АБИТУРИЕНТ

Н.А. Чупин

ПОДГОТОВКА К ЕГЭ ПО ИНФОРМАТИКЕ

ОПТИМАЛЬНЫЕ СПОСОБЫ ВЫПОЛНЕНИЯ ЗАДАНИЙ

ВЕРСИЯ 2013

Серия
«Абитуриент»

Н. А. Чупин

ПОДГОТОВКА К ЕГЭ ПО ИНФОРМАТИКЕ

ОПТИМАЛЬНЫЕ СПОСОБЫ ВЫПОЛНЕНИЯ ЗАДАНИЙ

Учебное пособие

ВЕРСИЯ 2013

Ростов-на-Дону
«ФЕНИКС»
2013

УДК 373.167.1:004
ББК 32.81я72
КТК 448
Ч-92

Чупин Н. А.

Ч-92 Подготовка к ЕГЭ по информатике : оптимальные способы выполнения заданий / Н. А. Чупин. — Ростов н/Д : Феникс, 2013. — 105, [1] с. : ил. — (Абитуриент).

ISBN 978-5-222-20691-1

Обсуждаются оптимальные способы решения задач ЕГЭ по информатике на примере демонстрационного варианта ЕГЭ 2013 года.

Для учащихся при подготовке к ЕГЭ по информатике и учителей информатики.

ISBN 978-5-222-20691-1

УДК 373.167.1:004
ББК 32.81я72

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Эта книга — для тех, кто не просто желает сдать единый государственный экзамен по информатике на какую-нибудь оценку, а стремится добиться высокого, близкого к максимальному результата. Это достойная и трудная цель, но добиться ее можно, если подойти к делу основательно.

Работая в вузе на курсах подготовки к ЕГЭ по информатике, а также занимаясь индивидуально с учащимися 11-х классов, я подготавливаю их к действиям в условиях острой нехватки времени. Многие недооценивают фактор времени при выполнении заданий ЕГЭ. Я же считаю, что большинство учащихся, будь у них неограниченное время, успешно справились бы со всеми заданиями уровней сложности Б (базового) и П (повышенного), и даже с отдельными заданиями уровня В (высокого). Но острая нехватка времени на ЕГЭ приводит к тому, что школьник, специально не готовившийся к ЕГЭ, показывает весьма посредственный результат.

Выделенное время — четыре полных часа — кажется очень большим. Однако, если представить его расклад по отдельным заданиями, окажется, что выполнять их надо в чрезвычайно быстром темпе. В самом деле, по плану организаторов на едином государственном экзамене по информатике для выполнения заданий частей А и В отводится 90 минут, и на выполнение заданий части С — 145 минут. В частях А и В находится в общей сложности 28 заданий. Это означает, что ученик должен делать одно задание в среднем за 3 минуты! Учитывая, что имеются

задания, где на поиск решения явно потребуется больше времени, ограничения на время выполнения большинства других заданий еще больше ужесточаются. Решать задачи в таком темпе учащимся на обычных школьных контрольных работах никогда не приходится. С такими ситуациями встречаются лишь участники олимпиад! Острый недостаток времени и важнейшее значение результата, ставящие выпускников в ситуацию сильнейшего стресса, мешают многим показать свои знания по максимуму.

Поэтому я стараюсь научить не просто правильным способам решений, а таким способам решений, которые максимально экономят время. Умение планировать время и создавать его запас за счет опережения графика возвращают учащемуся уверенность в себе. Отсюда — уменьшение количества ошибок, возможность перепроверки отдельных заданий, и в итоге — более высокие результаты.

Замечу, что часто в пособиях по подготовке к ЕГЭ приводятся универсальные способы решений, пригодные для всех случаев, без учета особенностей ЕГЭ. Например, для задания, где дана часть таблицы истинности логической формы F (это задания, подобные заданию А3 из демо-версии ЕГЭ по информатике 2013), в некоторых пособиях приводится громоздкий механизм построения таких форм практически из вузовского учебника по математической логике. Использовать такой способ на ЕГЭ — это все равно, что палить из пушки по воробьям. Предложенные четыре варианта ответов можно подтвердить или опровергнуть простым перебором.

