывается больше 30%, то следует выбрать меньшую границу для оценки «4» (но только если при этом «3» получит хоть кто-нибудь).
Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая должна вывести на экран наибольший балл, который набрали ученики, получившие «удовлетворительно», и количество таких учеников. Известно, что информатику сдавало больше 50 учеников. Также известно, что есть такое количество баллов, которое не получил ни один участник.

На вход программе сначала подается число учеников, сдавших экзамен. 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, причем ровно один между каждой парой (т.е. всего три пробела в каждой строке).

Пример входной строки:

```
Иванов Иван 50 87
```

Пример выходных данных:

```
45 703
```

31 (*Демо-вариант 2011 г.*)

На вход программе подается последовательность символов, заканчивающаяся символом #. Другие символы # во входной последовательности отсутствуют.

Программа должна вывести на экран латинскую букву, встречающуюся во входной последовательности наибольшее количество раз, и число этих раз (во второй строке).

Если таких букв во входной последовательности окажется несколько, программа должна вывести на экран все их через пробел в алфавитном порядке.

Строчные и прописные буквы не различаются.

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая должна решать поставленную задачу.

Пример входных данных:

```
Day, mice. "Year" - a mistake#
```

Пример выходных данных:

```
A
4
```

Другой вариант:

Пример входных данных:

```
ABCD ABCE ABCF#
```

Пример выходных данных:

```
A B C
3
```

Ответы

Раздел «Системы счисления»

1. 6. **2.** 7. **3.** 4. **4.** 11. **5.** 9. **6.** 5, 10, 25, 50. **7.** 3, 4, 6, 8, 12, 24. **8.** 2, 4, 8, 16, 32. **9.** 6, 9, 12, 18, 36. **10.** 8, 16, 32. **11.** 5, 13, 21, 29. **12.** 3, 7, 11, 15. **13.** 4, 12, 20, 28. **14.** 2, 6, 10, 14. **15.** 6, 14, 22, 30. **16.** 3. **17.** 4. **18.** 5. **19.** 2. **20.** 5. **21.** D2. **22.** 207. **23.** 2310. **24.** C9. **25.** E4. **26.** 1101,1. **27.** 10101,011. **28.** 101001,01. **29.** 100110,11. **30.** 11011,101. **31.** 640. **32.** 776. **33.** 353. **34.** 515. **35.** 367. **36.** 20B. **37.** 2A8. **38.** 1EF. **39.** 27D. **40.** 37C. **41.** 30. **42.** 101000. **43.** 312. **44.** 146. **45.** 8D. **46.** 8. **47.** 10010. **48.** C2. **49.** 232. **50.** 215.

Раздел «Информация и ее кодирование»

1. 512 бит. **2.** 448 бит. **3.** 256 бит. **4.** 592 бита. **5.** 488 бит. **6.** 1. **7.** 2. **8.** 16. **9.** 8. **10.** 2. **11.** 32. **12.** 1024. **13.** 4. **14.** 8192. **15.** 16. **16.** 2 в 26-й степени. **17.** 2 в 27-й степени. **18.** 2 в 25-й степени. **19.** 2 в 32-й степени. **20.** 2 в 36-й степени. **21.** 4. **22.** 8. **23.** 2. **24.** 16. **25.** 64. **26.** 12 Кбайт. **27.** 16 Кбайт. **28.** 900 Кбайт. **29.** 90 Кбайт. **30.** 64 Кбайт. **31.** 34. **32.** 29. **33.** 115. **34.** 14. **35.** 525. **36.** 18,31 Мбайт. **37.** 6,4 Мбайт. **38.** 10,98 Мбайт. **39.** 1,3 Мбайт. **40.** 3,2 Мбайт. **41.** 656. **42.** 267. **43.** 220. **44.** 328. **45.** 768. **46.** 134. **47.** 169. **48.** 180. **49.** 110. **50.** 525. **51.** 5. **52.** 4. **53.** 6. **54.** 8. **55.** 6. **56.** 729. **57.** 243. **58.** 81. **59.** 128. **60.** 1024. **61.** 7. **62.** 6. **63.** 8. **64.** 5. **65.** 7. **66.** 32. **67.** 64. **68.** 128. **69.** 64. **70.** 81. **71.** 4 байта. **72.** 63 бита. **73.** 32 байта. **74.** 25 байт. **75.** 90 байт. **76.** 8. **77.** 16. **78.** 4. **79.** 32. **80.** 4. **81.** 144. **82.** 99. **83.** 81. **84.** 135. **85.** 153. **86.** 98 байт. **87.** 108 байт. **88.** 25 байт. **89.** 147 байт. **90.** 72 байта. **91.** 36 байт. **92.** 135 бит. **93.** 27 байт. **94.** 45 байт. **95.** 54 байта. **96.** 10. **97.** 9. **98.** 7. **99.** 8. **100.** 6.

Раздел «Алгебра логики»

1. 4. **2.** 1. **3.** 2. **4.** 3. **5.** 3. **6.** 1. **7.** 1.

8. 1. **Рекомендации.** Поставьте к задаче вопрос с отрицанием: при каких значениях X данное высказывание будет ложным? Решите эту задачу, а потом преобразуйте ответ. **9.** 2. **10.** 4. **11.** 2. **12.** 4. **13.** 2. **14.** 2. **15.** 3. **16.** 5. **17.** 5. **18.** 9. **19.** 7. **20.** 9. **21.** 7. **22.** $(A \wedge \neg B) \vee (\neg A \wedge B)$. **23.** $\neg A \wedge \neg B$. **24.** B. **25.** 0. **26.** $(A \wedge B) \vee (\neg A \wedge \neg B)$. **27.** 0. **28.** $A \wedge B$. **29.** $\neg A \wedge \neg B$. **30.** $(A \wedge \neg B) \vee (\neg A \wedge B)$. **31.** 0. **32.** $(\neg A \wedge \neg B) \vee \vee (A \wedge B)$. **33.** $\neg A$. **34.** A. **35.** $\neg A \wedge B$. **36.** 0. **37.** $B \vee A \wedge \neg B$. **38.** $\neg B$. **39.** $\neg A \vee \neg B \vee \vee C$. **40.** $A \vee B$. **41.** 0. **42.** $(\neg A \wedge \neg B \wedge \neg C) \vee (A \wedge B \wedge \neg C)$. **43.** 0. **44.** $(A \wedge \neg B) \vee \vee (\neg B \wedge \neg C)$. **45.** $(\neg A \wedge \neg B) \vee (A \wedge \neg C) \vee (B \wedge \neg C)$. **46.** $(\neg A \wedge \neg B \wedge C) \vee (A \wedge \wedge \neg B \wedge \neg C)$. **47.** $Y \wedge Z$. **48.** Тождественно истинная формула. **49.** 4. **50.** 1. **51.** 2. **52.** 1. **53.** 3. **54.** 3. **55.** 2. **56.** 4. **57.** $(A \wedge C) \vee (\neg B \wedge C)$.

58. $(\neg A \wedge \neg B) \vee (\neg A \wedge \neg C) \vee (\neg B \wedge \neg C)$. **59.** $(\neg A \wedge \neg C) \vee (A \wedge C)$. **60.** $(A \wedge B) \vee (\neg A \wedge \neg B)$. **61.** $(A \wedge \neg C) \vee (\neg A \wedge C)$. **62.** $(\neg A \wedge \neg B) \vee (\neg A \wedge C) \vee \vee (\neg B \wedge C)$. **63.** $(A \wedge \neg B) \vee (\neg B \wedge C) \vee (\neg A \wedge B \wedge \neg C)$.

64.

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

65.

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

66.

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

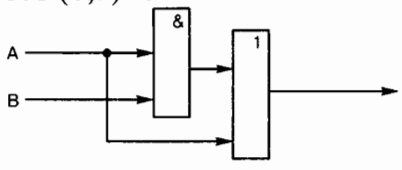
67. 3. **68.** 4. **69.** 3. **70.** 2. **71.** 1. **72.** 2. **73.** 1. **74.** 1. **75.** 1111. **76.** 1000. **77.** 0111. **78.** 1100. **79.** 010. **80.** 1010. **81.** 10. **82.** 30. **83.** 30. **84.** 8. **85.** 6. **86.** 2. **87.** 14. **88.** 6. **89.** 2. **90.** 3. **91.** 14. **92.** 12. **93.** 10. **94.** 4. **95.** 3. **96.** 8. **97.** 8. **98.** 24.

Раздел «Логические задачи»

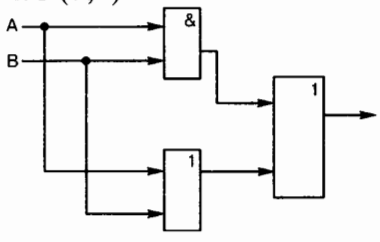
1. КМС. **2.** 1423. **3.** 1342. **4.** ВСРМН. **5.** АКВТМСНР. **6.** Кувшин изготовлен во Владимире в XV веке. **7.** СВП. **8.** О,С. **9.** ВРА. **10.** ЕЛП. **11.**О. **12.** М. **13.**Л. **14.** АМВЕ. **15.** АГЕВ. **16.** ГАЕВ. **17.** РШГБФ. **18.** АРГФН. **19.**СЛИ. **20.** ИИО. **21.** Н. **22.** РКУГИ. **23.** 2а1а2б3а4а.

Раздел «Логические схемы»

3. $F(0,1)=0$



4. $F(1,0)=0$



11. $\neg(A \wedge B \vee \neg C)$. **12.** $(A \wedge B \vee \neg A) \vee \neg(A \wedge B)$. **13.** 2.
14. На месте *X* стоит элемент *2-И*, на месте *Y* стоит *2-ИЛИ*.

Раздел «Исполнители алгоритмов»

2. 2212.

Решение: Поиск решения будем вести от ответа, приближаясь к исходному числу. Ближайшее число, из которого можно извлечь квадратный корень, будет 9. Из 9 вычитанием единицы можно получить 8, т.е. послед-

няя команда – 2. 9 можно получить, возведя 3 в квадрат. Таким образом, последние 2 команды – 12. А 3 получаем из исходного числа 5, дважды вычитая единицу, т.е. первые две команды в программе – 22. Тогда программа будет выглядеть так: 2212. Программа содержит 4 команды, значит условие задачи выполнено.

3. 21221. **4.** 12121. **5.** 22211.

7. **Решение:** Обозначим $R(n)$ – количество программ, которые преобразуют число 1 в число n. Обе команды исполнителя увеличивают исходное число, поэтому общее число команд в программе не может превосходить 16. $R(1) = 1$. Для каждого следующего числа рассмотрим, сколькими способами это число может быть получено из предыдущих чисел за одну команду исполнителя. Если число нечетное, то последней командой может быть только «Прибавь 1». Следовательно, количество программ для этого числа равно количеству программ для предыдущего числа:

$R(i) = R(i\text{-}1)$, если i нечетное.

Если число четное, то вариантов последней команды два: «Прибавь 1» или «Умножь на 2». Следовательно, $R(i) = R(i\text{-}1) + R(i/2)$, если i четное.

Для числа 16 количество программ можно подсчитать, составив таблицу:

Число	1	2	3	4	5	6
Количество команд исполнителя	1	1+1=2	2	2+2=4	4	2+4=6
Число	7	8	9	10	11	12
Количество команд исполнителя	6	4+6=10	10	4+10=14	14	6+14=20
Число	13	14	15	16		
Количество команд исполнителя	20	6+20=26	26	10+26=36		

9. 45.

Решение: Черепашке требуется прочертить 9 линий. Следовательно, за 8 поворотов направо Черепашка должна совершить полный оборот вокруг своей оси. Тогда $n = 360 / 8 = 40$.

10. Правильный шестиугольник.

11. Правильный пятиугольник, слева.

Решение: Черепашка нарисует пять линий и при этом повернется на 360 градусов вокруг своей оси и окажется в той же точке, что и в начале. В результате на экране появится правильный пятиугольник. Поскольку Чере-

пашка поворачивается против часовой стрелки, фигура будет нарисована слева от Черепашки.

12. Звезда, состоящая из 20 лучей.

Решение: За один шаг цикла Черепашка рисует линию, возвращается назад и поворачивается на 18 градусов. Таких линий, исходящих из одной точки, Черепашка нарисует 22. Но за 20 шагов Черепашка полностью обернется вокруг своей оси (360 / 18 = 20), и последние 2 линии она прочертит по уже нарисованным линиям. Следовательно, на экране появится звезда с 20 лучами.

14. 313. **15.** 1, A8. **16.** 4. **17.** 3, A2, A7, C6. **18.** 3.

20. Вперед 1.

Решение: Если всего команд 52, то команд "Назад 2" было 31, а "Вперед 3" всего 21. Кузнечик прыгнул вперед на 21 × 3 = 63 шага, а назад на 31 × 2 = 62 шага. Тем самым он оказался на 1 шаг впереди от первоначальной точки. Последовательность команд в алгоритме в данном случае не имеет значения.

22. 3.

Решение: Для первой бусины возможны 3 варианта, так как бусин, помеченных гласной буквой, три.

Для второй бусины тоже возможны 3 варианта, так как бусин, помеченных согласной буквой, три.

Для третьей бусины возможны только 2 варианта: из трех бусин мы можем выбрать только две, которых нет на втором месте.

Аналогично для четвертой бусины – 2 варианта.

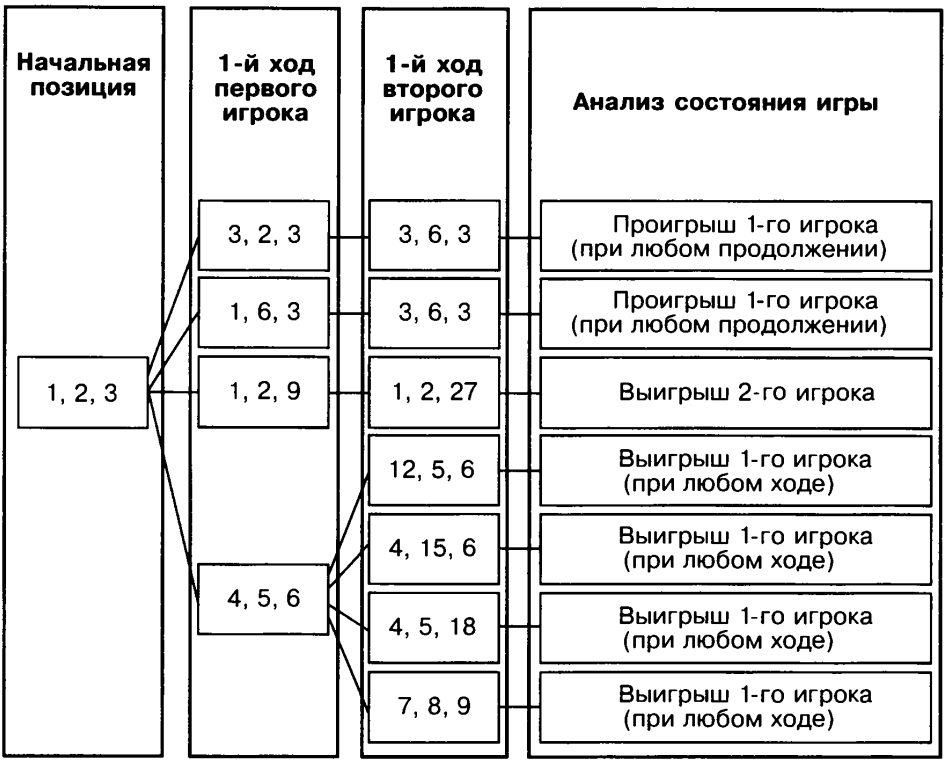
Теперь, если мы перемножим все варианты, то получим количество вариантов цепочек, построенных по правилу, изложенному в условии задачи: 3 × 3 × 2 × 2 = 36.

В результате получаем, что существует 36 вариантов цепочек, построенных по этому правилу.

23. 2. **24.** 1. **25.** 1. **26.** 3. **27.** 3. **28.** 4. **29.** 2. **31.** AACBAABA. **33.** 42. **34.** 340.

37. При правильной игре выигрывает первый игрок на втором или третьем ходу, если сделает первый ход 6, 2, а в случае ответного хода второго игрока 6, 4 или 6, 5 сделает второй ход 6, 7 и 6, 8 соответственно.

Решение: Для доказательства рассмотрим неполное дерево игры.



Вершина дерева игры – начальное состояние игры; на уровне 1 дерева показаны все 4 возможных состояния игры после 1-го хода 1-го игрока. Рассмотрим только ход первого игрока 3, 2 → 6, 2, поскольку нас интересует безошибочная игра, приводящая к победе игрока, сделавшего этот ход. Данный ход первого игрока приводит к победе, поэтому остальные ходы рассматривать не имеет смысла (хотя можно отметить, что любой ход первого игрока может привести к его победе при правильной игре). Два ответных хода второго игрока 12, 2 и 9, 2 приведут к победе первого игрока уже на втором ходе. Другие два хода 6, 4 и 6, 5 приведут к победе на 3-м ходе первого игрока, если он на втором ходе сделает правильный ход: 6, 4 → 6, 7 или 6, 4 → 6, 8 или 6, 5 → 6, 8. В этом случае любой ответный ход второго игрока не помешает первому игроку выиграть.

38. При правильной игре выигрывает второй игрок на втором ходе, если сделает ход 7,8 в ответ на ход первого игрока 5,6. Все другие варианты хода первого игрока приводят к победе второго игрока уже на первом ходе.

39. При правильной игре выигрывает второй игрок на четвертом ходе (при этом его первый ход должен быть 2,2 → 4,2, или 1,4 → 2,4, или 3,2 → 3,4).

40. В задаче необходимо построить дерево решений: для выигрывающего игрока необходимо указать правильную последовательность ходов, для проигрывающего игрока необходимо рассмотреть ***все*** возможные варианты ответных ходов. Так как заранее неизвестно, какой игрок выигрывает

при правильной игре обоих игроков, то будем рассматривать ***все*** возможные ходы ***обоих*** игроков.

Процесс игры можно описать с помощью таблицы, в которой в первом столбце содержится начальное количество камней в кучках. Каждый четный столбец соответствует возможным ходам первого игрока. Каждый нечетный столбец соответствует ходам второго игрока.

	1-й игрок	2-й игрок	1-й игрок	2-й игрок
5,6	12, 5	24, 5 — проигрыш	выигрыш	
		12, 10	24, 10	выигрыш
			12, 20	выигрыш
			36, 10	выигрыш
			12, 30	выигрыш
		36, 5 — проигрыш	выигрыш	
		12, 15 — проигрыш	выигрыш	
	6, 10	12, 10	24, 10	выигрыш
			12, 10	выигрыш
			36, 10	выигрыш
			12, 30	выигрыш
		6, 20 — проигрыш	выигрыш	
		18, 10 — проигрыш	выигрыш	
		6, 30 — проигрыш	выигрыш	
	18, 5	36, 5 — проигрыш	выигрыш	
		18, 10 — проигрыш	выигрыш	
		54, 5 — выигрыш		
		18, 15 — проигрыш	выигрыш	
	6, 15	12, 15 — проигрыш	выигрыш	
		6, 30 — проигрыш	выигрыш	
		18, 15 — проигрыш	выигрыш	
		6, 45 — выигрыш		

Ответ: Второй игрок выигрывает либо на втором, либо на четвертом ходе, если будет следовать стратегии, выделенной в таблице серым цветом. При всех остальных вариантах второго хода второго игрока у первого игрока есть выигрышные ходы.

Раздел «Представление алгоритмов»

1. 9. **2.** 5. **3.** 40. **4.** 6. **5.** S=2, P=6. **6.** 23. **7.** 3. **8.** 1. **9.** b= –1, d=24. **10.** 29. **11.** 90. **12.** 56. **13.** -19. **14.** P=56, B=13. **15.** 78. **16.** c=31, d=4. **17.** m= –260, d=8, n=31. **18.** a=16, b=6. **19.** 54, 14. **20.** 333, 999. **21.** 3. **22.** 3, 12, 3. **23.** 803, 811. **24.** 9, 23, 5. **25.** 0, 16. **26.** 21. **27.** -6, 10. **28.** 95. **29.** 3. **30.** 10. **31.** 17. **32.** 51. **33.** 55. **34.** 43. **35.** Меняет 3 столбец и диагональ. **36.** -30.

Раздел «Моделирование и компьютерный эксперимент»

1. 4. **2.** 2. **3.** 3. **4.** Иван–Гриша. **5.** Вика. **6.** Света. **7.** 2. **8.** Все равно. **9.** 1. **10.** Через 150 минут. **11.** Грушевое, 12:10. **12.** 6. **13.** 4. **14.** 25 минут. **15.** 2. **16.** Борис ждет 10 минут.

Раздел «Программные средства информационных и коммуникационных технологий»

1. Да, в обоих случаях.

Пояснение: Файлы и папки различаются по их полному имени, которое включает в себя «префикс» – путь от корня к данному файлу, зависящий от места файла. Например, **C:\School\Inform\Tasks.txt** и **C\:Games\ Tasks.txt** – файлы с одинаковыми названиями, но полные имена у них разные: оба находятся на диске **C:**, но первый – в папке **Inform**, которая, в свою очередь, содержится в папке **School**, а второй – в папке **Games**.

2. Да.

Пояснение: При именовании файлов имеет значение не только непосредственное имя (**Tasks**), но и расширение имени файла. В данном случае один файл имеет расширение **txt**, а другой **tif**.

3. txt, doc, rtf

4. а) нет,

б) да.

5. Переместить.

Пояснение: При копировании файлов на новом месте появляются копии файлов, а исходные файлы при этом остаются на старом месте без изменений. На операцию копирования требуется значительное количество времени (особенно при большом объеме информации), так как происходит физическая запись на диск.

При выполнении операции «Перемещение файлов» в пределах одного диска никакого физического перемещения информации не происходит. Непосредственно сами файлы физически остаются там, где они и были, перемещаются лишь названия (изменяются ссылки). В этом случае опе-

рация осуществляется практически мгновенно. А вот при перемещении файлов с одного диска (устройства) на другой происходит физический перенос информации, как и при копировании.

6. Копировать.

Пояснение: Операция «Перемещение файлов» удалит файлы из памяти компьютера Пети.

7. Ctrl. **8.** Shift.

9. *едвед*.doc.

Пояснение: Расширение надо выбрать **doc**, так как редактор MS Word 2003 обычно сохраняет файлы именно в этом формате. В названии файла обязательно будет присутствовать часть слова «едвед». Первую букву в шаблон брать нельзя, так как она может быть как прописная, так и строчная. Последняя буква тоже в шаблон не записывается, потому что в разных падежах слово имеет разные окончания. Так как и перед выбранным сочетанием букв, и после него может находиться некоторое количество других знаков, слева и справа записываются звездочки.

10.

А) **FFTT*. JPG**.

Б) **FFTT03*. JPG**.

В) **FFTT0711000?.JPG**.

Г) **FFTT1*. JPG**.

Д) Здесь выделить нужные файлы с помощью одного шаблона не получится. Более того, так как время в имени файла никак не фигурирует, мы не сможем отличить «дневные» фотографии от «ночных». Можно получить все фотографии за эти 2 дня с помощью двух шаблонов: **FFTT1231*.JPG** и **FFTT0101*.JPG**.

Е) Нет.

11. 1. **12.** 5. **13.** 12. **14. D:\ВЫПУСКНИКИ\диплом.doc**. **15. C:\ОБУЧЕНИЕ\РАБОТЫ\диплом.doc**. **16. C:\Windows\Web\Wallpaper\Ветер.bmp**. **17. A:\B\C2\text.txt**. **18. D:\ОБЩИЕ\УЧЕНИКИ\УСПЕВАЕМОСТЬ**. **19. C:\FF\K**. **20. A:\B2\C1\D1**. **21. D:\Обучение\Студент\Flash**. **22. D:\Проекты\Name.txt**.

Раздел «Технология обработки информации в электронных таблицах MS Excel»

1. 3. **2.** 2. **3.** 4. **4.** 4. **5.** 3. **6.** 2. **7.** 1. **8.** 2. **9.** 1. **10.** 3. **11.** 2. **12.** 3. **13.** 1. **14.** 2. **15.** 2. **16.** 3. **17.** 4. **18.** 1. **19.** 2. **20.** 1. **21.** 4. **22.** 31.01.2008. **23.** 11.05.2008. **24.** 29.04.2008. **25.** 20.04.2008. **26.** 27.02.2008. **27.** 27.08.2008. **28.** 14.10.2008. **29.** 13:00. **30.** 14:15. **31.** 12:35. **32.** 1:20. **33.** 5:10. **34.** 3:30. **35.** 4. **36.** 3. **37.** 2. **38.** 2. **39.** 3. **40.** 3. **41.** 2. **42.** 3. **43.** 4 10 “A”.

Раздел «Технология хранения, поиска и сортировки информации в базах данных»

1. 3. **2.** 4. **3.** 3. **4.** 1. **5.** 4. **6.** 1. **7.** 3. **8.** 2.

Раздел «Телекоммуникационные технологии»

1. 2. **2.** 3. **3.** 2. **4.** 1. **5.** 4. **6.** 1. **7.** 4. **8.** 2. **9.** 1. **10.** 3. **11** 1. **12.** ГАДЕЖБВ (http://narod.ru/ftp.rar). **13.** БГЕВАД (ftp://my_files.docs.ru/raspisaniye.docx). **14.** БЖВГАДЕ (www.midi.ru/catalog/mazurka.mid). **15.** АЖВБЕГД (http://www.microsoft.com/login.php). **16.** ГЖАБЗВДИЕ (http://my_first_site.narod.ru/raznoye/document.txt). **17.** ЕВДЖАГБ (http://www.files.org/games/tetris.zip). **18.** БГДАЕ (ftp://my_ftp.org/downloads/films/kinzaza.avi). **19.** ГЕАЖБВД (www.my_site.narod.ru/news/archive/page.php). **20.** ГБВА. **21.** 215.62.83.19. **22.** 252.183.27.239. **23.** 271. **24.** 282. **25.** 256. **26.** Б. **28.** ГВЖЕДБА. **29.** БГЖДВЕА.
30. ЕЖГБДБВБА.

Решение:

<протокол> <://> <www> <точка > <имя сайта><точка ><домен организации><точка><домен страны>

Адрес сайта: http://www.cmc.msu.ru

31. ВЖБДЕДГДА. **32.** БГЕДЖДВДА. **33.** 213.121.29.96. **34.** 213.121.29.97. **35.** 213.121.29.110. **36.** 213.121.29.111. **37.** 213.121.28.0. **38.** 213.121.28.1. **39.** 213.121.29.254. **40.** 213.121.29.255. **41.** 193.11.200.0. **42.** 193.11.200.1. **43.** 193.11.200.126. **44.** 193.11.200.127. **45.** 144.200.28.0. **46.** 144.200.28.1. **47.** 144.200.31.254. **48.** 144.200.31.255. **49.** 150.255.97.192. **50.** 150.255.97.193. **51.** 150.255.97.254. **52.** 150.255.97.255. **53.** 27. **54.** 5. **55.** 22. **56.** 9. **57.** 20. **58.** 27. **59.** 27. **60.** A. **61.** A. **62.** D. **63.** D. **64.** B. **65.** B. **66.** C. **67.** C.

Раздел «Ввод и вывод числовой информации. Выражения»

1

```
var A: Integer;
begin
  Write('Введите A: ');
  Readln(A);
  Write('A*A = ', A*A);
  Readln
end.
```

2

```
var A, B: Integer;
begin
  Write('Введите A: ');
  Readln(A);
```

```
  Write(' Введите B: ');
  Readln(B);
  Writeln('S=', A*B);
  Readln
end.
```

3

```
var A, B: Integer;
begin
  Write('Введите A: ');
  Readln(A);
  Write('Введите B: ');
  Readln(B);
  Writeln('(A+B)/2 =', (A+B)/2:5:1);
  Readln
end.
```

4

```
var A, B: Integer;
begin
  Write('Введите A: ');
  Readln(A);
  Write('Введите B: ');
  Readln(B);
  Writeln('Среднее геометрическое чисел A и B равно',
          Sqrt(A*B):6:3);
  Readln
end.
```

5

```
var A, B: Integer;
    C, S: Real;
begin
  Write('Введите A: ');
  Readln(A);
  Write('Введите B: ');
  Readln(B);
  C:=Sqrt(A*A+B*B);
  Writeln('Гипотенуза C=', C:5:2);
  S:=A*B/2;
  Writeln('Площадь S=', S:8:2);
  Readln
end.
```

6

```
var Alpha, A: Real;
begin
  Write('Введите угол Alpha: ');
  Readln(Alpha);
  A:=2*Pi*Alpha/360;
  Writeln('Sin (Alpha)=', Sin(A):6:4);
```

```
  Writeln('Cos (Alpha)=', Cos(A):6:4);
  Readln
end.
```

7

```
var N: Integer;
begin
  Write('Введите N: ');
  Readln(N);
  Writeln('Результат: ', (N-1)*(N-1), ', ', N*N,
          ',', (N+1)*(N+1));
  Readln
end.
```

8

```
var A, B, C1, C2: Integer;
begin
  Write('Введите A: ');
  Readln(A);
  Write('Введите B: ');
  Readln(B);
  C1:=(A+B)*(A+B)*(A+B)*(A+B);
  C2:= A*A*A*A + 4*A*A*A*B + 6*A*A*B*B + 4*A*B*B*B
       + B*B*B*B;
  Writeln(C1, ' = ', C2);
  Readln
end.
```

9

```
var Alpha, S2, C2: Real;
begin
  Write('Введите Alpha: ');
  Readln(Alpha);
  S2:=Sqr(Sin(Alpha));
  C2:=Sqr(Cos(Alpha));
  Write(S2:6:4, ' + ', C2:6:4, ' = ', S2+C2:6:4);
  Readln
end.
```

10*

```
var a, b, c, D, x1, x2: Real;
begin
  Writeln('Уравнение ax*x + bx + c = 0');
  Write('a=');
  Readln(a);
  Write('b=');
  Readln(b);
  Write('c=');
  Readln(c);
  D:=b*b-4*a*c;
  x1:=(-b+Sqrt(D))/(2*a);
```

```
    x2:=(-b-Sqrt(D))/(2*a);
    Writeln('x1=', x1:6:3);
    Writeln('x2=', x2:6:3);
    Writeln('(x1+x2)*a =', (x1+x2)*a:6:3, ' = -b');
    Writeln('x1*x2*a =', x1*x2*a:6:3, ' = c');
    Readln
end.
```

11

```
var A, B, C: Integer;
begin
 Write('Введите A: ');
 Readln(A);
 Write('Введите B: ');
 Readln(B);
 C:=A;
 A:=B;
 B:=C;
 Writeln('A=', A);
 Writeln('B=', B);
 Readln
end.
```

12*

```
var A, B: Integer;
begin
 Write('Введите A: ');
 Readln(A);
 Write('Введите B: ');
 Readln(B);
 A:=A+B;
 B:=A-B;
 A:=A-B;
 Writeln('A=', A);
 Writeln('B=', B);
 Readln
end.
```

Раздел «Условный оператор»

1

```
var Year: Integer;
begin
  Write ('Введите год: ');
  Readln(Year);
  if Year mod 4 > 0 then
    Write (Year, ' - обычный год.')
  else
    if Year mod 100 > 0 then
```

```
      Write (Year, ' - високосный год.')
    else
      if Year mod 400 > 0 then
        Write (Year, ' - обычный год.')
      else
        Write (Year, ' - високосный год.');
  Readln
end.
```

2

```
var N: Integer;
begin
  Write ('Введите N: ');
  Readln(N);
  if (N mod 2 = 0) AND ((N <= 10) OR (N >= 20)) then
    Write (Sqr(N))
  else
    Write (N);
  Readln
end.
```

3

```
var N: Integer;
begin
  Write ('Введите число: ');
  Readln(N);
  if N mod 2 = 0 then
    Writeln(N, ' - четное число.')
  else
    Writeln(N, ' - нечетное число.');
  Readln
end.
```

4

```
var N: Integer;
    D2, D3: Boolean;
begin
  Write ('Введите число: ');
  Readln(N);
  D2 := (N mod 2 = 0);
  D3 := (N mod 3 = 0);
  if D2 then
    Write (N, ' - четное число, ')
  else
    Write (N, ' - нечетное число, ');
  if D3 then
    Write ('делится на 3 ')
  else
    Write ('не делится на 3 ');
  if D2 and D3 then
```

```
    Writeln('и делится на 6')
  else
    Writeln('и не делится на 6');
  Readln
end.
```

5

```
var a, b, c, D, x, x1, x2: Real;
begin
  Writeln('Уравнение ax*x + bx + c = 0');
  Write('a=');
  Readln(a);
  Write('b=');
  Readln(b);
  Write('c=');
  Readln(c);
  D:=b*b-4*a*c;
  if D<0 then
    Writeln('Нет корней.')
  else
    if D=0 then
      begin
        x:=-b/(2*a);
        Writeln('x=', x:6:3)
      end
    else
      begin
        x1:=(-b+Sqrt(D))/(2*a);
        x2:=(-b-Sqrt(D))/(2*a);
        Writeln('x1=', x1:6:3);
        Writeln('x2=', x2:6:3);
      end;
  Readln
end.
```

6

```
var A, B: Integer;
begin
  Write ('Введите A: ');
  Readln(A);
  Write ('Введите B: ');
  Readln(B);
  if A < B then
    Write ('A + B = ', A+B)
  else
    Write ('A - B = ', A-B);
  Readln
end.
```

7

```
var N: Real;
begin
  Write ('Введите дальность выстрела: ');
  Readln(N);
  if (N>28) and (N<30) then
    Write ('ПОПАЛ')
  else
    Write ('НЕ ПОПАЛ');
  Readln
end.
```

8

```
var N: Real;
begin
  Write ('Введите дальность выстрела: ');
  Readln(N);
  if (N>28) and (N<30) then
    Write ('ПОПАЛ');
  if N>=30 then
    Write ('ПЕРЕЛЕТ');
  if N<=0 then
    Write ('НЕ БЕЙ ПО СВОИМ');
  if (N>0) and (N<=28) then
    Write ('НЕДОЛЕТ');
  Readln
end.
```

9

Способ 1:

```
var A, B: Real;
    Op: Char;
begin
  Write ('Введите A: ');
  Readln(A);
  Write ('Введите операцию (+ - * /): ');
  Readln(Op);
  Write ('Введите B: ');
  Readln(B);
  Write (A:7:3, ' ', Op, ' ', B:7:3, ' = ');
  if Op='+' then
    Write (A+B:7:3);
  if Op='-' then
    Write (A-B:7:3);
  if Op='*' then
    Write (A*B:7:3);
  if Op='/' then
    Write (A/B:7:3);
  Readln
end.
```

Второй способ показывает, как в Паскале можно сократить количество условных операторов в программе, используя оператор **Case**. Запишем только часть программы, заменяющую собой четыре условных оператора в предыдущем решении.

Способ 2:

```
case Op of
  '+': Write(A+B:7:3);
  '-': Write(A-B:7:3);
  '*': Write(A*B:7:3);
  '/': Write(A/B:7:3)
end;
```

Раздел «Циклы»

1

```
var I: Byte;
    Step: Longint;
begin
  Step:=1;
  for I:=1 to 16 do
    begin
      Step:=Step*2;
      Writeln(Step)
    end;
  Readln
end.
```

2

```
var F1, F2: Integer;
begin
  F1:=1;
  F2:=1;
  Writeln(F1);
  Writeln(F2);
  for I:=1 to 18 do
    begin
      Writeln(F1+F2);
      F2:=F1+F2;
      F1:=F2-F1;
    end;
  Readln
end.
```

3

```
const M=10;
var A, I, N: Integer;
begin
```

```
  N:=0;
  for I:=1 to M do
    begin
      Write ('Введите ', I, '-е число:');
      Readln(A);
      if A mod 2 = 0 then
        Inc (N)
    end;
  Writeln('Всего ', N, ' четных чисел');
  Readln
end.
```

4

```
const M=10;
var A, I, N1, N2: Integer;
begin
  N1:=0;
  N2:=0;
  for I:=1 to M do
    begin
      Write ('Введите ', I, '-е число:');
      Readln(A);
      if A > 0 then
        Inc (N1)
      else if A < 0 then
        Inc (N2)
    end;
  Writeln('Всего ', N1, ' положительных чисел');
  Writeln('Всего ', N2, ' отрицательных чисел');
  Writeln('Всего ', I-N1-N2, ' нулевых чисел');
  Readln
end.
```

5

```
var A, N: Integer;
begin
  N:=0;
  repeat
    Write ('Введите число (0 - выход):');
    Readln(A);
    if (A mod 2 = 1) and (A mod 5 > 0) then
      Inc (N)
  until A=0;
  Writeln('Всего ', N, ' нечетных чисел,
                        не делящихся на 5');
  Readln
end.
```

6

```
var A, N: Integer;
    S: Longint;
begin
  N:=0;
  S:=1;
  repeat
    Write ('Введите число (0 - выход):');
    Readln(A);
    if A in [1..9] then
      begin
        Inc (N);
        S:=S*A
      end;
  until A=0;
  Writeln('Всего ', N, ' чисел от 1 до 9,
           их произведение = ', S);
  Readln
end.
```

7

```
const N=10;
var A, I, Min, Max: Integer;
begin
  for I:=1 to N do
    begin
      Write ('Введите ', I, '-е число: ');
      Readln(A);
      if I=1 then
        begin
          Min:=A;
          Max:=A
        end;
      if A<Min then
        Min:=A;
      if A>Max then
        Max:=A
    end;
  Writeln('Max - Min = ', Max-Min);
  Readln
end.
```

8

```
var S, X, I: Integer;
begin
  S:=0;
  I:=1;
  repeat
```

```
    Write ('Введите ', I, '-е ', число: ');
    Readln(X);
     S:=S+X;
     I:=I+1
  until S>100;
  Writeln('Достаточно!');
  Writeln('Введено ', I-1, ' чисел');
  Readln
end.
```

9

```
var H, K: Char;
begin
  K:=0;
  repeat
    Write ('Введите символ: ');
    Readln(H);
    if H='F' then
      K:=K+1
  until K=5;
  Writeln('СТОП!');
  Readln
end.
```

10*

```
var H, K: Char;
begin
  K:=0;
  repeat
    Write ('Введите символ: ');
    Readln(H);
    if H='F' then
      K:=K+1
    else
      K:=0
  until K=5;
  Writeln('СТОП!');
  Readln
end.
```

11

```
var I, J, M, N: Integer;
begin
  Write ('M= ');
  Readln(M);
  Write ('N= ');
  Readln(N);
  for I:=1 to M do
```

```
    begin
      for J:=1 to N do
        Write ('*');
      Writeln;
    end;
  Readln
end.
```

12

А)
```
var N, I, J: Integer;
  begin
    Write ('N= ');
    Readln(N);
    for I:=1 to N do
      begin
        for J:=1 to N-I+1 do
          Write('*');
        Writeln
      end;
    Readln
  end.
```

Б) Приведем только тело цикла:
```
  for I:=1 to N do
    begin
      for J:=1 to I do
        Write('*');
      Writeln
    end
```

В) Чтобы во всех рядах ниже первого звездочки оказались на своем месте, ряд надо начать с пробелов, которые можно изобразить с помощью еще одного оператора цикла:
```
var N, I, J: Integer;
begin
  Write ('N= ');
  Readln(N);
  for I:=N downto 1 do
    begin
      for J:=1 to N-I do
        Write(' ');
      for J:=1 to I do
        Write('*');
      Writeln
    end;
  Readln
end.
```

13 **Способ 1:** организуем вывод с помощью двух циклов: перебор по строкам и печать чисел в строке.

```
var I, J, K: Integer;
  begin
    K:=10;
    for I:=1 to 9 do
      begin
        for J:=1 to 10 do
          begin
            Write (K:4);
            K:=K+1
          end;
        Writeln
      end;
    Readln
  end.
```

Способ 2: используем только один цикл, а переходить со строки на строку через каждые 10 цифр будем, проверяя остаток от деления нацело.

```
var k: Integer;
begin
  for K:=10 to 99 do
  begin
    Write(K:4);
      if (K mod 10) = 9 then
        Writeln
  end
end.
```

14

```
var N, Mul: Longint;
    Sum: Byte;
begin
  Write ('Введите число: ');
  Readln(N);
  Mul:=1;
  Sum:=0;
  repeat
    Mul:=Mul*(N mod 10);
    Sum:=Sum+(N mod 10);
    N:=N div 10
  until N=0;
  Writeln('Произведение цифр числа равно ', Mul);
  Writeln('Сумма цифр числа равна ', Sum);
  Readln
end.
```

Раздел «Массивы»

1

```
const N=10;
var A, B: array [1..N] of Integer;
    I: Byte;
begin
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  for I:=1 to N do
    B[I]:=A[N-I+1];
  for I:=1 to N do
    Writeln('B[', I, ']=', B[I]);
  Readln
end.
```

2

```
const N=10;
var A: array [1..N] of Integer;
    I: Byte;
    Ch, NCh: Longint;
begin
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  Ch:=0;
  NCh:=0;
  for I:=1 to N do
    if A[I] mod 2 = 0 then
      Ch:=Ch + A[I]
    else
      NCh:=NCh + A[I];
  Writeln('Четные - нечетные = ', Ch-NCh);
  Readln
end.
```

3 Приведем только тело второго цикла, считающего суммы. В остальном решение в точности совпадает с решением предыдущей задачи.

```
for I:=1 to N do
  if I mod 2 = 0 then
    Ch:=Ch + A[I]
  else
    NCh:=NCh + A[I]
```

4

```
const N=10;
var A: array [1..N] of Integer;
    I: Byte;
    S: Boolean;
begin
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  S:=True;
  for I:=1 to (N div 2) do
    if A[I] <> A[N-I+1] then
      S:=False;
  if S then
    Writeln('Массив симметричный')
  else
    Writeln('Массив не симметричный');
  Readln
end.
```

5

```
const N=10;
var A: array [1..N] of Integer;
    I, J: Byte;
    S: Boolean;
begin
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  S:=True;
  for I:=1 to N-1 do
    for J:=I+1 to N do
      if A[I] = A[J] then
        S:=False;
  if S then
    Writeln('Все элементы разные')
  else
    Writeln('Есть повторяющиеся элементы');
  Readln
end.
```

6

```
const N=10;
var A: array [1..N] of Integer;
```

```
    I, M: Byte;
    S: Longint;
begin
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  S:=0;
  M:=0;
  for I:=1 to N do
    if A[I] < 0 then
      begin
        S:=S + A[I];
        Inc (M)
      end;
  if M>0 then
    Writeln('Искомое число = ', S/M:7:3)
  else
    Writeln('Все числа в массиве неотрицательные');
  Readln
end.
```

7 Во многих задачниках приводится решение этой задачи за два прохода: сначала ищется максимум, потом подсчитывается, сколько чисел ему равны. Это неэффективное решение. Посмотрите, как легко оба этих действия можно выполнить за один просмотр массива.

```
const N=10;
var A: array [1..N] of Integer;
    I, M: Byte;
    Max: Integer;
begin
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  Max:=A[1];
  M:=1;
  for I:=2 to N do
    if A[I]>Max then
      begin
        Max:=A[I];
        M:=1
      end
```

```
    else if A[I]=Max then
            Inc(M);
  Writeln('Максимум = ', Max,
          ', всего таких чисел: ', M);
  Readln
end.
```

8

```
const N=10;
var A: array [1..N] of Integer;
    I, M, M1: Byte;
    Max: Integer;
begin
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  Max:=A[1];
  M:=1;
  M1:=1;
  for I:=2 to N do
    if A[I]>Max then
      begin
        Max:=A[I];
        M:=I;
        M1:=I
      end
    else if A[I]=Max then
            M1:=I;
  Writeln('Первый максимум A[', M, '],
           последний максимум A[', M1, ']');
  Readln
end.
```

9

```
const N=10;
var A: array [1..N] of Integer;
    I, M: Byte;
    Max: Integer;
begin
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  M:=0;
  if A[1]>A[2] then
```

```
    Inc(M);
  if A[N]>A[N-1] then
    Inc(M);
  for I:=2 to N-1 do
    if A[I]>A[I-1]+A[I+1] then
      Inc(M);
  Writeln('Количество искомых чисел = ', M);
  Readln
end.
```

10* **Способ 1.**

```
const N=10;
var A: array [1..N] of Integer;
    M: Byte;
begin
  for M:=1 to N do
    begin
      Write ('Введите A[', M, ']: ');
      Readln(A[M])
    end;
  M:=1;
  while (A[N]*A[N-M]>0) and (M<N-1) do
    Inc(M);
  if (M=N-1) and (A[N]*A[1]>0) then
    Inc(M);
  if A[N]=0 then
    Writeln('последнее число 0')
  else
    begin
      Write ('Количество чисел одного знака
              в конце = ', M, ', ');
      if A[N]>0 then
        Writeln('знак +');
      if A[N]<0 then
        Writeln('знак -');
    end;

  Readln
end.
```

Способ 2.