Для многих задач можно привести упрощенные способы решений в виде схем. Следует учитывать, что современное поколение склоняется к образному мышлению. Это следствие широкого распространения телевидения, а затем и мультимедийных компьютеров. Поэтому нынешние школьники предпочитают текстам схемы, рисунки, табли-

цы, диаграммы и другие образные способы представления решения задачи.

Один из важных способов экономии времени — записывать решение максимально кратко. Ведь на ЕГЭ от учащегося не требуется предъявить само решение (кроме части С), как это делается, например, в школе на контрольных работах по математике. Поэтому он может использовать сокращения, значки, схемы и другие подобные способы, если они ускоряют процесс решения задачи. Получается как бы конспект ее решения. Подобным образом работают стенографистки (кажется, сейчас это искусство утрачивается в связи с появлением соответствующих технических средств).

С другой стороны, процесс решения следует конспектировать так, чтобы при необходимости его проверки можно было восстановить весь ход решения. Например, школьник, пытающийся на рабочем поле робота нарисовать все его траектории движения (да еще гелевой авторучкой!), получит картину, в которой даже не видно исходных заданных перегородок.

Я советую всем учащимся брать с собой на ЕГЭ карандаш и ластик, и многие такие конспекты выполнять прямо на листах с заданиями рядом с данными.

Итак, мой рецепт выглядит так: **кратко рассуждать, кратко записывать, быстро проверять, контролировать время.** Это я и называю оптимальными способами выполнения заданий.

В данном пособии изложены как традиционные способы решения задач и оформления решений, которые даются в школьных учебниках информатики и ориентированы на использование в условиях классно-урочной системы, так и специальные способы решений и конспектирования решений, позволяющие повысить результаты в условиях ЕГЭ.

Во многих случаях я излагаю решения, может быть, слишком подробно, то есть рассказываю то, что хорошему учащемуся и так ясно. Это означает, что я становлюсь в позицию учителя информатики, который должен излагать решение так, чтобы оно было понятно любому учащемуся. Текст в этом случае становится методическими рекомендациями для преподавания соответствующих тем информатики. Впрочем, и такой подробный текст может быть полезен школьнику — с его помощью можно ликвидировать пробелы в знаниях, кроме того, совершенствовать умение правильно рассуждать. Это вполне пригодится, например, при учебе в вузе.

Ежегодно разработчик заданий ЕГЭ — Федеральный институт педагогических измерений (ФИПИ) производит некоторую модернизацию структуры заданий, в целом сохраняя, конечно, преемственность с предыдущим годом. Некоторые изменения носят косметический характер — уточняются тексты заданий, устраняются неясности и двусмысленности. Сюда же можно отнести и такие ситуации, когда текстовая оболочка меняется, но суть задания и, соответственно, путь решения, не меняются. Иногда изменения носят более радикальный характер. Очевидно, когда процент успешности выполнения некоторого задания превышает заданную величину, разработчики либо серьезно усложняют задание, либо заменяют его на другое.

Представление о структуре заданий 2013 года можно получить, анализируя демо-вариант, размещенный на сайте ФИПИ www.fipi.ru. В данной книге я рассматриваю эти задания и предлагаю их эффективные решения с точки зрения расхода времени, страховки от ошибок, возможности проверки.

Замечу, что после существенных изменений в структуре ЕГЭ по информатике 2012 года в сравнении с 2011 годом, ЕГЭ 2013 года, как показывает демонстрационный

вариант, изменился незначительно. А вот ЕГЭ-2012 по информатике достаточно сильно отличался от предыдущих лет по структуре заданий. Прежде всего, резко изменилось соотношение количества заданий типа А и В с 18–10 до 13–15. Отчасти это вызвано необходимостью реагировать на критику. Те, кто критикует систему ЕГЭ, очень часто упоминают, что задания части А провоцируют учащихся на угадывание ответов вместо «правильного решения» задачи, как требует школа. Такая критика поверхностная, но на поверхностную критику ФИПИ реагирует тоже поверхностно — уменьшает количество таких заданий, увеличивая за их счет число заданий в части В, иногда простой переформулировкой задания. Как известно, в заданиях части В не приводятся варианты ответов, а дается только общая форма: ввести число или ввести текст. Кроме того, думается, что увеличение в 2012 году числа заданий типа В вызвано введением заданий, которые не очень подходят под тип А. При этом было заметно, что разработчики использовали совершенно новые по сути и при этом более сложные задания.

Между прочим, это означает, что книги и пособия предыдущих лет не вполне подходят для подготовки к ЕГЭ в 2013 году. Я не говорю, что они совсем бесполезны. Всё-таки некоторые задания сохранились неизменными. Во-вторых, они в целом рассматривают те же разделы школьной информатики. Наконец, в-третьих, разработчики иногда возвращают некоторые задания из прошлых лет, исходя из опыта текущего года, замечаний от квалифицированных критиков (учителей информатики, методистов, преподавателей вузов, экспертов центров обработки и др.). Как раз в варианте 2013 года вновь появились после годичного отсутствия некоторые задания предыдущих лет.

Тем не менее, при подготовке к ЕГЭ крайне важно познакомиться именно с демо-вариантом 2013 года, который в максимальной степени соответствует тому, что будет на ЕГЭ.

Если же сравнивать демо-варианты 2012 года и 2013 года, отметим, что «1. Одно задание с кратким ответом по теме «Кодирование текстовой информации» заменено на задание по теме «Рекурсивные алгоритмы» раздела «Элементы теории алгоритмов». 2. Немного изменена последовательность заданий во второй части работы.» [это цитата из замечаний разработчиков ФИПИ].

На мой взгляд, сложность заданий, если судить по демо-варианту 2013 года в сравнении с демо-вариантом 2012 года, в целом даже чуть-чуть снизилась.

В общем, есть возможность побороться за высокий балл!

Обозначения

В тексте используются следующие обозначения.

? Этот знак обозначает текст задания.

√ Этот знак обозначает пояснения, замечания, подробности определений, особые случаи и другие подобные тексты.

▶ **Эти два знака** обозначают начало и конец ◀ решения.

Для удобства обсуждения заданий с учителем или другими учащимися, я дал каждому заданию условное название, которое позволит легче ориентироваться в списке заданий.

РЕШЕНИЕ ЗАДАЧ ЧАСТИ А

А1 Сколько единиц в двоичной записи числа?

? Сколько единиц в двоичной записи числа 255?

1) 1 2) 2 3) 7 4) 8

√ Задачу можно переформулировать так: требуется преобразовать заданное число **255** из десятичной системы счисления в двоичную и подсчитать, сколько двоичных единиц будет в полученной двоичной записи числа.

Конечно, различные системы счисления — это просто разные способы записи одного и того же числа. Сначала поговорим о способах преобразования чисел из десятичной в двоичную систему счисления. Это можно сделать несколькими способами.

Способ первый

Последовательно делим заданное в десятичной системе счисления число на **2**, записывая отдельно результат деления и остаток. Процесс заканчивается, когда в результате деления получим **0**.

Для примера возьмем не то число, что дано в задании, а какое-нибудь другое, скажем, **985**.

Еще раз подчеркнем, деление заканчивается, когда в левом столбце получим нуль (многие ошибочно останавливаются, когда в левом столбце появилась единица).

Теперь остатки при делениях (правый столбец) записываем в *обратном* порядке — начиная с последнего, то есть с нижнего, и двигаясь вверх, тем самым получим двоичную запись того же числа. В этом примере:

$$985_{10}=1111011001_2.$$

Показанный способ плох тогда, когда учащиеся медленно и с ошибками делят числа на **2** в уме. В этих ситуациях я шучу: «Мысленно добавляйте к числу слово "рублей" и ошибаться будете реже!». Шутка шуткой, но совет помогает!

Таблица 1

Получение двоичной записи десятичного числа

Результат деления	Остаток
985	
492	1
246	0
123	0
61	1
30	1
15	0
7	1
3	1
1	1
0	1

Способ второй

Такой способ хорош для небольших чисел, особенно если вы хорошо суммируете и вычитаете в уме.

Необходимо также иметь сведения о степенях двойки.

Показатель степени	9	8	7	6	5	4	3	2	1	0
Значение	512	256	128	64	32	16	8	4	2	1

Я рекомендую для успешного применения этого способа выучить наизусть степени числа **2**, например, до 12-й (еще лучше до 16-й степени). Это пригодится и в других заданиях.

Будем последовательно вычитать из числа степень двойки, максимальную, какую только возможно. В результате получим разложение числа по степеням двойки.

$$985_{10} = 512+473 = 512+256+217 =$$
$$= 512+256+128+89 = 512+256+128+64+25 =$$
$$= 512+256+128+64+16+9 =$$
$$= 512+256+128+64+16+8+1$$

Напомним, что **1** — это тоже степень двойки, только нулевая: $1=2^0$.