Как говорилось выше, многие задачи, предложенные в этом разделе, можно решить без использования массива. Покажем, как это делается, на примере этой задачи. Проверка производится во время ввода.

```
const N=10;
var I, M: Byte;
```

```
    A, C: Integer;
begin
  M:=0;
  for I:=1 to N do
    begin
      Write ('Введите ', I, '-е число: ');
      Readln(A);
      if I=1 then
        C:=A;
      if A*C>0 then
        Inc(M)
      else
        begin
          C:=A;
          M:=1
        end
    end;
  if A=0 then
    Writeln('последнее число 0')
      else
        begin
          Write ('Количество чисел одного знака
                  в конце = ', M, ', ');
          if A>0 then
          Writeln('знак +');
          if A<0 then
          Writeln('знак -')
        end;
  Readln
end.
```

11 В операторе вывода воспользуемся так называемым *форматным выводом* – укажем, сколько позиций требуется отвести для выводимого значения (в случае вещественного числа требуется также указать, сколько цифр напечатать после запятой).

```
var M: Array [1..12]of Integer;
var j: Integer;
    S: Real;
begin
  Writeln('Ввод масива');
  for J:=1 to 12 do
    begin
      Write ('Введите кол-во осадков в ', J,
             '-м месяце: ');
      Readln(M[j])
    end;
```

```
  S:=0;
  for J:=1 to 12 do
    S:=S+M[j]; {Нашли общегодовое кол-во осадков}
  S:=S/12; {Среднегодовое кол-во осадков}
  Writeln('N':3, 'кол-во осадков':15,
   'отклонение':15); {Напечатали заголовок таблицы}
  for J:=1 to 12 do
    Writeln(J:3, M[J]:15, M[J]-S:15:1);
  Readln
end.
```

12 При решении задачи надо попарно сравнивать символы, взятые с начала и с конца, а остановиться, очевидно, посередине. Однако в отличие от предыдущей задачи здесь не всегда надо идти до конца цикла: как только обнаружено первое несовпадение букв, проверку можно прекратить. Результаты проверки мы будем хранить в логической переменной-флаге *T*, которая равна TRUE, если на данном этапе несовпадений букв не выявлено, и становится равной FALSE, как только проверяемые буквы не совпадут.

Еще одно замечание – относительно описания и ввода массива. Мы не знаем, из скольких букв будет состоять слово, поэтому массив приходится описывать «с запасом» – на 30 элементов, использоваться же будут только *N* первых введенных букв.

В Паскале при вводе слово можно записать целиком (не забыв точку в конце!). Оператор READ будет за каждый проход цикла считывать из него по одной букве.

```
var Slovo: Array [1..30] of Char;
    J, N: Integer;
       T: Boolean;
begin
  Writeln('Введите слово по буквам, в конце - точка');
  N:=0;
  repeat
    N:=N+1;
    Read (Slovo[N])
  until Slovo[N]='.' ;
  Writeln;
  N:=N-1; {слово введено и подсчитано
           количество букв в слове}
  T:=TRUE;
  for J:=1 to N div 2 do
    if Slovo[J]<>Slovo[N-J+1] then
      begin
        T:=False;
        Break
      end;
```

```
  if not T then
    Write ('НЕ ');
  Writeln('палиндром.');
  Readln
end.
```

13 Число в заданной системе счисления вводится последовательно по цифрам и сразу производится его перевод в десятичную систему. Если не печатать число в заданной системе счисления, то задачу можно решить без использования массива.

```
var C: array [1..20] of Integer;
    I, K, C10, Sis: Integer; { Sis - система
                                        счисления}
begin
  Write ('Основание системы счисления ');
  Readln(Sis);
  Write ('Количество цифр в числе: ');
  Readln(K);
  Writeln('Цифры числа в ', 'Sis',
          '-ичной системе счисления');
  C10:=0;
  for i:=1 to k do
    begin
      Write (I, '-я цифра: ');
      Readln(C[i]);
      C10:=C10*Sis+C[i]
    end;
  for I:=1 to K do
    Write (C[i]);  {Напечатали введенное число}
  Write (' в ', sis, '-ичной системе  = ');
  Write (C10);
  Writeln('  в десятичной');
  Readln
end.
```

14 Вспомним алгоритм перевода чисел из десятичной системы в Q-ичную. Цифрами числа (начиная с последней) будут остатки от деления числа на Q, а число на каждом шаге будет нацело делиться на Q.

Так как цифры числа будут вычисляться в обратном порядке, сразу их печатать нельзя: придется сначала сохранить их в массиве. Причем этот массив надо заполнять с последней (крайней справа) цифры справа налево. При этом важно подсчитать, сколько позиций в массиве будет заполнено, так как на экран надо выводить только вычисленные цифры числа, а не «мусор», который хранится в незаполненных в программе ячейках массива.

Заметим, что ограничение $Q < 10$ связано только с тем, что для систем счисления с основанием, превышающим 10, пришлось бы определять дополнительные символы-цифры; алгоритм же останется без изменений.

```
var Q, N10, I, J: Integer;
    NQ: array [1..20] of Integer;
begin
  Write('Основание системы счисления - ');
  Readln(Q);
  Write('Число в 10-й системе - ');
  Readln(N10);
  I:=20;
  repeat
    NQ[I]:=N10 mod Q;
    I:=I-1;
    N10:=N10 div Q;
  until N10=0;
  for J:=I+1 to 20 do
    Write( NQ[J]);
  Writeln
end.
```

15

```
const N=10;
type Massiv= Array [1..N] of Real;
Var M, Pol, Otr: Massiv;
    J, KPol, Kotr: Integer;
procedure VvodMas (var M: Massiv; N: Integer);
var J: Integer;
  begin
    Writeln('Ввод массива');
      for J:=1 to N do
        begin
          Write ('Введите ', j, '-й элемент: ');
          Readln(M[j])
        end
  end;
procedure PrintMas (var M: Massiv; N: Integer);
var J: Integer;
  begin
    for J:=1 to N do
       Write (M[J], '  ');
    Writeln
  end;
begin
  VvodMas (M, N);
  KPol:=0;
```

```
  KOtr:=0; {в обоих массивах пока нет элементов}
  for J:=1 to N do
    begin
      if M[j]>0 then
        begin
          KPol:=KPol+1;
          Pol[KPol]:=M[J]
        end;
      if M[J]<0 then
        begin
          KOtr:=KOtr+1;
          Otr[KOtr]:=M[J]
        end
    end;
  Writeln('Исходный массив');
  PrintMas (M,N);
  Writeln('Положительные числа');
  PrintMas (Pol, KPol);
  Writeln('Отрицательные числа');
  PrintMas (Otr, KOtr);
  Readln
end.
```

16 Сохраним первый элемент в переменной *X*, произведем сдвиг элементов и запишем *X* на последнее место.

```
const N=10;
type Massiv= Array [1..N] of Integer;
var M: Massiv;
    j, X: Integer;
procedure VvodMas (var M: Massiv; N: Integer);
var J: Integer;
begin
  Writeln('Ввод массива');
  for J:=1 to N do
    begin Write ('Введите ', j, '-й элемент: ');
        Readln(M[j])
  end
end;
procedure PrintMas (var M: Massiv; N: Integer);
var J: Integer;
begin
  for J:=1 to N do
    Write(M[J],'  ');
  Writeln
end;
begin
```

```
  VvodMas (M,N);
  X:=M[1];
  for J:=1 to N-1 do
    M[J]:=M[J+1];
  M[N]:=X;
  PrintMas (M,N);
  Readln
end.
```

17 Надо подсчитать суммы правой и левой частей и сравнить их. Некоторую сложность представляет разделение массива на две части, потому что такое разделение зависит от того, четно ли количество его элементов. Воспользуемся операцией поиска остатка от деления MOD: если $N \bmod 2 = 0$, значит, N – четное.

```
const N=8;
type Massiv= Array [1..N] of Integer;
var M: Massiv;
    K, j, Sl, Sp: Integer;
procedure VvodMas (var M: Massiv; N: Integer);
var J: Integer;
begin
  Writeln('Ввод массива');
  for J:=1 to N do
    begin
      Write ('Введите ', j, '-й элемент: ');
      Readln(M[j])
    end
end;
procedure PrintMas (var M: Massiv; N: Integer);
var J: Integer;
begin
  for J:=1 to N do
    Write (M[J], '  ');
  Writeln
end;
begin
  VvodMas (M, N);
  PrintMas (M,N);
  Sl:=0;
  Sp:=0; {В начале суммы правой и левой частей -
          нулевые}
  for J:=1 to N div 2 do
    Sl:=Sl+M[j];  {Найдена сумма левой части}
  if N mod 2 =0 then
    K:=N div 2 + 1
  else
```

```
    K:=N div 2 + 2;
  for J:=K to N do
    Sp:=Sp+M[j];  {Найдена сумма правой части}
  if Sl=Sp then
    Writeln('билет счастливый')
  else
    Writeln('билет несчастливый')
end.
```

Раздел «Строки»

1 При решения этой задачи главное помнить, что строка – это массив символов и к ней можно обращаться поэлементно, как к любому другому массиву.

```
var S: string;
    I, N: Byte;
begin
  Write('Введите текст: ');
  Readln(S);
  N:=0;
  for I:=1 to Length(S) do
    if (S[I]='a') or (S[I]='A') then
      Inc(N);
  Writeln('Всего ', N, ' букв A');
  Readln
end.
```

2

```
var S: string;
    I, N: Byte;
begin
  Write('Введите текст: ');
  Readln(S);
  N:=0;
  for I:=1 to Length(S) do
    if S[I] in ['0'..'9'] then
      Inc(N);
  Writeln('Всего ', N, ' цифр в тексте');
  Readln
end.
```

3

```
var S: string;
    I, N: Byte;
begin
  Write('Введите текст: ');
  Readln(S);
  N:=0;
```

```
  for I:=1 to Length(S) do
    if (S[I] in ['a'..'z'])
    or (S[I] in ['A'..'Z']) then    Inc(N);
  Writeln('Всего ', N, ' латинских букв в тексте');
  Readln
end.
```

4

```
var S, S1: string;
    I: Byte;
begin
  Write('Введите: ');
  Readln(S);
  S1:='';
  for I:=1 to Length(S) do
    if S[I]<>' ' then
      S1:=S1+S[I];
  Writeln('Текст без пробелов: ', S1);
  Readln
end.
```

5

В этой задаче главное – не забыть все символы, которые могут использоваться в «обычном» тексте, и рационально проверить условие. В Паскале его можно записать в одну строку, пусть и в длинную, используя множества.

```
var S, S1: string;
    I: Byte;
begin
  Write('Введите текст: ');
  Readln(S);
  S1:='';
  for I:=1 to Length(S) do
    if S[I] in ['a'..'z', 'A'..'Z', ' ', '.',
',', ';', '!', '?', ':', '"', '-', '(', ')'] then
      S1:=S1+S[I];
  Writeln('Очищенный текст: ', S1);
  Readln
end.
```

6

При решении этой задачи нам понадобятся функции **Ord** и **Chr**, которые, соответственно, возвращают порядковый номер элемента перечислимого типа данных и символ с данным порядковым номером.

Заметим, что даже если мы не знаем точный порядок символов в таблице ASCII, мы можем с уверенностью сказать, что латинские символы (как заглавные, так и строчные) следуют по порядку от `'A'` до `'Z'`. Этот факт и используется в решении данной задачи. В любом случае учить к экзамену коды символов не требуется: в ЕГЭ нет задач, где бы это было необходимо.

```
var S, S1: string;
    I: Byte;
begin
  Write('Введите строку: ');
  Readln(S);
  S1:='';
  for I:=1 to Length(S) do
    if S[I] in ['A'..'Z'] then
      S1:=S1+Chr(Ord(S[I])+(Ord('a')-Ord('A')))
    else
      S1:=S1+S[I];
  Writeln(S1);
  Readln
end.
```

7

```
var I, J: Byte;
begin
  for I:=0 to 15 do
    begin
      for J:=0 to 15 do
        Write(Chr(16*I+J));
      Writeln
    end;
  Readln
end.
```

8

```
var S, S1: string;
    I, N: Byte;
begin
  Write('Введите строку:');
  Readln(S);
  I:=1;
  while S[I]=' ' do
    Inc(I); {считаем пробелы в начале строки}
  S:=Copy(S, I, 255); {убрали пробелы в начале}
  N:=Pos(' ', S); {нашли позицию следующего пробела}
  if N>0 then
    S1:=Copy(S, 1, N-1) {взяли первое слово}
  else
    S1:=S;
  Writeln('Первое слово - "', S1, '"');
  Readln
end.
```

9 В решении, приведенном ниже, обратите внимание на строку
`while (S[I]<>' ') and (I>0) do`
Очевидно, цикл должен остановиться, когда мы встретим пробел, либо когда *I* станет равным 0. В момент *I* = 0 значение *S*[*I*] = *S*[*0*]. Но что такое нулевой символ строки? Имеем ли мы право так писать? Да, имеем. В первом байте строки, т.е. в *S*[0], всегда хранится код фактической длины строки. Таким образом, обращение *S*[0] правомерно и не вызовет ошибку при выполнении программы.

```
var S, S1: string;
    I: Byte;
begin
  Write('Введите строку:');
  Readln(S);
  I:=Length(S);
  while S[I]=' ' do
    Dec(I);          {считаем пробелы в конце строки}
  S:=Copy(S, 1, I); {убрали пробелы в конце}
  S1:='';
  while (S[I]<>' ') and (I>0) do
    begin
      S1:=S[I]+S1;  {приписываем по одной букве}
                    {в начало слова}
      Dec(I)
    end;
  Writeln('Последнее слово - "', S1, '"');
  Readln
end.
```

10 Для решения этой задачи воспользуемся следующей хитростью. Заменим все символы строки, отличающиеся от пробела, на символ «*», припишем в конце строки еще один пробел и будем искать количество вхождений в текст пары символов «* » («звездочка» и пробел): это и будет искомое количество слов в тексте.
Отметим, что длина строки *S* в этом случае не может превышать 254 символа.

```
var S: string [254];
    S1: string;
    I, N: Byte;
begin
  Write('Введите строку:');
  Readln(S);
  S1:='';
  for I:=1 to Length(S) do
```

```
      if S[I]=' ' then
        S1:=S1+S[I]
      else
        S1:=S1+'*';
    S1:=S1+' ';
    N:=0;
    while Pos('* ', S1)>0 do
      begin
        Inc(N);
        S1:=Copy(S1, Pos('* ', S1)+1, 255)
      end;
    Writeln('Всего ', N, ' слов');
    Readln
  end.
```

11

```
var S: string;
    I: byte;
    C: Char;
begin
  Write('Введите строку:');
  Readln(S);
  I:=0;
  repeat
    Inc(I);
    C:=S[I];
    if C in ['A'..'Z'] then
      begin
        Insert(C, S, I);
        Inc(I)
      end;
  until I>=Length(S);
  Writeln('Измененная строка:', S);
  Readln
end.
```

Раздел «Файлы»

1

```
 var F: file of Integer;
    A, I, N :Integer;
begin
  Assign (F, 'xx.num');
  Write ('Сколько чисел?');
  Readln(N);
  ReWrite (F);
  for I:=1 to N do
```

```
    begin
      Write (i, '-е число: ');
      Readln(A);
      Write (F, A)
    end;
  Close (F);
  Readln
end.
```

2

```
var F: file of Integer;
    A, I: Integer;
begin
  Assign (F, 'xx.num');
  Reset (F);
  while not Eof(F) do
    begin
      Read (F, A);
      Write (A, ' ')
    end;
  Writeln;
  Close(F);
  Readln
end.
```

3 Как и во многих задачах на проверку, нам здесь понадобится логическая переменная. Вначале она равна TRUE и станет равной FALSE, если будет обнаружено несовпадение элементов. Будем считывать из файлов элементы, пока не закончится один из них или пока не обнаружится неравенство в какой-то паре. По окончании цикла не забудем, что возможен случай, когда начала файлов совпадают, но один заканчивается раньше другого; именно поэтому в конце проверяется сразу три условия.

```
var F1, F2:  file of Char;
  A, B: Char;
  T: Boolean;
begin
  Assign (F1, 'aa.txt');
  Assign (F2, 'bb.txt');
  Reset (F1);
  Reset (F2);
  T:=True;
  while (Not Eof(F1)) and (Not Eof(F2)) and T do
    begin
      Read (F1,A);
      Read (F2,B);
```

```
      if A<>B then
        T:=False
    end;
  if T and Eof(F1) and Eof(F2) then
    Writeln('Файлы одинаковы')
  else
    Writeln('Файлы различны');
  Readln
end.
```

4 Числа надо считывать по два, а печатать только второе. При этом надо обязательно убедиться, что файл не закончился и число существует.

```
var F: file of Real;
    X: Real;
begin
  Assign (F,'xx.num');
  Reset (F);
  while not Eof(F) do
    begin
      Read (F, X);
      if not Eof(F) then
        begin
          Read (F,X);
          Write (X:6:1, '  ')
        end
    end;
  Readln
end.
```

5 Необходимо предусмотреть случай, когда строки с указанным номером в файле нет. Все строки до заданного номера (или до конца файла) прочитываем в одну и ту же переменную. В результате в ней остается последняя из прочитанных строк.

```
var TT: Text;
    S: string;
    I, K: Integer;
begin
  Assign (TT, 'Slova.txt');
  Reset (TT);
  Write ('Какую строчку хотите увидеть? ');
  Readln(K);
  I:=0;
  while not Eof(TT) and (I<K) do
    begin
      Readln(TT, S);
      I:=I+1
```

```
    end;
  if I=K then
    Writeln(S)
  else
    Writeln('Строки с таким номером нет')
end.
```

6 Добавить информацию в читаемый файл нельзя, поэтому создадим дополнительный файл, в него будем записывать фамилию из файла **Famil.txt** и имя с клавиатуры. Потом закроем оба файла, откроем их уже в другом режиме и перепишем обновленные строки из дополнительного файла в заданный.

```
var TT, Dop: Text;
    Fam, Im: string;
begin
  Assign (TT, 'Famil.txt');
  Reset (TT);
  Assign (Dop, 'Dop.txt');
  Rewrite (Dop);
  While Not Eof(TT)  do
    begin
      Readln(TT,Fam);
      Write ('Ведите имя ',Fam,': ');
      Readln(Im);
      Writeln(Dop, Fam, ' ', Im)
    end;
  Close (Dop);
  Close (TT);
  Rewrite (TT);
  Reset (Dop);
  while not Eof(Dop) do
    begin
      Readln(Dop, Fam);
      Writeln(TT, Fam)
    end;
  Close (Dop);
  Close (TT)
end.
```

Раздел «Процедуры и функции»

1 Далее приводятся рекомендации по решению каждой задачи из пяти предложенных.

1. В этой задаче требуется вводить координаты вектора, которые должны заполнить трехмерный массив. Так как мы договорились перед именем

массива в любом случае ставить **Var**, описание параметров в заголовке процедур получается одинаковым, несмотря на то, создается ли массив процедурой или она работает с уже имеющимся массивом.