Заполним теперь таблицу.

Степени двойки	**512**	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Имеется ли в разложении числа	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**

В этой таблице единицы ставим у тех степеней двойки, которые присутствуют в полученном разложении, соответственно нули — у тех степеней, которых нет.

Вторая строка и есть запись числа в двоичной форме!

Иногда я применяю этот способ «наоборот» — для проверки правильности двоичной записи, полученной первым способом. Беру те степени двойки, которые соответствуют разрядам, где находятся единицы в полученной двоичной записи числа, суммирую такие степени — в итоге должно получиться исходное десятичное число. Не совпало —

значит, где-то ошибка в вычислениях. То есть примерно так:

$$1111011001_2 = 1*512+1*256+1*128+1*64+$$
$$+0*32+1*16+1*8+0*4+0*2+1*1 =$$
$$= 512+256+128+64+16+8+1 = 985_{10}.$$

Способ третий

Вообще-то, способ имеет ограниченное применение — хорош в тех случаях, когда числа «почти круглые», то есть близки к степеням двойки — чуть больше или чуть меньше. Но у нас как раз такой случай, и, похоже, в этом задании разработчики во всех вариантах так и сделают. Для случаев «почти круглых» чисел следует вспомнить следующее:

$2^N = 10...0_2$ (N нулей после единиц).

И еще:

$2^N - 1 = 1...1_2$ (N единиц).

Например, **512** — это **2** в **9**-ой степени, поэтому

$512_{10} = 10\ 0000\ 0000_2$, а $511_{10} = 1\ 1111\ 1111_2$.

Обратите внимание: если в денежных суммах мы делаем разбивку на группы по 3 цифры, то в информатике при записи двоичных чисел удобнее группировать по четыре цифры. Первую из двух формул удобно применять в случаях, когда заданное число чуть-чуть больше некоторой степени двойки, а вторую формулу следует применять, когда заданное число чуть меньше степени двойки. Выигрыш в том, что мы избегаем при двоичных вычислениях переносов и заёмов, которые могут привести в спешке к ошибкам. Однако для успешного — быстрого и правильного — использования данного способа требуется обязательно помнить степени двойки.

Приведем несколько примеров использования этого способа.

Пример 1. Получить двоичную запись числа 1025.

$$\begin{array}{r} +\ 1024_{10}=100\ 0000\ 0000_2 \\ 1 \\ \hline 1025_{10}=100\ 0000\ 0000_2 \end{array}.$$

Для скорости можно записать в черновике и так:

$$\begin{array}{l} 102\cancel{4}_{10}=100\ 0000\ 000\cancel{0}_2. \\ \quad\ \ 5 \qquad\qquad\qquad\qquad\ \ 1 \end{array}$$

Пример 2. Получить двоичную запись числа 1020.

$$\begin{array}{r} -\ \begin{array}{l} 1023_{10}=11\ 1111\ 1111_2 \\ \quad 3_{10} \qquad\qquad\qquad\ \ 11_2 \end{array} \\ \hline 1020_{10}=11\ 1111\ 1100_2 \end{array}.$$

√ В задании А1 в предыдущие годы как раз давались только числа, близкие к степеням двойки (чуть больше или чуть меньше). Поэтому рекомендуем использовать третий способ как самый быстрый и безопасный от ошибок. Однако первые два держать в запасе — вдруг разработчики в этом году решат усложнить задачу!

√ **Важно** за всеми разговорами о способах преобразования числа в двоичную систему счисления **не забыть** о том, что все-таки требовалось в задании. Ведь **нам нужно** не само двоичное число, а **количество единиц** в нем.

▶ Теперь приступим непосредственно к решению задания. Я предлагаю такой конспект решения:

$$255 = 2^8 - 1 = 1111\ 1111_2.$$

Обратите внимание на разбивку по четыре цифры — так легче их сосчитать!

Итак, ответ — **восемь** единиц. Дополнительных сложений или вычитаний в данном случае не потребовалось. В бланке ответов следует отметить **вариант ответа номер 4.** На бланке ответов это будет выглядеть так:

A1

1 ☐

2 ☐

3 ☐

4 ☒

Решение завершено. ◀

√ Подобная задача присутствовала еще в 2011 году, но требовалось определить количество *значащих нулей.* У многих участников это вызвало затруднения только из-за того, что они не знали, что такое *значащие нули* и *незначащие нули.* На всякий случай поясним эти понятия.

В компьютерном мире (по крайней мере, сегодня) минимальной адресуемой порцией информации является байт, состоящий из 8 битов. Поэтому двоичные числа хранятся в памяти компьютера блоками по 8, 16, 32 или 64 бит, то есть по одному, два, четыре или восемь байт. Возьмем минимальный вариант — восьмибитовое число, то есть такое, для которого выделен один байт.

Пример. Десятичное число 9 запишется в двоичной системе по битам выделенного для хранения этого числа байта так: 00001001. Первые четыре нуля называются не-

значащими, а последние два — значащими. При изучении темы «Системы счисления» с точки зрения алгебры мы не записываем незначащие нули и пишем просто 1001, то есть поступаем так же, как мы обычно действуем с десятичными числами. Однако в компьютере все равно все эти восемь битов выделены для хранения числа, поэтому должны быть заполнены, и они заполняются нулями, которые никак не влияют на величину числа и называются незначащими.

Еще пример. Десятичное число 72 при размещении в восьмибитовой ячейке памяти запишется как 01001000 и будет иметь один незначащий и пять значащих нулей.

Добавим, что количество незначащих нулей зависит от размера выделенной ячейки памяти, например, если для того же числа 72 выделена ячейка памяти размером 16 бит, это число запишется как 0000 0000 0100 1000 и будет иметь 9 незначащих нулей. Но количество значащих нулей будет по-прежнему равно пяти.

Эти пояснения я привожу на всякий случай — вдруг разработчики тестов будут использовать и такой вариант условия. В задании же, которое приведено в демо-варианте 2013 года, говорится о количестве единиц, и такой вопрос звучит для школьников намного яснее.

A2 | *Расстояние между населенными пунктами*

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

	A	B	C	D	E	F
A		3				
B	3		7	4	7	
C		7			5	
D		4			2	
E		7	5	2		3
F					3	

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

1) 11 2) 12 3) 13 4) 18

√ В задании подразумевается «двунаправленность» дорог — расстояния от X к Y и от Y к X везде одинаковы. Поэтому таблица симметрична относительно главной диагонали. В этой ситуации для ускорения времени можно сразу «забыть» про нижнюю часть и рассматривать только то, что выше диагонали.

▶

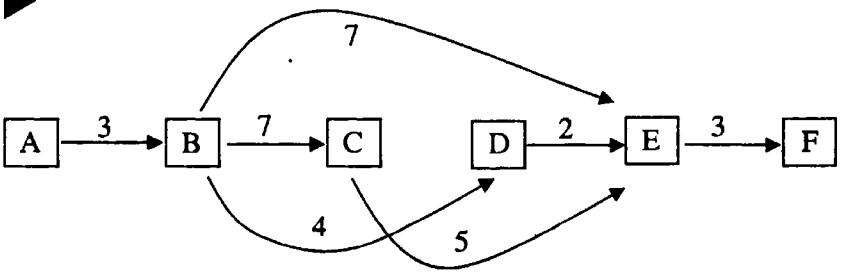


Рис. 1. Графическое изображение табличных данных

Для проверки правильности перевода таблицы в схему полезно сравнить количество стрелок из каждого пункта и количество чисел правее диагонали в соответствующей строке. Например, из пункта В выходит три стрелки и в строке В правее диагонали три числа 7,4,7 — эти числа и записаны на стрелках.

В принципе уже этого рисунка достаточно, чтобы увидеть кратчайший путь ABDEF. Расстояние по нему равно 12.

Правильный вариант ответа 2). ◀

√ Если схема дорог будет не так очевидна (я в это не верю; учитывая характеристику этого задания в спецификаторе ЕГЭ, оно не может быть очень сложным), более мощный метод можно освоить, изучив решение задания **B9**. Общая его идея — начиная от узла А, последовательно вычислять для каждого узла длину кратчайшего пути из А до этого узла. Таким образом, вычисленное для узла F число и будет условием задачи. Надо, однако, понимать и отличие задания **B9** от рассматриваемого нами сейчас задания **A2**. Там требуется найти *количество* возможных различных путей из начального пункта в конечный, здесь же — *длину* кратчайшего пути.