2. Даны два вектора, надо найти третий. Все массивы описываются одинаково. Результатом работы является вектор – массив (не одно значение, а много), поэтому для решения задачи используем процедуру (массив не может быть результатом функции).
3. Дано – массив, найти – длину (число). Значит, придется использовать функцию. Ее тип – **real**, так как при вычислении квадратного корня получается вещественный результат.
4. Дано – два массива, найти – некоторое число по формуле (одно значение). Используем функцию. Обратите внимание, что для нахождения суммы требуется ввести локальную переменную *S*, найти ее значение и присвоить его имени функции. Без этого обойтись нельзя, иначе имя функции пришлось бы писать в правой части оператора присваивания.
5. Дано – два массива, найти – число. Для нахождения заданного значения воспользуемся уже описанными выше функциями.
   Обратите внимание на вызовы процедур и функций. Вместо параметров, которые описаны с **Var**, можно подставлять только переменные; вместо остальных параметров – и переменные, и значения, и выражения. Обращение к процедуре записывается как отдельный оператор, обращение же к функции должно быть частью другого оператора (присваивания, вывода).

```
const N=3;
type Vect=array [1..N] of Integer;
{************************************Ввод вектора}
procedure Vvod (Var A: Vect);
var I: Integer;
begin
  Writeln('Ввод вектора');
  for I:=1 to N do
    begin
      Write ('Введите ', I, '-й элемент: ');
      Readln(A[i])
    end
end;
{************************************Печать вектора}
procedure Print (var A: Vect);
var I: Integer;
begin
  for I:=1 to N do
    Write (A[I]:8);
  Writeln
end;
```

```
{************************C=A+B*******Сумма векторов}
procedure Summa (var A, B, C: Vect);
var I: Integer;
begin
  for I:=1 to N do
    C[i]:=A[i]+B[i]
end;
{************************************Длина вектора}
function Length (var A: Vect): Real;
var I, S: Integer;
begin
  S:=0;
  for I:=1 to N do
    S:=S+Sqr(A[I]);
  Length:=Sqrt(S)
end;
{***************************Скалярное произведение}
function Skapr (var A, B: Vect): Integer;
var I, S: Integer;
begin
  S:=0;
  for I:=1 to N do
    S:=S+A[i]*B[i];
  Skapr:=S
end;
{************************************Косинус угла}
function CosUg (var A, B: Vect): Real;
begin
  CosUg:=SkaPr(A,B)/(Length(A)*Length(B))
end;
var X, Y, Z: Vect;
begin
  Vvod(X);
  Vvod(Y);
  Summa(X,Y,Z);
  Writeln('Сумма векторов');
  Print(X);
  Print(Y);
  Writeln('+_____________________________');
  Print(Z);
  Writeln('Длина этого вектора ', Length(Z):8:2);
  Writeln('их скалярное произведение ',Skapr(X,Y));
  Writeln('косинус угла между ними =',
           CosUg (X,Y):8:1)
end.
```

2 Для описания числа введем тип данных NUM – массив из 20 цифр.

1. Процедура похожа на процедуру ввода вектора, только надо учесть, что вводятся не все элементы массива, а нужное их количество. Оставшиеся элементы надо обнулить.
2. При выводе сначала просмотрим число до первой значащей (>0) цифры. Учтем, что таковой может и не оказаться (если число равно 0). Оставшиеся цифры печатаем.
3. Дано – число, найти – четное оно или нет. Такого типа функции (дающие ответ «да» или «нет») обычно пишутся как функции типа **boolean**, где тело функции состоит из одного оператора.
4. Для определения делимости на 3 надо найти суму цифр числа. Обратите внимание на последний оператор: булевской переменной присваивается значение условного (булевского) выражения.
5. Сумму будем вычислять «в столбик», при этом начинаем с последней цифры. Если сумма двух цифр больше 10, надо 1 «запомнить в уме» и на следующем шаге прибавить к сумме цифр (в качестве «ума» выступает переменная *Z*). Суммируем все цифры числа, даже незначащие: это не повлияет на результат.
   Следует учесть, что если результатом будет число из 21-й цифры, то программа его вычислит неправильно.

```
type Num=Array [1..20] of 0..9;
{*************************************Ввод числа}
procedure Vvod (var A: Num);
var I, N: Integer;
begin
  Write ('Сколько цифр в числе? ');
  Readln(N);
  Writeln('Введите число по цифрам:');
  for I:=20-N+1 to 20 do
    begin
      Write ('Введите ', I-20+N, '-ю цифру: ');
      Readln(A[I])
    end;
  for I:=1 to 20-N do
    A[I]:=0
end;
{*************************************Печать числа}
procedure Print (var A: Num);
var I, J: Integer;
begin
  I:=1;
```

```
  while (A[i]=0) and (I<20) do
    I:=I+1;
  for J:=I to 20 do
    Write (A[J]);
  Writeln
end;
{************************************Четность числа}
function Chetn (var A: Num): Boolean;
begin
  Chetn:=A[20] mod 2 = 0
end;
{************************************Делимость на 3}
function Del3 (var A: Num): Boolean;
var I, S: Integer;
begin S:=0;
  for I:=1 to 20 do
    S:=S+A[i];
  Del3:= S mod 3 = 0
end;
{***************************C=A+B*******Сумма чисел}
procedure Summa (var A, B, C: Num);
var I, Z, R: Integer;
begin
  Z:=0;
  for I:=20 downto 1 do
  begin
    R:=A[i]+B[i]+Z;
    if R>9 then
      begin
        C[i]:=R-10;
        Z:=1
      end
    else
      begin
        C[i]:=R;
        Z:=0
      end
  end
end;
var X, Y, Z: Num;
begin
  Vvod(X);
  if Chetn(X) then
    Writeln('четное');
  if Del3(X) then
    Writeln('делится на 3');
```

```
  Vvod(Y);
  Summa (X, Y, Z);
  Writeln('Сумма чисел');
  Print(X);
  Print(Y);
  Write('равна ');
  Print(Z);
  Readln
end.
```

3 Базой рекурсии (нерекурсивной ветвью функции) является значение N=1, рекурсивный шаг определяется по формуле факториала: $N! = (N-1)! * N$. При работе с предлагаемой программой следует иметь в виду, что если ввести $N \leq 0$, то программа работать не будет: возникнет ошибка «Переполнение стека», так как рекурсия не сможет остановиться.

```
function Fakt (N: Integer): LongInt;
begin
  if N=1 then
    Fakt:=1
  else
    Fakt:=Fakt(N-1)*N
end;
var N: Integer;
begin
  Write ('Введите N: ');
  Readln(N);
  Writeln(N, '! = ', Fakt(N))
end.
```

4 В этой задаче нерекурсивная ветвь – это вычисление суммы цифр однозначного числа. Для шага рекурсии воспользуемся тем, что остаток от деления числа на 10 равен его последней цифре, а целая часть – числу без последней цифры.

```
function SCif (N: Integer): Integer;
begin
  if N<10 then
    SCif:=N
  else SCif:=SCif(N div 10)+ (N mod 10)
end;
var N: Integer;
begin
  Write ('Введите N: ');
  Readln(N);
  Writeln('Сумма цифр числа ', N, ' равна ', Scif(N))
end.
```

5 Окончание рекурсии – ввод точки. В этом случае делать ничего не надо, поэтому нерекурсивная ветвь в этой задаче пустая (но она должна присутствовать – рекурсия выполняется не в любом случае, а только если введенный символ не равен точке). Обратите внимание, что приглашение «Введите слово с точкой в конце» написано в основной программе, а не в рекурсивной процедуре, иначе оно бы повторялось столько раз, сколько букв в слове. И еще одно замечание: если в процедуре операторы **Print** и **Write(H)** поменять местами, то слово будет печататься в прямом порядке.

```
procedure Print;
var H: Char;
begin
  Read (H);
  if H<>'.' then
    begin
      Print;
      Write(H);
    end
end;
begin
  Writeln('Введите слово с точкой в конце: ');
  Print;
  Writeln;
end.
```

6

```
var I, N: Integer;
function Fib (N: Byte): Longint;
begin
  if (N=1) or (N=2) then
    Fib:=1
  else
    Fib:=Fib(N-1)+Fib(N-2)
end;
begin
  Write ('Enter N: ');
  Readln(N);
  for I:=1 to N do
    Writeln('F(', I, ')=', Fib(I));
  Readln
end.
```

7 Во всех задачах нерекурсивной ветвью является простейший случай. В первых двух задачах – это пустая последовательность (тогда сумма равна 0). При вводе с клавиатуры – это ввода числа 0, в случае файла – это пустой файл. Массив же пустым не бывает, поэтому здесь простейшим

случаем будет нахождение суммы в массиве, состоящем из одного элемента ($N = 1$).

Для осуществления рекурсивного шага надо к имеющемуся элементу прибавить суму оставшихся (которая вычисляется рекурсивно).

1.

```
function SumP: Integer;
var A: Integer;
begin
  Write ('Число: ');
  Readln(A);
  if A=0 then
    SumP:=0
  else
    SumP:= SumP+A
end;
begin
  Writeln('Вводите числа. Признак конца - 0.');
  Writeln(SumP, '  -  это сумма.')
end.
```

2.

```
type FF=file of Integer;
procedure ZapF (var F: FF);
var N, A, I: Integer;
begin
  Write ('Сколько чисел? ');
  Readln(N);
  ReWrite (F);
  for I:=1 to N do
    begin
      Write (I, '-е число: ');
      Readln(A);
      Write (F, A)
    end;
  Close (F)
end;

function SumF (var F: FF): Integer;
var A: Integer;
begin
  Read (F,A);
  if Eof(F) then
    SumF:=0
  else
    SumF:=Sumf(F)+A
end;
```

```
var F: FF;
begin
  Assign (F, 'F.pas');
  Zapf(F);
  Reset (F);
  Writeln('Сумма равна ',Sumf(F))
end.
```

3.

```
type Mas=Array [1..30] of Integer;
procedure ZapM (var M: Mas; var N: Integer);
var I: Integer;
begin
  Write ('Сколько чисел? ');
  Readln(N);
  for I:=1 to N do
    begin
      Write (I,'-е число: ');
      Readln(M[I])
    end
end;

function SumM (var M: Mas; N: Integer): Integer;
begin
  if N=1 then
    SumM:=M[1]
  else SumM:=SumM(M,N-1)+M[N]
end;

var M: Mas;
    N:Integer;
begin
  ZapM(M,N);
  Writeln('Сумма равна ',SumM(M,N))
end.
```

8*

```
var S, S1: string;
    N, M: Byte;

function MyLength(var S: string): Byte;
begin
  MyLength:=Ord(S[0])
end;

function MyConcat(var S, S1: string): string;
var Temp: string;
```

```
    L, L1, I: Byte;
begin
  Temp:=S;
  L:=MyLength(S);
  L1:=MyLength(S1);
  if L+L1>255 then
    Dec(L1, L+L1-255);
  Temp[0]:=Temp[0] + L1;
  for I:=1 to L1 do
    Temp[L+I]:=S1[I];
  MyConcat:=Temp
end;

function MyCopy(var S: string; N, M: Byte): string;
var Temp: string;
    L, I: Byte;
begin
  L:=MyLength(S);
  if N+M-1<L then
    L:=N+M-1
  else if N>L then
         begin
           MyCopy:='';
           Exit
         end;
  Temp[0]:=Chr(L-N+1);
  for I:=N to L do
    Temp[I-N+1]:=S[I];
  MyCopy:=Temp
end;

procedure MyInsert(var S1, S: string; N: Byte);
var L, L1, I: Byte;
begin
  L:=MyLength(S);
  if N>=L then
    S:=MyConcat(S, S1)
  else
    begin
      L1:=MyLength(S1);
      if L+L1>255 then
        Dec(L1, L+L1-255);
      S[0]:=S[0] + L1;
      L:=L+L1;
      for I:=1 to L-N do
        S[L-I+1]:=S[L-L1-I+1];
```

```
      for I:=1 to L1 do
        S[N+I]:=S1[I]
    end
end;

begin
  Write('Enter S :');
  Readln(S);
  Write('Enter S1:');
  Readln(S1);
  Write('Enter N: ');
  Readln(N);
  Write('Enter M: ');
  Readln(M);
  Writeln('MyLength(''', S, ''')=', MyLength(S));
  Writeln('MyConcat(''', S, ''', ''', S1, ''')=''',
          MyConcat(S, S1), '''');
  Writeln('MyCopy(''', S, ''', ', N, ', ', M,
         ')=''', MyCopy(S, N, M), '''');
  Write('MyInsert(''', S, ''', ''', S1, ''', ', N,
       ')=''');
  MyInsert(S1, S, N);
  Write(S, '''');
  Readln
end.
```

Раздел «Смешанные задачи»

1 Как вы можете увидеть в решении ниже, иногда результаты работы рекурсии приходится «выправлять» вручную в граничных случаях (например, когда пользователь введет пустую строку).

```
var S: string;
    N: Byte;
procedure Count(S: string);
var I, T: Byte;
begin
  I:=1;
  Inc(N);
  while (S[I]=' ')
  and (I<=Length(S)) do
    Inc(I);
  S:=Copy(S, I, 255);
  T:=Pos(' ', S);
  if T>0 then
```

```
    begin
      S:=Copy(S, T+1, 255);
      Count(S)
    end
end;
begin
  Write('Введите строку:');
  Readln(S);
  N:=0;
  Count(S);
  if (S[Length(S)]=' ') or (S='') then
    Dec(N);
  Writeln('Всего ', N, ' слов');
  Readln
end.
```

2

```
var A, B, C, Cos1, Cos2, Cos3: Real;
begin
  Write('Введите сторону A: ');
  Readln(A);
  Write('Введите сторону B: ');
  Readln(B);
  Write('Введите сторону C: ');
  Readln(C);
  Cos1:=(B*B+C*C-A*A)/(2*B*C);
  Cos2:=(A*A+C*C-B*B)/(2*A*C);
  Cos3:=(A*A+B*B-C*C)/(2*A*B);
  if (Abs(Cos1)>=1) or (Abs(Cos2)>=1) or
      (Abs(Cos3)>=1) then
    Writeln('Не может быть такого треугольника')
  else
    begin
      if (A=B) and (B=C) then
        Writeln ('Равносторонний треугольник')
      else if (A=B) or (B=C) or (A=C) then
             Writeln('Равнобедренный треугольник');
      if (Cos1<0) or (Cos2<0) or (Cos3<0) then
        Writeln('Тупоугольный треугольник')
      else if (Cos1=0) or (Cos2=0) or (Cos3=0) then
             Writeln('Прямоугольный треугольник')
           else
             Writeln('Остроугольный треугольник')
    end;
  Readln
end.
```

3

```
var F: Text;
    L: Byte;
    S: string;
begin
  Assign(F, 'test.txt');
  Reset(F);
  L:=0;
  while not Eof(F) do
    begin
      Readln(F, S);
      if Length(S)>L then
        L:=Length(S)
    end;
  Close(F);
  Reset(F);
  while not Eof(F) do
    begin
      Readln(F, S);
      if Length(S)=L then
        Writeln(S)
    end;
  Close(F);
  Readln
end.
```

4

```
var F: Text;
    I, J: Byte;
    S: string;
    A: array [0..255] of Integer;
begin
  Assign(F, 'test.txt');
  for I:=0 to 255 do
    A[I]:=0;
  Reset(F);
  while not Eof(F) do
    begin
      Readln(F, S);
      Inc(A[Length(S)])
    end;
  Close(F);
  for I:=1 to 255 do
    if A[I]>0 then
      begin
        J:=1;
        Reset(F);
```

```
        while (not Eof(F)) and (J<=I) do
          begin
            Readln(F, S);
            if Length(S)=I then
              begin
                Writeln(S);
                Inc(J)
              end
          end;
        Close(F);
      end;
  Readln
end.
```

5 **Способ 1.**

Сложность этой задачи – в том, что числа, которые встречаются в массиве несколько раз, выписать надо только один раз. Но как «пометить» уже выписанные числа? Если бы массив был записан на листочке бумаги, мы могли бы их подчеркнуть; здесь же приходится заводить дополнительный массив. Его компоненты будут равны FALSE, если мы не знаем, встречается ли это число в массиве еще раз, и TRUE, если такое число уже было обнаружено.

```
const N=10;
var I, J, Num, Count: Byte;
    A: array [1..N] of Integer;
    B: array [1..N] of Boolean;
begin
  for I:=1 to N do
    begin
      B[I]:=False;
      Write('Введите ', I, '-й элемент: ');
      Readln(A[I])
    end;
  Count:=0;
  for I:=1 to N do
    if not B[I] then
      begin
        B[I]:=True;
        Num:=1;
        Inc(Count);
        Write(A[I], ' встречается ');
        for J:=I+1 to N do
          if (A[J]=A[I]) and (not B[J]) then
            begin
              Inc(Num);
              B[J]:=True
```

```
            end;
          Writeln(Num, ' раз');
        end;
    Writeln('Всего ', Count, ' разных чисел');
    Readln
end.
```

Способ 2.

А вот другой способ решения этой задачи: без дополнительного массива, зато с многочисленными проверками. Каждый раз перед тем, как написать число, будем смотреть, не встречалось ли оно, проверяя весь массив до него (в задаче это реализовано проверкой «назад» – от числа к первому элементу). Заметим, что для первого числа программа также будет работать правильно, так как цикл **For** от *I*-1 до 1 при *I*=0 ошибкой не является, а просто ни разу не выполнится.

Приведем здесь только часть алгоритма, отличающуюся от первого варианта:

```
Count:=0;
for I:=1 to N do
  begin
    Num:=1;
    T:=True;
    for J:=I-1 downto 1 do
      if A[I]=A[J] then
        T:=False;
    if T then
      begin
        Inc(Count);
        Write(A[I], ' встречается ');
        for J:=I+1 to N do
          if A[J]=A[I] then
            Inc(Num);
          Writeln(Num, ' раз');
      end
  end;
Writeln('Всего ', Count, ' разных чисел');
```

6 **Способ 1.**

```
const N=1;
      M=100;
var A, B, X: Integer;
    C: Char;
begin
  Writeln('Загадайте число от ', N, ' до ', M,
          'и нажмите Enter');
```

```
  Readln;
  A:=N;
  B:=M;
  repeat
    X:=(B+A) div 2;
    Writeln('Ваше число больше ', X, '? (Y/N)');
    Readln(C);
    if (C='Y') or (C='y') then
      A:=X+1
    else
      B:=X
  until A=B;
  Writeln('Вы загадали число ', A);
  Readln
end.
```

Способ 2.

```
var I,Z,N,K : Integer;
    Otv   : Char;
begin
  Write('Введите начало диапазона '); Readln(N);
  Write('Введите конец  диапазона '); Readln(K);
  I:=0;
  repeat
    Z:=(N+K)div 2;
    repeat {Защита от несанкционированного ввода,}
           {чтобы не ввели недопустимый ответ}
    Write('Введите знак: задуманное число ', Z,' ');
      Readln(Otv);
    until (Otv='=')Or(Otv='>')Or(Otv='<');
    I:=I+1;
    if Otv='<' then
      K:=Z;
    if Otv='>' then
      if N=Z then
        N:=N+1
      else N:=Z;
    if Otv='=' then
      K:=N {Для единообразия}
  until K=N;
  Writeln('Ответ ',K);
  Writeln('Число шагов - ', I)
end.
```

7 Эту задачу можно решить разными способами. Вот один из них, в котором вся хитрость – в задании исходных значений массивов строк.

```
const C1: array [2..9] of string [15] =
  ('двадцать ', 'тридцать ', 'сорок ', 'пятьдесят ',
   'шестьдесят ', 'семьдесят ', 'восемьдесят ',
  'девяносто ');
   C2: array [0..9] of string [15] =
 ('', 'один', 'два', 'три', 'четыре', 'пять',
  'шесть', 'семь', 'восемь', 'девять');
var I: Byte;
begin
  Write('Введите число от 20 до 99: ');
  Readln(I);
  Write(C1[I div 10], C2[I mod 10]);
  Readln
end.
```

8

```
var I, J: Byte;
function F(A, B: Byte): Boolean;
var X, Y: Boolean;
begin
  if A=1 then X:=True
    else
      X:=False;
  if B=1 then Y:=True
    else
      Y:=False;
  F:=(not X) or Y
end;
begin
  Writeln('A  B  F(A,B)');
  for I:=0 to 1 do
    for J:=0 to 1 do
      Writeln(I, '  ', J, '  ', Ord(F(I, J)));
  Readln
end.
```

Если же представлять константы «истина» и «ложь» не в виде 1 и 0, а в виде TRUE и FALSE, как принято в Паскале, то программа будет совсем простой, так как логические константы в Паскале можно печатать. Они также могут быть аргументами оператора **For**.

```
Function F(A,B: Boolean):Boolean;
begin
  F:=not A or B
end;
```

```
var I, J: Boolean;
begin
  Writeln('       A        B              F(A,B)');
  for I:=True downto False do
    for J:=True downto False do
      Writeln(I:9, J:9, F(I,J):15);
  Readln
end.
```

9 Заметим, что в условии задачи нет уточнения, чему может равняться *N*, поэтому ограничимся рассмотрением чисел, квадрат которых «помещается» в тип данных **Longint** Паскаля, т. е. не более чем четырехзначных чисел (некоторые пятизначные числа тоже можно было бы рассмотреть, но не все, поэтому ограничимся четырехзначными). Будем переводить сравниваемые числа в строки и использовать процедуры и функции для работы со строками.

```
const L=9999;
var N: Longint;
    I: Integer;
    S, S1: string;
begin
  for I:=1 to L do
    begin
      N:=I*I;
      Str(N, S);
      Str(I, S1);
      S:=Copy(S, Length(S)-Length(S1)+1, 255);
      if S=S1 then
        Writeln(I, '*', I, '=', N)
    end;
  Readln
end.
```

10

```
var I: Integer;
    A, B, C, D: Byte;
begin
  for I:=1000 to 9999 do
    begin
      A:=I mod 10;
      B:=(I div 10) mod 10;
      C:=(I div 100) mod 10;
      D:=(I div 1000) mod 10;
      if (A<>B) and (A<>C) and (A<>D)
      and (B<>C) and (B<>D) and (C<>D) then
        Write (I, ' ');
```

```
    end;
  Readln
end.
```

11 Существует несколько способов решения этой задачи. Первый состоит в том, чтобы запускать внутренний цикл «с начала в конец» или «с конца в начало» в зависимости от четности переменной внешнего цикла. Второй – в том, чтобы во внутреннем цикле выражать записываемое число через индекс переменной цикла. Ниже мы приводим оба способа, а вы можете сравнить их по эффективности. По мнению авторов книги, первый способ решения относится к категории решений «в лоб», а второй – к более «вдумчивым». На ЕГЭ, когда условие задачи начинается со слов «напишите максимально эффективную программу», за решение первым способом вы можете получить на один балл меньше, чем за решение вторым способом.