Мы проиллюстрируем такой более мощный метод на примере *других данных* — задание **A2** из демо-варианта 2013 дает мало места для применения более мощного метода. Таблицу данных записывать не будем, приведем только графическую схему (рис. 2).

Последовательность действий при решении задачи.

1. Для пункта А назначим условное значение 0 (расстояние от А до А) . Это число в кружочке.

2. Для пункта В к предыдущему числу прибавим 3 — расстояние от А до В. Оно же и является кратчайшим расстоянием от начального пункта до пункта В.

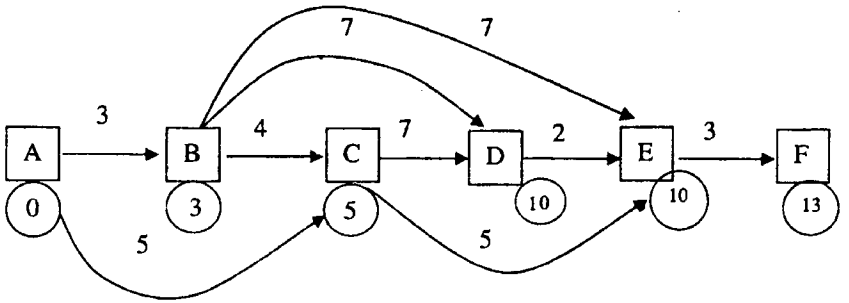


Рис. 2. Графическое изображение табличных данных

3. В пункт С мы можем прибыть двумя способами: либо из пункта А до пункта С, либо из пункта В до пункта С. Для каждой из этих двух возможностей нужно вычислить сумму кратчайшего расстояния до пункта Р (в данном случае это может быть либо А, либо В) и длину пути РС. Из полученных сумм находим наименьшую min(3+4, 0+5)=5 — она будет кратчайшим расстоянием от А до С. Записываем ее в кружочке у данного пункта.

4. Аналогично для пункта D вычисляем:

min(3+7, 5+7)=10.

5. Для пункта Е вычисляем:

min(3+7, 5+5, 10+2)=10.

6. Для пункта F минимум выбирать не приходится — дорога в F только одна, поэтому просто к длине кратчайшего пути от А до Е прибавим расстояние EF и получим ответ данной задачи: 10+3=13.

Напоминаю, что мы сейчас решили некоторую условную задачу, чуть более сложную, чем в демо-варианте, с целью продемонстрировать более мощный метод. Задача из демо-варианта уже решена выше.

А3 Таблица истинности выражения

? Дан фрагмент таблицы истинности выражения *F*:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	F
1	1	0	1	1	1	1	0
1	0	1	0	1	1	0	0
0	1	0	1	1	0	0	1

Каким выражением может быть *F*?

1) $\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4 \wedge x_5 \wedge \neg x_6 \wedge \neg x_7$

2) $\neg x_1 \vee x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5 \vee \neg x_6 \vee x_7$

3) $x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4 \wedge x_5 \wedge x_6 \wedge \neg x_7$

4) $x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4 \vee \neg x_5 \vee x_6 \vee \neg x_7$

√

Таблица истинности конъюнкции

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

Таблица истинности дизъюнкции

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

«Бесхитростный» способ — последовательно проверять каждый вариант ответа вычислением значения F для каждой строки и сравнением с заданным в таблице. Наверное, так бы и пришлось поступать, если бы формулы были чуть сложнее и в них в строке были бы перемешаны и конъюнкции, и дизъюнкции.

Применим чуть-чуть хитрости, чтобы найти решение быстрее. В каждой формуле в вариантах ответов присутствуют либо только конъюнкции, либо только дизъюнкции. Но *дизъюнкции дают 0 только в единственной строке (это не зависит от количества переменных)*, у нас же в таблице в условии *F принимает значение 0 в двух строках*. Это позволяет сразу *отбросить* вариант ответа 2) и вариант ответа 4). В них дизъюнкции!