Способ 1.

```
const M=4;
      N=6;
var A: array [1..M, 1..N] of Integer;
    I, J: Byte;
    S: Integer;
begin
  S:=1;
  for I:=1 to M do
    if (I mod 2)=1 then
      for J:=1 to N do
        begin
          A[I, J]:=S;
          Inc(S)
        end
    else
      for J:=N downto 1 do
        begin
          A[I, J]:=S;
          Inc(S)
        end;
  for I:=1 to M do
    begin
      for J:=1 to N do
        Write(A[I, J]:4);
      Writeln
    end;
  Readln
end.
```

Способ 2.

```
const M=4;
      N=6;
var A: array [1..M, 1..N] of Integer;
    I, J: Byte;
begin
  for I:=1 to M do
    for J:=1 to N do
      if (I mod 2)=1 then
        A[I, J]:=(I-1)*N+J
      else
        A[I, J]:=I*N-J+1;
  for I:=1 to M do
    begin
      for J:=1 to N do
        Write(A[I, J]:4);
      Writeln
    end;
  Readln
end.
```

12 **Вариант 1** (данные находятся в массиве).

Так же, как и в предыдущей задаче, приведем два решения, но на этот раз алгоритмически более сложный способ проигрывает прямому и более простому. Второй способ намного легче читаем и более понятен.

Способ 1. Важной особенностью решения является то, что выход из цикла осуществляется, когда индекс массива становится больше количества элементов в массиве. Поэтому для работы с массивом из N элементов мы вводим массив размером $N+1$, помещая в $A[N+1]$ произвольное число (например, 0). Тогда обращение к элементу $A[N+1]$ вполне правомерно.

```
const N=10;
var A: array [1..N+1] of Integer;
    I, J, Max: Integer;
begin
  A[N+1]:=0;
  for I:=1 to N do
    begin
      Write('Введите ', I, '-й элемент массива: ');
      Readln(A[I])
    end;
  I:=0;
  Max:=1;
  J:=0;
  repeat
```

```
    I:=I+J;
    J:=0;
    repeat
      Inc(J);
    until (I+J>N) or (A[I]<>A[I+J]);
    if J>Max then
      Max :=J
  until I>=N;
  Writeln('Наибольшее количество чисел подряд - ',
           Max);
  Readln
end.
```

Способ 2.

```
const N=10;
var A: array [1..N] of Integer;
    I, Max, Count, Pred: Integer;
begin
  for I:=1 to N do
    begin
      Write('Введите ', I, '-й элемент массива: ');
      Readln(A[I])
    end;
  Pred:=1;
  Max:=1;
  Count:=1;
  for I:=2 to N do
    if A[I]=A[Pred] then
      Inc(Count)
    else
      begin
        Pred:=I;
        if Count>Max then
          Max:=Count;
        Count:=1
      end;
  if Count>Max then
    Max:=Count;
  Writeln('Наибольшее количество чисел подряд - ',
           Max);
  Readln
end.
```

Вариант 2 (данные находятся в файле).

Приведем лишь решение, аналогичное способу 2 решения предыдущего варианта.

```
var F: file of Integer;
    Pred, Max, I, Count: Integer;
begin
  Assign(F, 'nums.int');
  Reset(F);
  Max:=1;
  Count:=1;
  if not Eof(F) then
    Read(F, Pred);
  while not Eof(F) do
    begin
      Read(F, I);
      if I=Pred then
        Inc(Count)
      else
        begin
          Pred:=I;
          if Count>Max then
            Max:=Count;
          Count:=1
        end
    end;
  if Count>Max then
    Max:=Count;
  Writeln('Наибольшее количество чисел подряд - ',
           Max);
  Readln
end.
```

Вариант 3 (данные находятся в строке).

Поскольку при вводе строки числа, написанные в ней, будут представлены в виде символов, придется каждое число «выуживать» из строки, преобразовывая его из текстового написания в числовое. Для этого преобразования мы написали процедуру **GetNextNum**, поскольку однотипные действия повторяются в программе несколько раз. Ее параметр передается по ссылке, так как в процессе работы процедура изменяет его; помимо этого она «укорачивает» саму строку *S*.

Заметим также, что эта программа будет корректно работать лишь в том случае, если пользователь вводит целые числа, разделенные ровно одним пробелом. Обработку некорректного ввода информации пользователем мы оставим за рамками этого решения.

```
var S, S1: string;
    I, Max, Count, Pred, Err: Integer;
procedure GetNextNum(var N: Integer);
  var P: Integer;
```

```
  begin
    P:=Pos(' ', S);
    if P=0 then
      P:=255;
    S1:=Copy(S, 1, P-1);
    Val(S1, N, Err);
    S:=Copy(S, P+1, 255);
  end;
begin
 Write('Введите целые числа, разделенные пробелами:');
  Readln(S);
  GetNextNum(Pred);
  Max:=1;
  Count:=1;
  while S<>'' do
    begin
      GetNextNum(I);
      if I=Pred then
        Inc(Count)
      else
        begin
          Pred:=I;
          if Count>Max then
            Max:=Count;
          Count:=1
        end;
    end;
  if Count>Max then
    Max:=Count;
  Writeln('Наибольшее количество чисел подряд - ',
           Max);
  Readln
end.
```

13 Площадь многоугольника равна сумме площадей треугольников, из которых он состоит. Вычисление площади треугольника, когда известны длины трех сторон, проведем по формуле Герона и оформим в виде функции.

```
var X1, X2, X3, X4, D, S: Real;
function Geron (a,b,c: Real): Real;
var p: Real;
begin
  P:=(a+b+c)/2;
  Geron:=Sqrt(p*(p-a)*(p-b)*(p-c))
end;
```

```
begin
  Write('Введите длины 4 сторон прямоугольника ');
  Readln(X1, X2, X3, X4);
  Write('Введите длину диагонали  ');
  Readln(D);
  Writeln('Площадь = ',
          Geron(X1,X2,D)+Geron(X3,X4,D) :6:2)
end.
```

14 Воспользуемся алгоритмом Евклида для нахождения наибольшего общего делителя (НОД) двух чисел. Оформим его в виде функции от двух параметров. Суть алгоритма: пока числа не равны, надо вычитать из большего числа меньшее, а когда они сравняются, это и будет их НОД.

```
var P, R, S, T  : Integer;
function NOD(A, B: Integer): Integer;
var C: Integer;
begin
  while A<>B do
    begin
      if B>A then
        begin
          C:=B;
          B:=A;
          A:=C
        end;
      A:=A-B
    end;
  Nod:=A
end;
begin
  Write('Введите 4 числа ');
  Readln(P,R,S,T);
  Writeln('НОД(',P,',',R,',',S,',',T,')=',
          NOD(Nod(P,R),Nod(S,T)))
end.
```

15 Для определения, является ли число простым, напишем логическую функцию. Отдельно рассмотрим числа 2 и 3 – они простые; в остальных случаях заданное число будем последовательно делить на 2, 3 и т.д., пока делитель не превысит половину заданного числа (на самом деле можно закончить вычисления и раньше – когда делитель превысит квадратный корень из заданного числа). Если число ни на что не разделилось – оно простое, если же делитель нашелся, то заканчиваем цикл с ответом FALSE.
В основной программе вызовем эту функцию в условном операторе внутри цикла.

Несмотря на то, что в тексте этой программы обращение к функции написано один раз, оформление решения задачи «является ли заданное число простым» вполне оправдано. Написанная таким образом программа легче читается и отлаживается. Поставленная задача разделена на две части: определение «простоты» одного числа и применение этой функции к последовательности чисел.

```
var K, N : Integer;
function Prost(N: Integer): Boolean;
  var T: Boolean;
      D: Integer;
begin
  if N<4 then
    Prost:=True
  else
    begin
      T:=True;
      D:=2;
      while (D <= N div 2) and T do
        begin
          T := (N mod D)<>0;
          D:=D+1
        end;
      Prost:=T
    end
end;
begin
  Write('Число  - ');
  Readln(K);
  for N:=2 to K do
    begin
      Write(N, '  ');
      if not (Prost(N)) then
        Write('не ');
      Writeln('является простым')
    end.
```

16 Воспользуемся описанной в предыдущей программе функцией. Будем проверять с ее помощью на «простоту» соседние нечетные числа. Обратите внимание, что наша функция только вычисляет свое значение, но ничего не выводит на печать. Если бы мы вставили печать в тело функции, то предыдущая программа была бы даже несколько проще, но для этой задачи такая функция уже бы не подошла.

```
var K, N: Integer;
    S1, S2: Boolean;
```

```
function Prost(N: Integer): Boolean;
  var T: Boolean;
      D: Integer;
begin
  if N<4 then
    Prost:=True
  else
    begin
      T:=True;
      D:=2;
      while (D <= N div 2) and T do
        begin
          T := (N mod D)<>0;
          D:=D+1
        end;
      Prost:=T
    end
end;
begin
  Write('Число  - ');
  Readln(K);
  N:=3;
  S1:=Prost(N);
  while N<K do
    begin
      N:=N+2;
      S2:=Prost(N);
      if S1 and S2 then
        Writeln(N-2, ' ', N);
      S1:=S2
    end
end.
```

17 Воспользуемся той же функцией **Prost**, которая используется в предыдущих двух задачах. Тогда основная программа будет выглядеть следующим образом. В цикле **While** будем проверять все четные *N*, меньшие *K*. Выпишем число. Затем попробуем подобрать пару составляющих его слагаемых – простых чисел. Будем проверять все пары, начиная с первого простого числа 2. Если ни одной такой пары найдено не будет, то на экране останется выписанным число, не удовлетворяющее гипотезе. Если же пара будет найдена, выпишем ее и выйдем из цикла. Заметим, что использованный для этой цели оператор **Break** осуществляет выход только из одного цикла (поиск слагаемых данного числа); цикл по исследованию всех четных чисел продолжается. Если оператор **Break** не добавлять, то будут выписаны *все* пары простых слагаемых для каждого числа.

```
begin
  Write('Число  - ');
  Readln(K);
  N:=4;
  While N<=K do
    begin
      Write(N);
      for I:=2 to N-2 do
        if Prost(I) and Prost(N-I) then
          begin
            Writeln('=',I,'+',N-I);
            Break
          end;
      N:=N+2
    end
end.
```

18 Оформим задачу «есть ли в заданном слове удвоенные буквы» в виде отдельной логической функции. В теле этой функции будем проверять все буквы слова: не равна ли какая-то из них соседней. В основной программе надо аккуратно выполнить все действия для чтения слов (строк) из файла и к каждому применить написанную функцию.

```
type Slovo=String[20];
var Ft: Text;
    S : Slovo;
function Udvoen(S: Slovo): Boolean;
var I: Integer;
begin
  Udvoen:=False;
  for I:=1 to Length(S)-1 do
    if S[I]=S[I+1] then
      begin
        Udvoen:=True;
        Break
      end;
end;
begin
  Assign(Ft, 'Slova.txt');
  Reset(Ft);
  while (not Eof(Ft)) do
    begin
      Readln(Ft,S);
      if Udvoen(S) then
        Writeln(S)
    end;
end.
```

19 Печать одного прямоугольника оформим в виде процедуры. В основной программе несколько раз обратимся к ней с соответствующими параметрами.

```
procedure Ris(M,N: Integer; C: Char);
var I,J: Integer;
begin
  for I:=1 to M do
    begin
      for J:=1 to N do
        Write(C);
      Writeln
    end
end;
begin
  Ris(2,10,'*');
  Ris(4,15,'@');
  Ris(6,20,'#')
end.
```

20 Если бы числа хранились в массиве, то можно было бы предложить очень простое (хотя и неэффективное) решение: за первый проход массива найти максимум, а при втором проходе подсчитать, сколько раз он входит в массив. Однако последовательность два раза «пробежать» нельзя (числа в памяти не хранятся), поэтому рассмотрим более сложный, но эффективный метод (для массива он тоже годится).

Каждое введенное число (не 0) будем проверять, не больше ли оно текущего максимума. Если это так, то не только поменяем максимум, но и начнем заново подсчет количества ($KM = 1$). В случае же обнаружения числа, равного максимуму, KM увеличивается на 1.

```
var I, KM: Integer;
    Max, P: Real;
begin
  I:=1;
  Write(I, '-й элемент  ');
  Readln(Max);
  KM:=1;
  repeat
    I:=I+1;
    Write(I, '-й элемент  ');
    Readln(P);
    if P<>0 then
      if P>Max then
        begin
          P:=Max;
```

```
            KM:=1
          end
        else if P=Max then
               KM:=KM+1
  until P=0;
  Writeln('Максимум ',Max:8:1, ' встречается '
           KM:4, ' раз' )
end.
```

21 Некоторая сложность этой задачи состоит в том, что во время просмотра массива при изменении минимума надо также менять «предминимум» (у нас он обозначен *MIN1*), а если минимум не меняется, то надо проверить, не меняется ли «предминимум».

```
const N=10;
var I: Integer;
    Min, Min1: Real;
    Mas: array [1..N] of Real;
begin
  for I:=1 to N do
    begin
      Write(I, '-й элемент  ');
      Readln(Mas[i])
    end;
  Min:=Mas[1]; Min1:=Mas[2];
  for I:=3 to N do
    begin
      if Mas[i]<Min then
        begin
          Min1:=Min;
          Min:=Mas[i]
        end
      else if Mas[i]<Min1 then
             Min1:=Mas[i]
    end;
  Writeln(Min:8:1, Min1:10:1)
end.
```

Раздел «Сложные задачи»

1

```
const N=10;
var A: array [1..N] of Integer;
    S: Integer;
    I, CM: Byte;
begin
  S:=0;
```

```
  CM:=0;
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I]);
      if A[I]>0 then
        S:=S + A[I]
      else if A[I]<0 then
        Inc(CM)
    end;
  Writeln('Сумма положительных элементов = ', S);
  Writeln('Число отрицательных элементов = ', CM);
  Readln
end.
```

2

```
const N=10;
var A: array [1..N] of Integer;
    I, U: Shortint;
    Sort: Boolean;
begin
  U:=0;
  Sort:=True;
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I]);
      if I>1 then
        begin
          if ((A[I]>A[I-1]) and (U<0))
          or ((A[I]<A[I-1]) and (U>0)) then
            Sort:=False;
          if A[I]>A[I-1] then
            U:=1;
          if A[I]<A[I-1] then
            U:=-1
        end
    end;
  if Sort then
    Writeln('Массив упорядочен')
  else
    Writeln('Массив НЕ упорядочен');
  Readln
end.
```

3

```
const N=5;
      M=5;
```

```
var A: array [1..N] of Integer;
    B: array [1..M] of Integer;
    C: array [1..M+N] of Integer;
    I, J, K: Byte;
begin
  Writeln('Массив A[I] упорядочен по возрастанию.');
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  Writeln('Массив B[J] упорядочен по возрастанию.');
  for J:=1 to M do
    begin
      Write ('Введите B[', J, ']: ');
      Readln(B[J])
    end;
  I:=0;
  J:=0;
  for K:=1 to M+N do
    begin
      if I=N then
        begin
          Inc(J);
          C[K]:=B[J]
        end
      else if J=M then
        begin
          Inc(I);
          C[K]:=A[I]
        end
           else if A[I+1]<B[J+1] then
             begin
               Inc(I);
               C[K]:=A[I]
             end
                else
                  begin
                    Inc(J);
                    C[K]:=B[J]
                  end
    end;
  for K:=1 to M+N do
    Writeln('C[', K, ']=', C[K]);
  Readln
end.
```

4

```
const N=10;
var A: array [1..N] of Integer;
    Temp: Integer;
    I, J, NMin: Byte;
begin
  for I:=1 to N do
    begin
      Write ('Введите A[', I, ']: ');
      Readln(A[I])
    end;
  for I:=1 to N-1 do
    begin
      NMin:=I;
      for J:=I+1 to N do
        if A[J]<A[NMin] then
          NMin:=J;
      Temp:=A[NMin];
      A[NMin]:=A[I];
      A[I]:=Temp
    end;
  for I:=1 to N do
    Writeln('A[', I, ']=', A[I]);
  Readln
end.
```

5

```
var S, S1: string;
    I, J, L, L1: Byte;
    Eq: Boolean;
begin
  Write ('Введите строку: ');
  Readln(S);
  Write ('Введите искомую подстроку: ');
  Readln(S1);
  L:=Length(S);
  L1:=Length(S1);
  for I:=1 to L-L1+1 do
    begin
      Eq:=True;
      for J:=1 to L1 do
        Eq:=Eq and (S[I+J-1]=S1[J]);
      if Eq then
        Break
    end;
  if Eq then
    Writeln('Подстрока "', S1, '" входит в строку
```

```
            "', S, '" начиная с ', I, '-го символа.')
  else
    Writeln('Подстрока "', S1, '" НЕ входит в строку
            "', S, '".');
  Readln
end.
```

6

```
const Eps=0.00001;
var A, B, C: Real;
  function F(X: Real): Real;
    begin
      F:=Cos(X/2)
    end;
begin
  Writeln('Введите A: ');
  Readln(A);
  Writeln('Введите B: ');
  Readln(B);
  repeat
    C:=(A+B)/2;
    if F(A)*F(C)<0 then
      B:=C
    else
      A:=C
  until B-A<Eps;
  Writeln('Корень функции равен: ', (A+B)/2:10:5);
  Readln
end.
```

7

Алгоритм был подробно описан ранее. Базой рекурсии здесь служит то же условие, которое является условием прекращения цикла в нерекурсивном решении: $B - A < Eps$.

```
const Eps=0.00001;
var A, B, C: Real;
  function F(X: Real): Real;
    begin
      F:=Cos(X/2)
    end;
  function Kor(A, B: Real):Real;
    var C: Real;
    begin
      C:=(A+B)/2;
      if B-A<Eps then
        Kor:=C
      else
        if F(A)*F(C)<0 then
```

```
            Kor:=Kor(A,C)
          else Kor:=Kor(C,B)
      end;
begin
  Write ('A= ');
  Readln(A);
  Write ('B= ');
  Readln(B);
  Writeln('Корень функции равен ', Kor(A, B):10:5);
  Readln
end.
```

8

```
var N, M, K, I: Integer;
begin
  Writeln('Введите целое число: ');
  Readln(N);
  I:=2;
  M:=0;
  K:=Round(Sqrt(N))+1;
  while I<=K do
    begin
      if (N mod I = 0) and (M=0) then
        M:=I;
      Inc(I)
    end;
  if M=0 then
    M:=N;
  Writeln('Наименьший делитель числа ', N,
          ' равен: ', M);
  Readln
end.
```

9

```
var N, M, K, I: Integer;
function ND (N: Integer): Integer;
  begin
    I:=2;
    M:=0;
    K:=Round(Sqrt(N))+1;
    while I<=K do
      begin
        if (N mod I = 0) and (M=0) then
          M:=I;
        Inc(I)
      end;
    if M=0 then
```

```
      M:=N;
    ND:=M
  end;

begin
  Writeln('Введите целое число: ');
  Readln(N);
  Write ('Разложение на множители числа ',
         N, ' = ');
  repeat
    M:=ND(N);
    Write(M);
    N:=N div M;
    if N>1 then
      Write ('*');
  until N=1;
  Readln
end.
```

10

```
const N=5;
      M=5;
var A: array [0..N] of Integer;
    B: array [0..M] of Integer;
    C: array [0..M+N] of Longint;
    I, J: Byte;
begin
  for I:=N downto 0 do
    begin
      Write ('Введите коэффициент A при X^', I,
             ': ');
      Readln(A[I])
    end;
  for J:=M downto 0 do
    begin
      Write ('Введите коэффициент B при X^', J,
             ': ');
      Readln(B[J])
    end;
  for I:=0 to M+N do
    C[I]:=0;
  for I:=0 to N do
    for J:=0 to M do
      C[I+J]:=C[I+J] + A[I]*B[J];
  Write ('A(x)*B(x)=');
  for I:=M+N downto 0 do
    if C[I]>0 then
```

```
      Write ('+', C[I], 'X^', I)
    else if C[I]<0 then
          Write (C[I], 'X^', I);
  Readln
end.
```

11

```
const N=5;
type Massiv= array [1..N] of Integer;
var M: Massiv;
    j, K, X, Mesto: Integer;
    T: Boolean;
procedure VvodMas (var M: Massiv; N: Integer);
var J: Integer;
begin
  Writeln('Ввод массива');
  for J:=1 to N do
    begin
      Write ('Введите ', j, '-й элемент: ');
      Readln(M[j])
    end
end;
Procedure PrintMas (var M: Massiv; N: Integer);
var J: Integer;
begin
  for J:=1 to N do
     Write (M[J],'  ');
  Writeln
end;
begin
  VvodMas (M, N);
  {Сортировка пузырьком}
  K:=N;
  repeat
    T:=True; {перестановок не было}
    K:=K-1;
    for J:=1 to K do
      If M[J]>M[J+1] then
        begin
          X:=M[J];
          M[J]:=M[J+1];
          M[J+1]:=X;
          T:=False {перестановка произошла}
        end;
  until T;
  PrintMas (M, N);  { Печать массива }
end.
```

12 Алгоритм сортировки простыми вставками основан на том, что, если начало массива уже отсортировано, очередной элемент вставляется в это начало на место, для чего часть последовательности сдвигается вправо.