Из оставшихся вариантов 1) и 3) уже легче выбрать правильный. Обратим внимание на ту единственную строку в таблице, которая дает F, равное 1 (обычно как раз первым делом рассматривают «особенную», не похожую на других строку). Чтобы это случилось, необходимо, чтобы каждый из «кусочков», объединенных знаками конъюнкции (то есть переменная со знаком отрицания или без него) давал единицу. Но, взяв первую же переменную x_1, сразу видим, что вариант 3) не подходит. В нем x_1 стоит без знака отрицания, и при $x_1=0$, как дано в третьей строке таблицы, уже никак не может получиться

$$F= 0 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4 \wedge x_5 \wedge x_6 \wedge \neg x_7 = 1.$$

Правильный вариант ответа 1).

Дополнительная проверка

Проверим на всякий случай найденный вариант ответа 1 для двух других строк таблицы.

В первой строке таблицы

$x_1=1$ $x_2=1$ $x_3=0$ $x_4=1$ $x_5=1$ $x_6=1$ $x_7=1$

$F = \neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4 \wedge x_5 \wedge \neg x_6 \wedge \neg x_7$
$= \neg 1 \wedge 1 \wedge \neg 0 \wedge 1 \wedge 1 \wedge \neg 1 \wedge \neg 1 =$
$= 0 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 0 = 0.$

В таблице в первой строке именно такое F.

Во второй строке таблицы

$x_1=1 \quad x_2=0 \quad x_3=1 \quad x_4=0 \quad x_5=1 \quad x_6=1 \quad x_7=0$

$F = \neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4 \wedge x_5 \wedge \neg x_6 \wedge \neg x_7 =$
$= \neg 1 \wedge 0 \wedge \neg 1 \wedge 0 \wedge 1 \wedge \neg 1 \wedge \neg 0 =$
$= 0 \wedge 0 \wedge 0 \wedge 0 \wedge 1 \wedge 0 \wedge 1 = 0.$

И здесь тоже совпадение!

Эти две проверки убеждают нас, что мы не ошиблись в подсчетах. ◀

√ Конъюнкция называется еще логическим умножением, поэтому «кусочки» правильнее называть сомножителями, но нам важно быстрее решить задачу, а не отточить правильные математические формулировки. Последнее — одна из привилегий вузовского курса математической логики, куда нам (ну то есть вам) еще нужно поступить!

А4 | *Задача про маски файлов*

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

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

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

В каталоге находятся 6 файлов:

asc.wma
casting.wmv
last.wma
pasta.wmvx
pasta.wri
vast.wma

Определите, по какой из масок из этих 6 файлов будет отобрана указанная группа файлов:

casting.wmv
last.wma
pasta.wmvx
vast.wma

1) ?as*.wm?
2) *as?.wm*
3) ?as*.wm*
4) ?as*.w*

√ Так сложилось в компьютерном мире: имя файла состоит из основной части имени файла и расширения имени файла — они разделены точкой. Иногда «**основная части имени файла . расширение имени файла**» называют ошибочно полным именем файла. На самом деле «полное имя файла» — это когда к этому слева добавлен еще и путь к файлу, включающий имя диска и последовательность вложенных каталогов, приводя-

щая нас к файлу. Для краткости часто допускают неточность и говорят

«имя файла . расширение»

вместо

«основная части имени файла . расширение имени файла».

В некогда популярной операционной системе *MS DOS* для имен файлов действовало так называемое ограничения «8.3», по которому имя файла должно быть не более чем из восьми символов, а расширение — не более чем из трех символов. Разрешалось использовать только символ подчеркивания, цифры и буквы латиницы (в последних версиях разрешались и буквы кириллицы). В задании же имеются расширения из 4-х символов, поэтому речь идет о так называемых длинных именах файлов, впервые появившихся в операционной системе *Windows*. Причем в *Windows* в основной части имени файла и в расширении имени разрешалось использовать и точку. Это привело к некоторой двусмысленности, например, в имени файла exp.mmm.t.x.t трудно определить, какая из точек разделяет основную часть имени файла и в расширение имени. Это затруднительно для пользователя, но не для файловой системы, которая просто хранит эти две части имени файла «в разных местах».

В задании все-таки неявно присутствует предположение: точка не может встретиться в основной части имени или в расширении, и может служить только для разделения этих частей.

Маска подобна фильтру, который **некоторые имена файлов пропускает**, то есть разрешает им двигаться дальше и отображаться в ответах, **а некоторые имена** (не соответствующие его требованиям) через него **пройти не могут**.