Так как для работы подобного алгоритма надо иметь отсортированную последовательность, проверим, в правильном ли порядке стоят два первых элемента, и, если надо, поменяем их местами. Теперь мы имеем отсортированную последовательность из двух элементов и можем в нее вставлять третий элемент. На последующих шагах отсортированная часть будет расти: в последовательность из трех элементов будем вставлять четвертый..., из I–1 элементов – N-й.

Может оказаться, что элемент стоит правильно, если «кандидат на вставку» – $M[i]$ – больше предыдущего элемента, тогда делать ничего не надо. Иначе вставлять придется, и надо найти элементу место. Для этого будем рассматривать все предшествующие элементы. Закончить просмотр надо, если мы найдем меньший элемент или если массив закончится. Обратите внимание: когда условное выражение состоит из нескольких отношений, связанных логической операцией (в данном случае **and**), отношения надо брать в скобки!

Как только место для вставки найдено, освобождаем его. Для этого все элементы, начиная с I–1 до стоящего после найденного, сдвигаем на одну позицию вправо.

Здесь приведена не полная программа, а фрагмент для решения данной задачи. Чтобы превратить его в работоспособную программу, его надо дополнить описаниями переменных, вводом и выводом массива.

В программе есть одна строка, заключенная в фигурные скобки – так в Паскале записываются комментарии. Эта строка не нужна для работы алгоритма (она, будучи заключенной в фигурные скобки, и не работает). Однако если вы уберете эти скобки, то на каждом шаге будет печататься массив, и вы сможете пошагово наблюдать, как происходит сортировка вставками.

```
If M[1]>M[2] Then Begin K:=M[2];
                        M[2]:=M[1];
                        M[1]:=K
                  End;
  For I:=3 to N do
  Begin
   { for ii:=1 to N do Write(M[ii],' ');
          Writeln('#', i);   }
     If M[i]<M[i-1]
          Then Begin J:=i-2 ;
                 While (M[i]<M[j]) and (J>=1) do
                        J:=J-1;
                 K:=M[I];
```

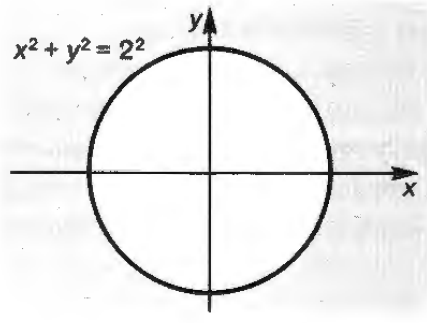
```
                    For L:=I downto J+2 do
                            M[L]:=M[L-1];
                    M[J+1]:=K
                  end;
end;
```

Раздел «Типовые задачи части «С» ЕГЭ»

1 Первая неточность, которую допустил программист, – не совсем корректное форматирование текста программы. По правилам синтаксиса **Else** относится к идущей непосредственно перед ним конструкции **If – Then**. Отформатируем программу так, чтобы она стала более наглядной.

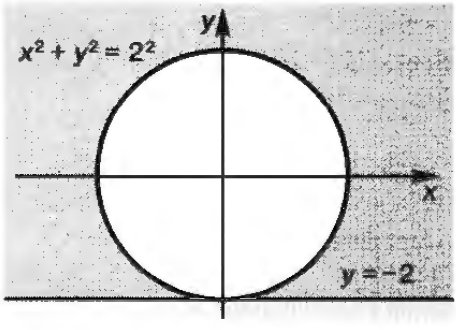
```
var x, y:real;
begin
    readln (x, y);
    if x*x +y*y >= 4 then
        if y >= -2 then
            if y <= x then
                write ('Принадлежит')
            else
                write ('Не принадлежит')
end.
```

Первый оператор **If** определяет следующую область:



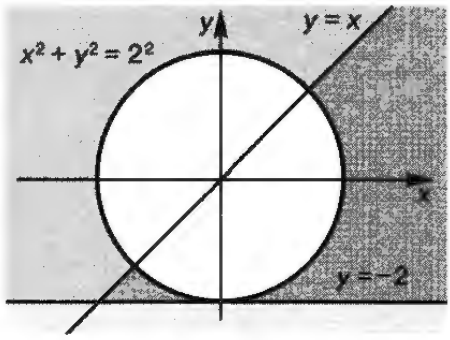

```
var x, y:real;
begin
    readln (x, y);
    if x*x +y*y >= 4 then
    …
end.
```

Следующий оператор **If** из этой области определяет подобласть:




```
var x, y:real;
begin
    readln (x, y);
    if x*x +y*y >= 4 then
    if y >= -2 then
      …
end.
```

Наконец, последний оператор **If** выделяет подобласть:



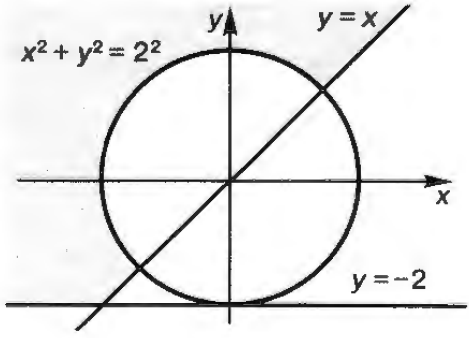

```
var x, y:real;
begin
     readln (x, y);
     if x*x +y*y >= 4 then
      if y >= -2 then
        if y <= x then
        write ('Принадлежит')
      …
end.
```

Сравнив образец в условии задачи и то, что мы получили сейчас, обнаруживаем ошибку – это лишняя область $x > 0$. Следовательно, необходимо добавить еще одно условие: $x <= 0$.

```
var x, y:real;
begin
 readln (x, y);
 if x*x +y*y >= 4 then
      if y >= -2 then
           if y <= x then
                if x <= 0 then
                     write ('Принадлежит')
      …
end.
```

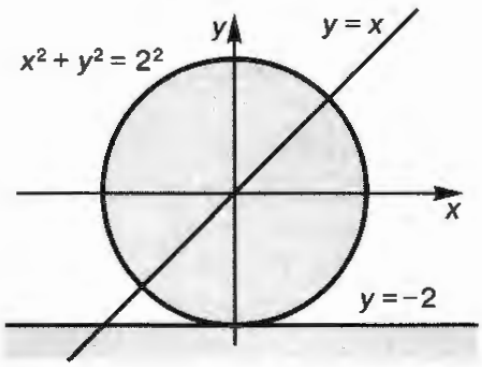
Программа выведет сообщение «Не принадлежит», если не будет выполнено только последнее условие. При невыполнении всех предыдущих условий вообще никакого сообщения выведено не будет, чего быть не должно. Это означает, что **Else** следует добавить для каждого **Then**.

В программе же **Else** указано только для последнего **Then**, т.е. «Не принадлежит» будет напечатано только для области:

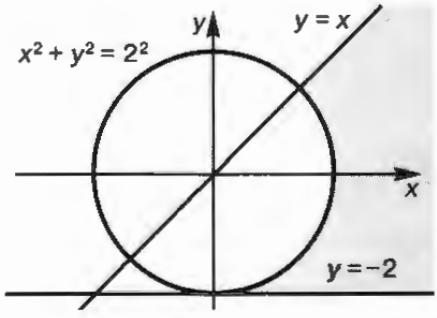


Ответом является:

1. Любая пара (x, y), для которой выполняется условие $x^2+y^2<4$ или $y<-2$, так как для этой области не будет напечатано ничего.



2. Любая пара (x, y), для которой выполняется условие $y>=-2$ и $y<=x$ и $x>0$ и $x^2+y^2>4$, так как для этой области будет напечатано «Принадлежит», хотя в условии задачи такая область не заштрихована.



Исправленные варианты программы.

Вариант 1.

```
var x, y:real;
begin
 readln (x, y);
 if x*x +y*y >= 4 then
       if y >= -2 then
            if y <= x then
                 if x <= 0 then
                      write ('Принадлежит')
                 else
                      write ('Не принадлежит')
            else
                 write ('Не принадлежит')
       else
            write ('Не принадлежит')
 else
       write ('Не принадлежит')
end.
```

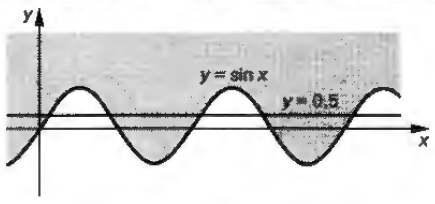
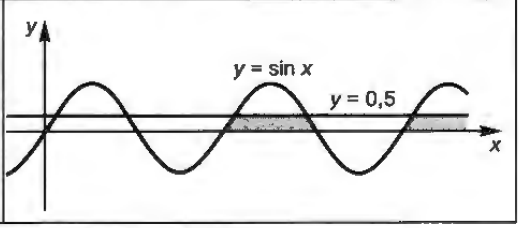
Вариант 2.

Можно то же самое написать короче, используя оператор логического **И** (and):

```
var x, y:real;
begin
     readln (x, y);
     if (x*x +y*y >=4) and
          (y>= -2) and (y<=x) and (x<=0) then
               write ('Принадлежит')
     else
          write ('Не принадлежит')
end.
```

В качестве ответа требуется указать любую точку из интервалов, полученных выше. Например, **точка с координатами (0,1).**

2 Ответом является:

1) любая пара (x, y), для которой выполняется $y>\sin(x)$ или $y<\sin(x)$ и $y>0.5$	2) любая пара (x, y), для которой выполняется $y<\sin(x)$ и $y<0.5$ и $y>0$ и ($x>5$ или $x<0$)
	

Исправленные варианты программы.

Вариант 1.

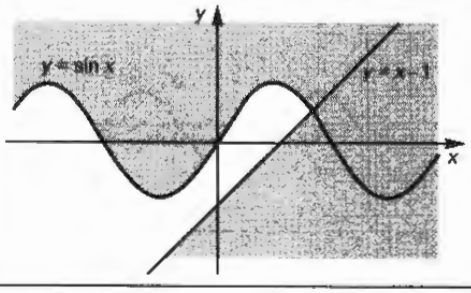
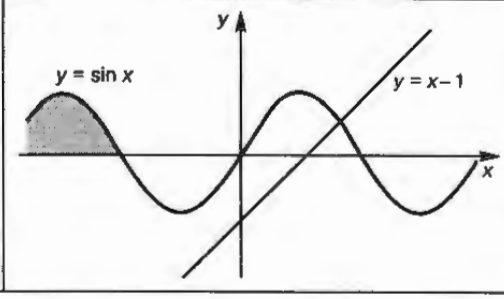
```
var x, y: real;
begin
     readln (x,y);
     if y<=sin(x) then
          if y<=0.5 then
               if y>=0 then
                    if x<=5 then
                         if x>=0 then
                              write ('принадлежит')
                         else
                              write ('не принадлежит')
                    else
                         write ('не принадлежит')
               else
                    write ('не принадлежит')
          else
               write ('не принадлежит')
     else
          write ('не принадлежит')
end.
```

Вариант 2.

Можно то же самое написать короче, использовав оператор логического **И** (and):

```
var x, y: real;
begin
     readln (x,y);
     if y<=sin(x)and(y<=0.5)and(y>=0)and(x<=5)
and(x>=0)then
          write ('принадлежит')
     else
          write ('не принадлежит')
end.
```

3 Ответом является:

1) любая пара (x, y), для которой выполняется $y>\sin x$ или $y<x-1$	2) любая пара (x, y), для которой выполняется $y<=\sin x$ и $x<-2\pi$
	

Исправленные варианты программы.

Вариант 1.

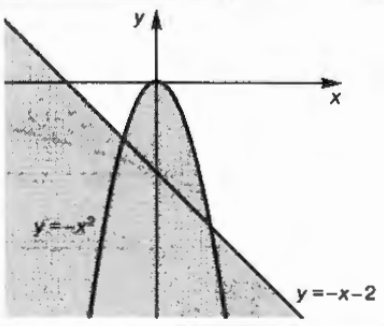
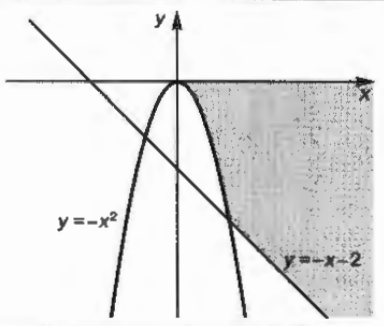
```
var x, y: real;
begin
     readln (x, y);
     if y<=sin(x) then
          if y>=x-1 then
               if y>=0 then
                    if x>=0 then
                         write ('принадлежит')
                    else
                         write ('не принадлежит')
               else
                    write ('не принадлежит')
          else
               write ('не принадлежит')
     else
          write ('не принадлежит')
end.
```

Вариант 2.

Можно то же самое написать короче, использовав оператор логического **И** (and).

```
var x, y: real;
begin
    readln (x, y);
    if (y<=sin(x)) and (y>=x-1) and (y>=0) and
(x>=0) then
         write  ('принадлежит')
    else write ('не принадлежит')
end.
```

4 Ответом является

1) любая пара (x, y), для которой выполняется $y<-x^2$ или $y<-x-2$	2) любая пара (x, y), для которой выполняется $y<-2$ и $y>=-x-2$ и $x>0$ и $y>x^2$
	

Исправленные варианты программы.

Вариант 1.

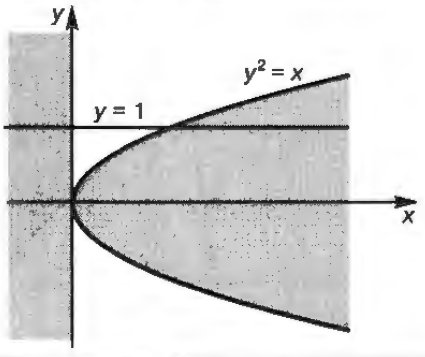
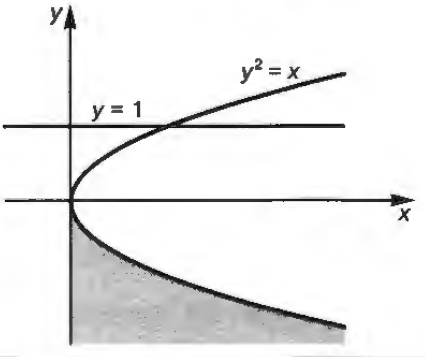
```
var x,y: real;
begin
     readln (x, y);
     if y>=-x*x then
          if y>=-x-2 then
               if y<=0 then
                    if x <= 0 then
                         write ('принадлежит')
                    else
                         write ('не принадлежит')
                         else
                    write ('не принадлежит')
          else
               write ('не принадлежит')
     else
          write ('не принадлежит')
end.
```

Вариант 2.

Можно то же самое написать короче, использовав оператор логического **И** (and):

```
var x,y: real;
begin
    readln (x, y);
    if (y>=-x*x) and (y>=-x-2) and (y<=0)and (x<=0)
then
        write ('принадлежит')
    else
        write ('не принадлежит')
end.
```

5 Ответом является

1) любая пара (x, y), для которой выполняется $x>y^2$ или $x<0$	2) любая пара (x, y), для которой выполняется $x<=y^2$ и $y<0$ и $x>0$
	

Исправленные варианты программы.

Вариант 1.

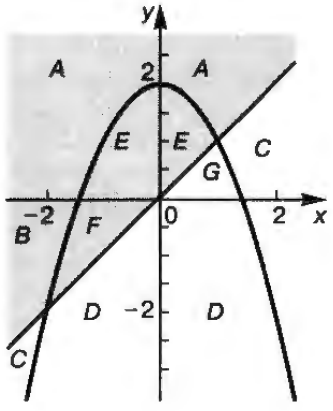
```
var x, y:real;
begin
    readln (x,y);
    if x<=y*y then
        if x>=0 then
            if y<=1 then
                if y>= 0 then
                    writeln ('принадлежит')
                else
                    writeln ('не принадлежит')
            else
                writeln ('не принадлежит')
        else
```

```
                writeln ('не принадлежит')
    else
        writeln ('не принадлежит')
end.
```

Вариант 2.

```
var x, y:real;
begin
 readln (x,y);
 if (x<=y*y) and (x>=0) and (y<=1) and (y>= 0) then
      writeln ('принадлежит')
 else
      writeln ('не принадлежит')
end.
```

6 Первый оператор **If** определяет область:

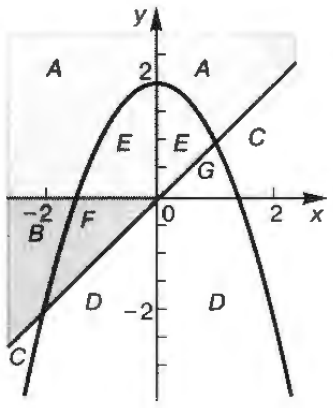



```
var x,y: real;
begin
     readln(x,y);
     if y>=x then
      write('принадлежит')
     else
      write('не принадлежит')
end.
```

Заполним первый столбец таблицы. В этом случае предполагается, что в программе проверяется только первое условие. «Да» означает, что будет напечатано «принадлежит», «нет» означает, что будет напечатано «не принадлежит».

Область	$y>=x$	$y>=0$	$y<=2-x*x$	Программа выведет	Область обрабатывается верно
A	Да				
B	Да				
C	Нет				
D	Нет				
E	Да				
F	Да				
G	Нет				

Следующий оператор **If** из этой области вырезает подобласть:

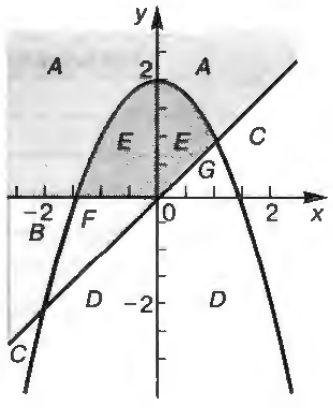



```
var x,y: real;
begin
     readln(x,y);
     if y>=x then
      if y>=0 then
            write('принадлежит')
      else
         write('не принадлежит')
end.
```

В программе присутствует один **Else**. По правилам языков программирования он относится к ближайшему **Then**, т.е. в данном случае на **Else** мы попадаем при y<0. Заполним второй столбец таблицы с учетом первого и второго условий. Все области, которые в предыдущем случае после проверки первого условия обрабатывались ветвью **Else**, при наличии двух условий обрабатываться не будут, так как для них ветвь **Else** не предусмотрена. В таблице для этих областей стоит прочерк.

Об-ласть	***y>=x***	***y >= 0***	***y<=2-x*x***	**Программа выведет**	**Область обрабатывается верно**
A	Да	Да			
B	Да	Нет			
C	Нет	–			
D	Нет	–			
E	Да	Да			
F	Да	Нет			
G	Нет	–			

Следующий оператор **If** из этой области выделяет подобласть:




```
var x,y: real;
begin
     readln(x,y);
     if y>=x then
      if y>=0 then
            if y<=2-x*x then
              write('принадлежит')
            else
          write('не принадлежит')
end.
```

Заполним третий столбец с учетом всех трех условий. Условию «$y<=2-x*x$» будут соответствовать только области «E»: области «C», «D» и «G» рассматриваться не будут, потому что были отсечены первым условием, а области «B» и «F» были отсечены вторым условием.

Область	$y>=x$	$y >= 0$	$y<=2-x*x$	Программа выведет	Область обрабатывается верно
A	Да	Да	Нет		
B	Да	Нет	–		
C	Нет	–	–		
D	Нет	–	–		
E	Да	Да	Да		
F	Да	Нет	–		
G	Нет	–	–		

Заполним последние два столбца таблицы. Программа ведет себя правильно, только если входные значения принадлежат областям «A» и «E». Во всех остальных областях программа не выведет ничего (области «B», «C», «D», «F» и «G»).

Область	$y>=x$	$y >= 0$	$y<=2-x*x$	Программа выведет	Область обрабатывается верно
A	Да	Да	Нет	не принадлежит	Да
B	Да	Нет	–	–	Нет
C	Нет	–	–	–	Нет
D	Нет	–	–	–	Нет
E	Да	Да	Да	принадлежит	Да
F	Да	Нет	–	–	Нет
G	Нет	–	–	–	Нет

Исправляем ошибки.

1) **Else** должен быть добавлен после каждого **Then**;

2) область «F» в условии задачи закрашена и при этом для нее выполняется условие «$y<0$». Следовательно, во втором операторе **If** в **Else** должен быть вложенный оператор **If**, который проверяет условие принадлежности области «F»;

3) область «G» в условии также закрашена, и при этом для нее выполняется условие «$y<x$». Это означает, что в **Else** для первого оператора **If** должны быть вложенные операторы **If**, которые проверяют условия принадлежности области «G».

Исправленная программа.

Вариант 1.

```
var x,y: real;
begin
     readln(x,y);
     if y>=x then
          if y>=0 then
               if y<=2-x*x then
                    write('E: принадлежит')
               else
                    write('A: не принадлежит')
          else
               if y<=2-x*x then
                    write('F: принадлежит')
               else
                    write('B: не принадлежит')
     else
          if y>=0 then
               if y<=2-x*x then
                    write('G: принадлежит')
               else
                    write('C: не принадлежит')
          else
               if y<=2-x*x then
                    write('D: не принадлежит')
               else
                    write('C: не принадлежит')
end.
```

Вариант 2.

Можно то же самое написать короче, использовав оператор логического **И (AND)**.

Для области «E» должны выполняться условия:

```
(y>=x) and (y>=0) and (y<=2-x*x)
```

Для области «F» должны выполняться условия:

```
(y>=x) and (y<0) and (y<=2-x*x)
```

Для области «G» должны выполняться условия:

```
(y<x) and (x>0) and (y<=2-x*x) and (y>0)
```

«Принадлежит» надо печатать для объединения (**OR**) областей «E», «F» и «G». У нас получилось логическое выражение, которое можно упростить несколькими способами.

```
((y>=x) and (y>=0) and (y<=2-x*x)) or ((y>=x) and
(y<0) and (y<=2-x*x)) or ((y<x) and (x>0) and (y<=2-
x*x) and (y>0)) = (y<=2-x*x) and ((y>=x) and (y>=0)
or (y>=x) and (y<0) or (y<x) and (x>0) and (y>0)) =
(y<=2-x*x) and ((y>=x) or (y<x) and (x>0) and (y>0))
```

В результате получаем достаточно компактную программу:

```
var x,y: real;
begin
    readln(x,y);
    if (y<=2-x*x) and ((y>=x) or
    (y<x) and (x>0) and (y>0)) then
        write('принадлежит')
    else
        write('не принадлежит')
end.
```

7 Ответом является:

Область	$y>=0$	$x>=0$	$y<=\text{abs}(x-2)$	Программа выведет	Область обрабатывается верно
A	да	нет	–	–	нет
B	да	нет	–	–	нет
C	да	да	Нет	Не принадлежит	да
D	да	да	Да	принадлежит	да
E	да	да	Да	принадлежит	нет
F	нет	–	–	–	нет

```
var x, y: real;
begin
    readln(x, y);
    if (y>=0) and (x>=0) and (Y<=abs(x-2))and (x<=2)
    then write ('принадлежит')
    else  write ('не принадлежит');
end.
```

8 Ответом является:

Область	$y >= x$	$y >= 0$	$y <= \cos(x)$	Программа выведет	Область обрабатывается верно
A	да	неизв	-	-	нет
B	нет	-	-	-	нет
C	нет	-	-	-	нет
D	да	нет	-	-	нет
E	да	да	да	принадлежит	да
F	нет	-	-	-	нет

```
var x, y: real;
begin
    readln(x, y);
    if (y >= x) and (y>=0) and (y <= cos(x)
       and (y > - pi /2)
    then write('принадлежит')
    else write('не принадлежит');
end.
```

9 Ответом является:

Область	$y <= x$	$y <= -x$	$y >= x*x-2$	Программа выведет	Область обрабатывается верно
A	нет	-	-	-	нет
B	нет	-	-	-	нет
C	да	нет	-	-	нет
D	да	да	да	принадлежит	да
E1	нет	-	-	-	нет
E2	да	нет	-	-	нет
F	да	да	нет	не принадлежит	да

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

10

```
const
N=20;
maxValue=1000; {все заранее известные параметры всегда
лучше описывать в виде констант в начале программы}
```

```
var
a: array [1..N] of integer;
i, min: integer;
begin
  for i:=1 to N do
    readln(a[i]); {ввели значения массива}
    min:=maxValue; {вначале присваиваем минимуму
                    максимально возможное значение}
      for i:=1 to N do {проверяем одновременное
            (and) выполнение следующих условий}
    if (a[i] mod 2=0)   {очередной элемент массива
                         четный}
    and (a[i] mod 3<>0)  {очередной элемент массива
                          не делится на 3}
    and (a[i]<min)     {очередной элемент массива
                меньше всех предыдущих значений}
    then
      min:=a[i]; {у нас новый минимум}
  writeln(min);
end.
```

11

```
const N=30;
var
  a: array [1..N] of integer;
  i,
  s:   integer;   {Сумма   отрицательных   элементов
                   массива}
begin
  for i:=1 to N do
    readln (a[i]);
  s:=0;
  for i:=1 to N do
    if a[i]<0 then
      s:= s + a[i];
  if   s=0   then   {Если   сумма   не   изменилась,
                     то отрицательных элементов нет}
    writeln ('отрицательных элементов нет')
  else
    writeln (s)
end.
```

12

```
const N=28;
var
   a: array [1..N] of integer;
```

```
   i, j, min: integer;
begin
     for i:=1 to N do
         readln (a[i]);
     min:= 100;
{Присвоим переменной, в которой будет находиться минимальная положительная оценка, максимально возможную положительную оценку. Теперь в цикле по всем элементам массива найдем, нет ли среди значений меньшей положительной оценки, чем максимально возможная. Напомним, что в данном случае оценка считается положительной, если она больше 40}
     for i:=1 to N do
         if (a[i] >= 40) and (a[i] < min) then
            min := a[i];
     writeln (min);
end.
```

13 **Способ 1.**

Единственный счетчик будем использовать и в качестве признака того, что имеется отрицательное число, и для определения номера отрицательного числа. Начальное значение счетчика i положим равным 1. Счетчик i будем увеличивать до тех пор, пока либо не дойдем до конца массива ($i \geq N$), либо пока не встретим отрицательное число ($a[i]<0$). После этого проверим знак $a[i]$. Если $a[i]$ – положительное, то это означает, что мы вышли из цикла по первому условию ($i \geq N$) и отрицательных чисел нет. Если $a[i]$ отрицательное, то $a[i]$ – первое встретившиеся отрицательное число и i – это наименьший номер отрицательного элемента.

```
const
  N=50;
var
  a: array [1..N] of real;
  i: integer;      {Наименьший номер
                   отрицательного элемента}
begin
  for i:=1 to N do
    readln (a[i]);
  i:=1;      {Вначале предполагаем,
             что это первый элемент}
  while (i<N) and (a[i]>=0) do {Цикл выполняется
  либо до конца массива, либо пока не найдем
  отрицательное число}
    i := i +1;
  if a[i]>=0 then
```

```
    writeln ('нет таких')
  else
    writeln (i);
end.
```

Способ 2.

Одну переменную будем использовать в качестве признака, имеется ли хотя бы одно отрицательное число. Другую переменную будем использовать для определения номера отрицательного числа.

```
const
 N=50;
var
  a: array [1..N] of real;
  i: integer;      {Наименьший номер
                   отрицательного элемента}
  j: Boolean;      {Признак, есть ли
                   отрицательное число}
begin
  for i:=1 to N do
    readln (a[i]);
  j:=false; {Вначале предполагаем,
      что отрицательных чисел нет}
  i:=1;
  while (i<N) and (a[i]>=0) do {Цикл выполняется
      либо до конца массива, либо пока не найдем
      отрицательное число}
    begin
      i := i +1;
      j := true
    end;
  if j then
    writeln (i)
  else
    writeln ('нет таких')
end.
```

14 **Способ 1.**

Решим задачу с использованием двух циклов. В первом цикле с предусловием найдем первое положительное число. Затем в цикле с параметром, начиная с этого номера до конца массива, будем искать положительное число, меньшее первого.

```
const
  N=40;
var
  a: array [1..N] of real;
```

```
   i, j: integer;
   min: real;
begin
   for i:=1 to N do
     readln (a[i]); {Введем массив}
   j:=1;
   while (j<N) and (a[j]<=0) do {Цикл либо до конца
            массива, либо до первого положительного
            числа}
     j:=j+1;
   if a[j] <= 0 then {Последний просмотренный
         элемент в предыдущем цикле - отрицательный,
         т.е. в предыдущем цикле дошли до конца
массива. Следовательно, положительных элементов нет}
     writeln ('такого элемента нет')
else {Положительные элементы есть. Они начинаются
с номера j. Найдем номер минимального элемента и
присвоим его j}
   begin
     for i:=j to N do
       if (a[i] > 0) and (a[i] < a[j])
       then
         j:=i;
     writeln (a[j])
   end;
end.
```

Способ 2.

Найдем минимальное положительное число с помощью единственного цикла с параметром. Первый оператор **If** означает, что поиск ведется только среди положительных чисел. Номер минимального положительного числа запоминаем в переменной *j*. Вначале *j* присваиваем 0 – признак того, что положительных чисел пока не нашли. Во вложенном операторе **If** проверяем выполнение хотя бы одного из двух условий: еще не было положительных чисел или если нашли положительное число, меньшее минимального найденного к настоящему моменту числа, номер которого запомнили в *j*.

```
const
   N=40;
var
   a: array [1..N] of real;
   i, j: integer;
   min: real;
begin
   for i:=1 to N do
```

```
      readln (a[i]); {Введем массив}
   j:=0;
   for i:=1 to N do
      if a[i]>0 then
         if (j=0) or (a[i]<a[j]) then
{Последовательность проверки условий важна! Если
поставить первым проверку условия (a[i]<a[j]),
то будет ошибка выход за границы массива при
первом положительном элементе, так как наш массив
начинается с 1}
            j:=i;
   if j=0 then
      writeln ('такого элемента нет')
   else
      writeln (a[j])
end.
```

15 Требуется во-первых, найти элемент, равный *X*, а во-вторых, этот элемент должен иметь минимальный номер среди всех элементов, равных *X*.

Способ 1.

Будем использовать цикл с параметром и просматривать массив с конца. В этом случае последним найденным элементом, равным *X*, будет элемент с наименьшим номером.

```
const
    N=30;
var
    a: array [1..N] of integer;
    i, j, x: integer;
begin
    for i:=1 to N do readln (a[i]);
    readln (x);
    j:=0;
    for i:= N downto 1 do
        if a[i]=x then
            j:=i;
    if j=0 then
        writeln ('нет таких элементов')
    else
        writeln (j);
end.
```

Способ 2.

Будем использовать цикл с предусловием до тех пор, пока либо не дойдем до конца массива, либо не встретим элемент массива, не равный *X*.

```
const
    N=30;
```

```
var
    a: array [1..N] of integer;
    i, j, x: integer;
begin
    for i:=1 to N do readln (a[i]);
        readln (x);
    i:=1;
    while (i<N) and (a[i]<>x) do
        i:= i+1;
    if a[i] <> x then
        writeln ('нет таких элементов')
    else
        writeln (i);
end.
```

16 В переменной *max* будем хранить текущее максимальное значение, а в переменной *max*2 – текущее второе по величине максимальное значение.

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, k, max, max2: integer;
begin
    for i:=1 to N do readln (a[i]);
        max:= a[1];    {Предполагаем, что текущий
                       максимальный элемент - первый}
        max2:= a[2];   {а текущий второй по величине
                        элемент - второй}
    if max < max2 then {Если первый элемент меньше
второго, то предположения были неверными, и надо
поменять местами значения max и max2}
        begin
            max:= a[2];
            max2:= a[1]
        end;
    for i:=3 to N do   {Начиная с третьего элемента
    сравниваем значения элемента с текущим
    максимумом и текущим вторым максимумом}
    if a[i] > max then
        begin
            max2:= max;
            max:= a[i]
        end
    else
        if a[i] > max2 then
            max2:= a[i];
```

```
    writeln (max2);
end.
```

17

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, j, k: integer;
begin
    for i:=1 to N do
        readln (a[i]);
    k:=0;   {Счетчик положительных элементов}
    for i:=1 to N do
    if a[i] > 0 then
    begin
        k:= k+1;
        if k=3 then
        j:=i;    {j - номер третьего по порядку
                  положительного элемента}
    end;
    if k<3 then
        writeln ('такого элемента нет')
    else
        writeln (j);
end.
```

18

```
const
    N=40;
var
    a: array [1..N] of integer;
    i,
    l,   {Количество элементов текущей возрастающей
         последовательности подряд идущих элементов}
    lmax,    {Количество элементов в максимально
              длинной последовательности}
    s,   {Сумма элементов текущей возрастающей
       последовательности подряд идущих элементов }
    smax: integer;     {Сумма элементов в
            максимально длинной последовательности }
begin
    for i:= 1 to N do readln (a[i]);
    lmax:=0;
    l:=1;
    s:= a[1];
    for i:=2 to N do
        if a[i] > a[i-1] then
```

```
        begin   {Последовательность возрастающая,
          увеличиваем количество элементов и сумму}
            l:=l+1;
            s:= s+a[i]
        end
        else{Последовательность перестала быть
                                        возрастающей}
        begin
            if l>lmax then
            begin     {Текущая возрастающая
      последовательность содержит больше элементов,
      чем максимально длинная}
                lmax:= l;
                smax:=s
            end;
            l:=1;
            s:=a[i]
        end;
    if l>lmax then           {Длину последней
возрастающей последовательности мы в цикле не
сравнивали с максимально длинной. Это надо сделать
сейчас.}
        smax:=s;
    writeln (smax)
end.
```

19

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, j,
    min,     {Номер наименьшего элемента}
    min2,    {Номер второго по величине элемента}
    s: integer;  {Сумма двух элементов}
begin
    for i:=1 to N do readln (a[i]);
    min:=1;  {Предполагаем, что минимальный элемент
             - первый}
    min2:=2;     {Предполагаем, что следующий
                 минимальный элемент - второй}
    if a[min] > a[min2] then{Если предположения
оказались неверными, то номер наименьшего элемента -
2 номер второго по величине - 1}
    begin
        min:=2;
        min2:=1
    end;
```

```
    for i:=3 to N do
        if a[i] < a[min] then     {Нашли элемент,
        значение которого меньше, чем предыдущий
        минимум}
        begin
            min2 := min;
            min := i
        end
        else
            if a[i] < a[min2] then {Нашли элемент,
                значение которого меньше значения
                второго по величине элемента}
                min2:=i;
    writeln (min, min2);
end.
```

20

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, j,
    min,     {Номер наименьшего элемента}
    min2,    {Номер второго по величине элемента}
    s: integer;  {Сумма двух элементов}
begin
    for i:=1 to N do readln (a[i]);
    min:=1;       {Предполагаем, что минимальный
                  элемент - первый}
    min2:=2; {Предполагаем, что следующий
             минимальный элемент - второй}
    if a[min] > a[min2] then{Если предположения
        оказались неверными, то номер наименьшего
        элемента - 2 номер второго по величине - 1}
    begin
        min:=2;
        min2:=1
    end;
    for i:=3 to N do
        if a[i] < a[min] then     {Нашли элемент,
        значение которого меньше, чем предыдущий
        минимум}
begin
            min2 := min;
        min := i
    end
    else
```

```
        if a[i] < a[min2] then    {Нашли элемент,
        значение которого меньше значения второго
        по величине элемента}
            min2:=i;
    writeln (min, min2);
end.
```

21

```
const
    N=40;
var
    a: array [1..N] of integer;
    i, j,
    min, min2,    {Номера элементов с минимальной
                  разницей}
    s: integer;   {Разница между двумя элементами}
begin
    for i:=1 to N do readln (a[i]);
    min := 1;
    min2 := 2;
    s := abs (a[1]-a[2);
    for i:=1 to N-1 do
    for j:=i+1 to N do {Сравним i-й элемент со
                  всеми элементами до конца массива}
        if abs (a[i]-a[j]) < s then    {Нашли два
                  элемента, у которых разница меньше}
        begin
            min := i;
            min2 := j;
            s := abs (a[i]-a[j])
        end;
    writeln (min, min2)
end.
```

22

Первым делом необходимо определить данные, которые понадобятся нам в программе. Единственная константа – это номер школы, который мы знаем заранее. Все остальные данные – это переменные.

Фамилию и имя ученика можно хранить в одной переменной типа **string**, так как и считывать, и печатать фамилию и имя мы будем вместе. Для фамилии и имени нам понадобятся три переменные – для текущего считанного значения имени ученика, для ученика, набравшего максимальный балл, и для ученика, набравшего второе по величине количество баллов. Текущую букву, цифру или пробел будем считывать в переменную типа **char**. Остальные переменные будут иметь тип **integer**.

Первым делом в программе необходимо обнулить все значения. Затем надо считать количество учеников, т.е. количество строк, которые будут введены далее.

Теперь пишем цикл по количеству строк. В цикле будем обрабатывать строку, формат которой определен в задании. Вначале в цикле ищем второй пробел, что означает, что введены фамилия и имя очередного ученика. Для ввода очередного символа используем процедуру **Read**, а не **Readln**, так как всё вводится в одной строке.

Следующие вводимые значения – номер школы и оценка ученика, которые являются целыми числами. Здесь уже используется процедура **Readln**, потому что оценка – это последнее значение в строке. Сразу же сравниваем введенный номер школы с интересующим нас номером (все остальные школы не рассматриваем). Сравниваем текущий балл с максимальным. Если нашли очередной максимум, то запоминаем это в соответствующих переменных – фамилию и имя ученика и значение максимального балла. Пока данный максимальный балл набрал один ученик, поэтому присваиваем переменной, в которой подсчитываем количество учеников, набравших максимальный балл, значение 1. Предыдущий максимум при этом становится вторым по величине значением. Если текущий балл равен максимальному, то увеличиваем количество учеников, набравших максимальный балл. Если текущий балл равен второму по величине значению, то увеличиваем число учеников, набравших второе по величине количество баллов.

В последней части программы надо вывести значения, которые требуются в задании. Вся информация для этого у нас уже есть – максимальный балл; фамилия и имя ученика, первого в списке с таким баллом; количество учеников, набравших максимальный и второй по величине баллы.

```
const ourSchool = 50; {Номер школы задается в виде константы,
    потому что он известен заранее}
var currentName, {фамилия и имя ученика, введенные в текущей
    строке}
    maxName1,     {фамилия и имя ученика, набравшего максимальный
    балл}
     maxName2: string [52]; {фамилия и имя второго ученика,
    набравшего максимальный балл. Длина строк равна 52, так как
    максимальная длина имени 20 символов, максимальная длина
    фамилии 30 символов, между фамилией и именем один пробел,
    признаком конца имени также является пробел, который поместим
    в строку, так как у нас будет цикл с постусловием}

    currentLetter: char; {текущая считанная буква имени или
    фамилии}

    i,      {номер текущей разбираемой строки}
    N,      {количество вводимых строк, т.е. длина списка
    учеников}
```

```
schoolNumber,      {номер школы в текущей строке, нам нужна школа
    №50, поэтому будем сравнивать его с 50}
    estimate,            {балл текущего ученика}
    maxEstimate1,        {максимальный балл}
    sumMax1,             {число учеников, набравших maxEstimate1
     баллов}
    maxEstimate2,        {второй по величине балл}
     sumMax2: integer; {число учеников, набравших maxEstimate2
     баллов}
begin
{Обнулим все значения}
     maxEstimate1:=0;
     maxEstimate2:=0;
     maxName1:='';
     maxName2:='';
     sumMax1:=0;
     sumMax2:=0;

{Введем количество учеников}
     readln (N);

{Цикл по всему списку}
     for i:=1 to N do
     begin
           currentName:=''; {Значение должно обнуляться перед
    считыванием очередной строки списка учеников}
          repeat
               read (currentLetter);
               currentName := currentName + currentLetter
           until currentLetter =' '; {Ввели фамилию очередного
    ученика}
          repeat
               read (currentLetter);
               currentName := currentName + currentLetter
             until currentLetter =' '; {Ввели имя очередного
    ученика}

          readln (schoolNumber, estimate); {Ввели номер школы и
    оценку текущего ученика}

          if schoolNumber = ourSchool then {анализируем оценки
    только учеников своей школы}
              if estimate > maxEstimate1 then {текущий балл -
    лучший}
            begin
                 maxEstimate2 := maxEstimate1; {максимальный
```

```
балл становится вторым по величине}
            maxName2 := maxName1; {фамилия и имя ученика,
набравшего второй по величине балл}
              sumMax2 := sumMax1; {количество учеников,
набравших второй по величине балл}
           maxEstimate1 := estimate; {максимальным баллом
становится балл текущего ученика}
         maxName1 := currentName; {фамилия и имя ученика,
набравшего текущий балл}
            sumMax1 := 1 {пока такой ученик один}
        end
        else {текущий балл - не лучший}
            if estimate = maxEstimate1 then {текущий балл
- такой же, как лучший}
            begin
              sumMax1 := sumMax1 + 1; {считаем количество
учеников, получивших лучший балл}
               maxEstimate2 := maxEstimate1; {запоминаем
лучший балл}
              maxName2 := currentName {запоминаем фамилию
и имя ученика, набравшего лучший балл}
            end
            else
                if estimate = maxEstimate2 then {текущий
балл - такой же, как второй лучший}
                begin
                        sumMax2 := sumMax2 + 1; {считаем
количество учеников, получивших второй лучший балл}
                 maxEstimate2 := maxEstimate1; {запоминаем
второй лучший балл}
                      maxName2 := currentName {запоминаем
фамилию и имя ученика, набравшего второй лучший балл, если
ученик с максимальным баллом будет один, и ученик со вторым
лучшим баллом будет один, то напечатаем его фамилию и
имя}
                end
                else
                  if estimate > maxEstimate2 then {текущий
балл - больше второго по величине балла}
                    begin
                     sumMax2 := 1; {начинаем заново считать
количество учеников, набравших второй лучший балл}
                      maxEstimate2 := estimate; {запоминаем
второй лучший балл}
                       maxName2 := currentName {запоминаем
фамилию и имя ученика, набравшего второй лучший балл}
```

```
                        end
                        else
                              if estimate = maxEstimate2 then
{текущий балл такой же, как второй лучший}
                                 sumMax2 := sumMax2 +1 {считаем
количество учеников, набравших второй лучший балл. Если
таких учеников окажется больше одного, то их фамилии и
имена печатать не будем}
      end;
       if (sumMax1 = 2) or (sumMax1 = 1)and (sumMax2 = 1)
then {два лучших ученика: два ученика набрали лучший балл
ИЛИ один ученик набрал лучший балл, и один ученик набрал
второй лучший балл}
      begin
            writeln (maxName1); {напечатаем их фамилии и
имена}
          writeln (maxName2)
      end
      else
           if (sumMax1 = 1) and (sumMax2 > 1) then  {один
ученик набрал лучший балл, но второй по величине балл
набрали несколько учеников, поэтому напечатаем только
одного ученика с лучшим баллом}
             writeln (maxName1)
         else {Здесь мы окажемся, если количество учеников,
получивших максимальный балл, больше 2, или количество
учеников, получивших второй по величине балл, равно нулю.
Второй вариант исключен, потому что по условию сдавало
больше 5-ти учеников школы}
              writeln (sumMax1)
end.
```

23 Программа читает входные данные по строкам, не запоминая их в массиве. Будем составлять только список встретившихся задач и количество запросов по каждой из них. После чтения очередной строки с названием задачи просматривается список уже введенных задач. Если данная задача есть в списке, то количество запросов для нее увеличивается на 1. В противном случае название задачи добавляется в список, длина которого по условию задачи не должна превышать 11.

После окончания ввода производится сортировка получившегося списка по количеству запросов для каждой задачи. Затем выводится список из трех задач с указанием частоты встречаемости или весь список, если его длина меньше трех.

```
var   N,     {Количество запросов}
      Num,   {Количество разных названий задач}
```

```
    i,      {Сначала счетчик по всему количеству запросов, по-
    том индекс цикла при упорядочивании списка названий задач
    по количеству повторений}
    j,      {Сначала текущий индекс в списке уже встретивших-
    ся названий задач, потом индекс цикла при упорядочивании
    списка названий задач по количеству повторений}
    t: integer; {Вспомогательная переменная, чтобы поменять
    местами количество повторений при упорядочивания массива
    Count по числу повторений. Менять местами элементы масси-
    ва taskList будем с помощью переменной currentTask}
    Count: array[1..11] of integer; {Число повторений каждого
    названия задачи}
    currentTast: string;      {Название текущей считанной за-
    дачи}
    taskList: array[1..11] of string;     {Список названий за-
    дач}
begin
      ReadLn (N);        {Ввели количество запросов}
      for i:=1 to N do
      begin
      ReadLn(currentTask); {Ввели очередное название задачи}
                         {Ищем его? поиск в списке уже встре-
тившихся}
            j:=1;
      {Цикл выполняется либо до конца списка заполненных на-
званий задач (j>Num), либо пока не найдем название считанной
задачи в списке уже встретившихся (currentTask=taskList[j])}
            while (j<=Num) and (currentTask<>taskList[j]) do
            j:=j+1;
            if j<=Num then {Данное название уже есть в списке,
                        поэтому увеличиваем счетчик повторений}
                  Count[j]:=Count[j]+1
            else  {Данного названия задачи в списке нет, по-
            этому добавляем название задачи в конец списка,
            счетчик повторений устанавливаем в 1, увеличиваем
            количество разных названий задач}
            begin
                  taskList[j]:=currentTask;
                  Count[j]:=1;
                  Num:=Num+1
            end
      end;
    {Сортируем массивы taskList и Count в порядке убывания
    значений массива Count. Начиная с последнего элемента
    поднимаем вверх текущий максимальный элемент}
      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;
                  currentTask:=taskList[j];
                  taskList[j]:=taskList[j-1];
                  taskList[j-1]:=currentTask;
            end;
{Должны вывести первые три значения упорядоченного массива taskList}
            if Num >= 3 then   j := 3
            else
                  j := Num;
                  i := 1;
    {Печатаем элементы массива taskList, если значение
    элемента массива Count для них не меньше,
    чем третьего или последнего (если всего названий
    меньше трех) элемента массива Count}
              while (i <= Num) and (Count[i] >= Count[j])
              do
              begin
                 WriteLn(taskList[i], ' ', Count[i]);
                 i := i + 1;
              end
    end.
```

24

```
var
    middleBall, {Средний балл в каждой школе}
    studNumber: array [1..99] of integer; {Количество
        учеников в каждой школе, сдававших информатику}
    ch: char; {Очередная буква имени и фамилии. Ни имя,
        ни фамилию нам нигде использовать не надо,
        поэтому просто будем считывать их, запоминать
        нигде не будем}
    i,          {Счетчик циклов}
    N,          {Количество вводимых строк}
    shNumber,   {Номер очередной школы}
    ball,  {Балл очередного ученика}
    maxBall,    {Макс. средний балл среди всех школ}
    maxSchNumber: integer; {Количество школ
                          с максимальным средним баллом}
```

```
    begin
     for i:=1 to 99 do {Обнуляем массивы}
     begin
          middleBall [i] := 0;
          studNumber [i] := 0
     end;
     readln (N);              {Ввели количество строк}
     for i:=1 to N do     {Перебираем все входные строки}
     begin
       repeat read (ch) {Имя вводится в той же
                         строке, поэтому используем read(),
                         а не readln()}
       until ch=' ';  {Введена фамилия}
       repeat read (ch) {Номер школы вводится в той
                  же строке, поэтому используем read(),
                  а не readln()}
       until ch=' ';  {Введено имя}
       readln (shNumber, ball); {Ввели номер школы и
балл ученика}
       middleBall [shNumber] := middleBall [shNumber]+
          ball; {Считаем сумму баллов каждой школы}
       studNumber [shNumber] := studNumber [shNumber]+1;
             {Считаем количество учеников каждой школы}
    end;
    for i:=1 to 99 do
     if studNumber [i] >0 then {Хотя бы один ученик i-й
                                школы сдавал информатику}
      middleBall[i] :=middleBall[i] div studNumber[i];
                   {Считаем средний балл по i-й школе}
    maxBall := 1;
    maxSchNumber := 1;
    for i:=2 to 99 do {Ищем максимум среди средних баллов}
       if middleBall [i] > middleBall [maxBall] then
         begin
          maxBall := i;
          maxSchNumber := 1
         end
       else
         if middleBall[i] = middleBall[maxBall] then
                          {Считаем количество максимумов}
            maxSchNumber := maxSchNumber +1;
```

```
    if maxSchNumber = 1 then
          writeln (maxBall, ' ',middleBall [maxBall])
    else
          writeln (maxSchNumber)
    end.
```

25

```
var
    schoolList, {Список школ}
    studNumber: array [1..99] of integer; {Число учеников
                каждой школы, сдававших ЕГЭ по информатике}
    ch: char; {Очередная буква имени и фамилии. Ни имя,
          ни фамилию нам нигде использовать не надо,
          поэтому просто будем считывать их, запоминать
          нигде не будем}
    i,          {Индекс массива}
    N,          {Число вводимых строк}
    shNumber, {Номер школы в очередной считанной строке}
    ball,       {Балл очередного ученика}
    avg,        {Средний балл по району}
    bestSchool: integer;   {Количество школ, в которых
                           средний балл выше среднего}
begin
    for i:=1 to 99 do {Обнуляем массивы}
    begin
     schoolList [i] :=0;
     studNumber [i] :=0
    end;
    readln (N);          {Ввели количество строк}
    for i:= 1 to N do   {Перебираем все входные строки}
     begin
        repeat read (ch)
        until ch=' '; {Введена фамилия}
        repeat read (ch)
        until ch=' '; {Введено имя}
        readln (shNumber, ball); {Ввели номер школы и
                                  балл ученика}
        schoolList [shNumber] := schoolList [shNumber]+
        ball;        {Считаем сумму баллов по школе}
        studNumber [shNumber] := studNumber [shNumber]+1
              {Считаем количество учеников из школы}
     end;
```

```
    avg := 0;
    for i:=1 to 99 do
     if studNumber [i] > 0 then
     begin
        avg := avg + schoolList [i]; {Считаем сумму
                                      баллов по району}
        schoolList [i]:=schoolList [i] div studNumber [i]
                {Считаем средний балл по каждой школе}
     end;
    avg := avg div N; {Считаем средний балл по району}
    bestSchool:=0;
    for i:=1 to 99 do
     if schoolList [i] > avg then {Отбираем школы, где
                          средний балл выше районного}
     begin
        bestSchool:= bestSchool +1; {Подсчитываем
                                    количество таких школ}
        ball := schoolList [i]; {Запоминаем средний балл
        какой-нибудь из них, т.к. средний надо выводить
        только в том случае, если такая школа одна}
        write (i, ' ')
     end;
    writeln;
    if bestSchool =1 then
     writeln ('Средний балл = ', ball)
end.
```

26

```
var
    studentNumber,    {Число учеников в каждой школе,
                       сдававших информатику}
    maxBall,     {Наибольший балл в каждой школе}
    topStudent:array [1..99] of integer;    {Число учени-
    ков в каждой школе, набравших наибольший балл в школе}
    bestStudents: array [1..99] of string [52];
                {Фамилия лучшего ученика в каждой школе,
                набравшего набравшего наибольший балл}
    name:string [52]; {Очередная считанная фамилия, ее
                запомним в массиве bestStudent, если
                ученик набрал максимальный балл по школе}
    ch: char;    {Очередной считанный символ фамилии или
                имени}
```

```
    i,
    N,     {Количество входных строк}
    schoolNumber,     {Номер школы}
    ball:integer;     {Балл очередного ученика}
begin
    for i:=1 to 99 do {Обнуляем массивы}
    begin
     studentNumber[i] := 0;
     topStudent [i] := 0;
     maxBall [i] := -1
    end;
    readln (N);       {Ввели количество строк}
    for i:=1 to N do   {Перебираем все входные строки}
    begin
     name := '';
     repeat
        read (ch);
        name := name+ch
     until ch = ' '; {Введена фамилия и записана
                      в переменную name}
     repeat  read (ch)
     until ch = ' '; {Введено имя, его записывать
                      никуда не требуется}
     readln (schoolNumber, ball);     {Ввели номер школы
                                       и балл ученика}
     if ball > maxBall [schoolNumber] then {Если текущий
                балл лучше максимального балла учеников
                данной школы}
     begin
        maxBall [schoolNumber] := ball;    {Запоминаем
                            новый лучший балл по школе}
        bestStudents [schoolNumber] := name;
                                  {и фамилию ученика}
        topStudent [schoolNumber] := 1     {Сбрасываем
                счетчик учеников, набравших наибольший
                балл по школе}
     end
     else
        if ball = maxBall [schoolNumber] then    {Если
                      текущий балл равен наибольшему}
        topStudent [schoolNumber] := topStudent
        [schoolNumber] +1; {Увеличиваем число учеников,
```

```
        получивших максимальный балл по школе}
        studentNumber [schoolNumber] := studentNumber
        [schoolNumber] +1  {Подсчитываем количество уче-
        ников из каждой школы, сдававших информатику}
     end;
     for i:=1 to 99 do
        if studentNumber [i] >= 3 then {Выбираем только
        школы, из которых сдавало больше трех учеников}
          if topStudent [schoolNumber] >1 then
         {Если учеников с максимальным баллом несколько}
               writeln (i, ' ', topStudent[i])
          else
               writeln (i, ' ', bestStudents[i])
end.
```

27

```
var
    studentNumber,    {Число учеников в каждой школе,
                      сдававших информатику}
    maxBall: array [1..99] of integer;      {Наибольший
                                    балл в каждой школе}
    ch:char;{Очередной считанный символ фамилии или имени}
    i,
    N,                {Количество входных строк}
    schoolNumber,     {Номер школы}
    ball,             {Балл очередного ученика}
    schoolQuantity: integer;     {Количество школ, в кото-
    рых больше двух учеников набрали максимальный балл}
begin
    for i:= 1 to 99 do      {Обнуляем массивы}
    begin
     studentNumber [i] := 0;
     maxBall [i] := -1
    end;
    readln (N);        {Ввели количество строк}
    for i:=1 to N do        {Перебираем все входные стро-
    ки}
    begin
     repeat
        read (ch)
     until ch = ' ';        {Введена фамилия}
```

```
    repeat
       read (ch)
    until ch = ' '; {Введено имя, ни фамилию, ни имя
                запоминать не надо - они в дальнейшем
                нигде не используются}
    readln (schoolNumber, ball);      {Ввели номер школы
                                      и балл ученика}
    if ball > maxBall [schoolNumber] then {Сравниваем
                текущий балл с лучшим баллом по школе}
    begin
       maxBall [schoolNumber] := ball;
                            {Меняем лучший балл по школе}
       studentNumber [schoolNumber] := 1  {Число
            учеников в данной школе с таким баллом =1}
  end
  else
    if ball = maxBall [schoolNumber] then {В данной
          школе уже были ученики, набравшие такой балл}
    studentNumber [schoolNumber] := studentNumber
    [schoolNumber] +1;    {Увеличиваем количество
                учеников в данной школе с таким баллом}
  end;
  schoolQuantity:=0;
  for i:=1 to 99 do
    if studentNumber [i] > 2 then  {Ищем школы, у
          которых максимальный балл набрало больше двух
          учеников}
    begin
       schoolQuantity:=schoolQuantity+1; {Подсчитываем
       количество таких школ}
       ball := maxBall [i];     {Запоминаем балл в
       какой-нибудь из них. Если таких школ окажется
       одна, нам надо будет распечатать ее балл}
       write (i, ' ')      {Выводим номера школ, через
                           пробел}
  end;
  if schoolQuantity=0 then
    writeln ('Нет школ, в которых больше двух учеников
                набрали максимальный балл')
  else
  begin
    writeln;
    if schoolQuantity=1 then
          writeln ('Максимальный балл = ', ball)
  end
end.
```

28

```
var
    s: string;          {Введенная строка}
    flag: boolean;      {Признак того, что текущий символ
                внутри слова. Если false - то предыдущий
                символ не был буквой}
    i,
    k,
    wordLen: integer; {Длина очередного слова}
begin
    readln (s);
    flag := false;
    for i:=1 to length (s) do
    begin
     if (upcase (s[i]) >= 'A') and (upcase (s[i]) <= 'Z')
     then  {Очередной символ в считанной строке является
           буквой}
        if flag then  {Символ внутри слова, надо подсчи-
        тывать длину слова}
          wordLen := wordLen+1
        else
        begin   {Это начало нового слова}
          flag := true;
          wordLen := 1
        end
     else
        if flag then  {У нас встретилась не буква и
        установлен признак того, что предыдущим символом
        была буква. Поэтому слово закончилось, надо при-
        ступать к кодированию}
        begin
          flag := false;
          for k:=1 to wordLen do
          begin
            if ord (upcase (s[i-k])) - ord ('A') +
            wordLen >25 then {Необходим циклический
            сдвиг, т.е. сдвиг на wordLen-26 символов
            влево}
              s [i-k] := chr (ord (s[i-k]) + wordLen -26)
            else      {Очередному символу просто
                      присваиваем значение, сдвинутое на
                      wordLen символов вправо}
                s [i-k] := chr (ord (s [i-k]) + wordLen)
          end
        end
    end;
```

```
    writeln (s)
end.
```

29

```
var
    sameGrade: array [0..100] of integer;  {sameGrade[i]
                  - число учеников, набравших балл «i»}
    ch: char;
    i,
    N,
    schoolNumber,     {Номер школы}
    ball,
    twentyPercent,    {Число учеников, составляющих 20%
                           от всего количества учеников}
    bestStudent: integer;  {Сколько на самом деле
                получилось отличников в соответствии с
                требуемыми критериями}
begin
    for i:=1 to 100 do     {Обнуляем массив, т.к. пока
    не знаем сколько учеников набрало определенный балл}
     sameGrade [i] := 0;
    readln (N);       {Ввели количество строк}
    for i:=1 to N do       {Анализируем каждую входную
                           строку}
    begin
     repeat
          read (ch)
     until ch = ' ';  {Введена фамилия, нигде ее
                     не запоминаем, т.к. она нам не нужна}
     repeat
          read (ch)
     until ch = ' ';  {Введено имя, нигде его не
                      запоминаем, т.к. оно нам не нужно}
     readln (schoolNumber, ball);     {Ввели номер школы
                                        и балл ученика}
     sameGrade [ball] := sameGrade [ball] +1
                      {Подсчитываем количество учеников,
                      набравших данный балл}
    end;
    twentyPercent := N div 5;    {Определили, сколько
                                 должно быть отличников}
    bestStudent := 0;
```

```
    i := 101;
    while bestStudent < twentyPercent do    {Подсчитываем
    количество учеников, набравших максимальные баллы до
    тех пор, пока их число не станет больше или равным
    20% от общего числа учеников}
    begin
     i := i -1;
     bestStudent := bestStudent + sameGrade [i]
    end;
    if bestStudent = twentyPercent then     {«отлично»
    можно поставить ровно 20% учеников}
     writeln (i)
    else
     if sameGrade [i] = bestStudent then    {наибольший
                        балл набрало больше 20% учеников}
          writeln (i)
     else  {Ученики, набравшие «i» баллов, не получат
«отлично», т.е. «отлично» получат меньше 20%}
     begin {Надо найти следующий по величине балл, кото-
рый набрал хотя бы один ученик}
         i := i +1;
         while sameGrade [i]=0 do
           i:=i+1;
         writeln (i)
     end
end.
```

30

```
var
    sameGrade: array [0..100] of integer;  {sameGrade[i]
    - число учеников, набравших балл «i»}
    ch: char;
    i,
    N,
    schoolNumber,     {Номер школы}
    ball,
    studentNumber,
    studentRating: integer;
begin
    for i:=0 to 100 do     {Обнуляем массив, т.к. пока
    не знаем сколько учеников набрало определенный балл}
     sameGrade[i] := 0;
```

```
readln (N);        {Ввели количество строк}
for i:=1 to N do        {Анализируем каждую входную
строку}
begin
 repeat
    read (ch)
 until ch=' ';    {Введена фамилия, нигде ее не
                  запоминаем, т.к. она нам не нужна}
 repeat
    read (ch)
 until ch=' ';    {Введено имя, нигде его
               не запоминаем, т.к. оно нам не нужно}
 readln (schoolNumber, ball);     {Ввели номер школы
                                    и балл ученика}
 sameGrade [ball] := sameGrade [ball] +1
end;
studentRating :=0;
for i:=0 to 40 do {Подсчитываем количество учеников,
получивших двойки}
 studentRating := studentRating + sameGrade [i];
studentNumber := (N - studentRating) * 30 div 100;
 {Количество учеников, которые должны получить трой-
                                  ки, должно быть 30%}
studentRating := 0;     {Подсчитаем, сколько учеников
                        на самом деле получат двойки}
i := 40;              {Меньше 40 баллов - это двойка}
while (studentRating < studentNumber) and (i < 60)
do     {Больше 60 баллов - это уже четверка}
begin
 i := i +1;
 studentRating := studentRating + sameGrade[i]
end;
if studentRating = studentNumber then  {Ровно 30%
учеников получат тройки, «i» - балл, за который ста-
вят тройку}
 writeln (i, ' ', studentRating)
else
 if (i=60) and (studentRating < studentNumber) then
       {Тройку получат меньше 30% учеников, т.к.
        много учеников получило 60 и более баллов}
    writeln ('60 ', studentRating)
```

```
      else
         if studentRating = sameGrade [i] then
           writeln (i, ' ', studentRating)
         else
         begin
           studentRating := studentRating - sameGrade
           [i];
           i := i -1;
           while sameGrade [i] = 0 do
                 i := i -1;
           writeln (i, ' ', studentRating)
         end
end.
```

31

```
var
     symbol,              {Очередной символ входной строки}
     maxMeeting: char; {Наиболее часто встречаемый символ}
     symbolFrequency:  array ['A'..'Z'] of integer;
     {Массив всех букв алфавита, значением элементов бу-
     дет число появлений каждой буквы во входной строке}
     k: integer; {Количество букв, появляющихся
                  максимальное число раз}
begin
     for symbol:='A' to 'Z' do    {Обнуляем массив, т.е.
        считаем, что пока каждая буква встречается 0 раз}
      symbolFrequency [symbol] := 0;
     read (symbol);    {Ввели первый символ входной строки}
     while symbol <> '#' do {Анализируем каждый символ до
                             тех пор, пока не встретим «#»}
     begin
     if (upcase (symbol) >= 'A') and (upcase (symbol) <= 'Z')
     then   {Большие и маленькие буквы различать
            не требуется, поэтому все символы преобразуем
            в большие буквы}
      inc (symbolFrequency [upcase (symbol)]);
            {Увеличиваем счетчик появления каждой буквы}
     read (symbol)      {Ввели очередной символ входной
                        строки}
     end;
     k := 1;
     maxMeeting := 'A';
```

```
    for symbol := 'B' to 'Z' do {Ищем максимальное
          число вхождений (maxMeeting) и количество
          таких максимальных вхождений (k)}
     if symbolFrequency [symbol] > symbolFrequency
     [maxMeeting] then     {Нашли более часто встретив
                 шийся символ, чем все символы, который
                 просмотрели до этого}
        begin
          maxMeeting := symbol; {Запомнили его}
          k:=1  {Пока такая частота появления у одного
                                          символа}
        end
        else
          if symbolFrequency [symbol] = symbolFrequency
          [maxMeeting] then     {Такое количество раз
          уже встречались другие символы, т.е. надо
          увеличить счетчик букв, которые появляются
          максимальное число раз}
            k:= k +1;
    if k = 1 then     {Чаще всего встречается одна буква,
          которая является значением переменной maxMeeting}
     writeln (maxMeeting)
    else  {Максимальная частота появления у нескольких
    букв, чтобы их найти, просмотрим весь массив от на-
    чала до конца. При этом автоматически буквы будут
    печататься в алфавитной порядке}
     for symbol := 'A' to 'Z' do
        if symbolFrequency [symbol] =
        symbolFrequency [maxMeeting] then
          write (symbol, ' ');  {Между найденными сим-
          волами будет выводиться пробел}
    writeln (symbolFrequency [maxMeeting])
                 {Максимальная частота появления букв,
                 которые были выведены до этого}
end.
```