В простых случаях смысл шаблонных символов **?** и * легко понятен, и затруднения возникают только от того,

что пользователи никогда ими не пользовались. В связи с этим хочу сказать несколько слов о «чайниках». В компьютерном сленге «чайником» называют самоучку, который до многого дошел «своим умом». «Чайник» — это не совсем плохая характеристика пользователю. По крайней мере, он научился решать с помощью компьютера свои задачи! Только делает это не самым эффективным способом. Большинство школьников сейчас многие часы проводят за компьютером, но они по-прежнему остаются «чайниками». Одна из задач школьной информатики — сделать из «чайников» более-менее квалифицированных пользователей!

▶ Для решения задачи можно применить несколько стратегий.

1. Последовательно проверять каждый из четырёх вариантов ответов на наборе исходных файлов.

2. Заметим, что особую роль играют имена файлов asc.wma и pasta.wri, а именно: та маска, которую мы ищем, не пропускает из шести исходных файлов только эти два. Больше того, так как искомая маска пропускает при этом файл pasta.wmvx, делаем вывод, что причина кроется в расширении имени файла. Это соображение позволяет быстро забраковать вариант ответа номер 1 — он требует ровно 3 символа в расширении имени и никак бы не пропустил pasta.wmvx, и также забраковать вариант ответа номер 4 — для расширения файлов он требует w*, то есть обязательную первую букву w и дальше всё что угодно, но под такое ограничение одинаково годится и .wri, и wmvx. В оставшихся вариантах ответов 2) и 3) в расширениях одно и то же wm*. Поэтому имеет смысл рассматривать только маски для основной части имени файла и при этом особое внимание обратить на тот самый файл asc.wma. Быстрее всего заметить, что в варианте ответа 3) маска ?as*.wm* требуется, чтобы перед as в начале стоял еще какой-то символ. Поэтому такая маска не пропустит имя

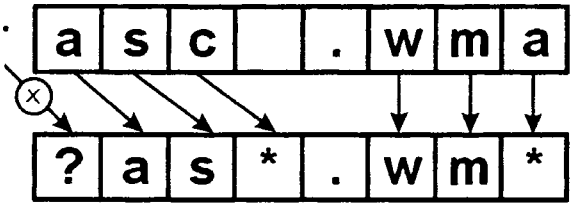
asc.wma. А вот маска в варианте 2) *as?.wm* такое имя пропустит, так как * может обозначать и «ничего».

Итак, мы нащупали вариант ответа 3). Проверим его на всех шести именах файлов.

Проверять условия соответствия имени файла маске следует отдельно для основной части имени и для расширения имени! Имя соответствует маске, если оно успешно прошло обе проверки!

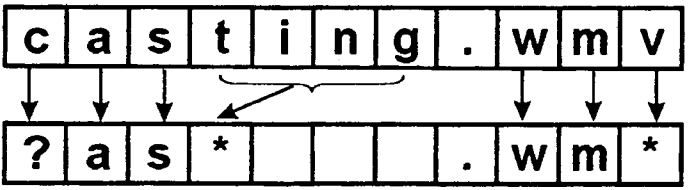
Файл 1. asc.wma

Поясним решение с помощью помещенной ниже схемы. В ней вверху написано имя файла, внизу маска. Стрелка показывает соответствие, а стрелка с крестиком показывает несоответствие между элементом имени файла или расширения и соответствующим элементом маски.



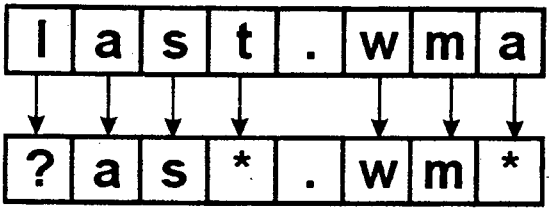
Вывод: файл asc.wma маске **не удовлетворяет**.

Файл 2. casting.wmv



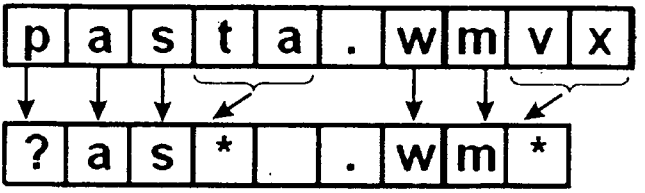
Вывод: файл casting.wmv маске удовлетворяет.

Файл 3. last.wma



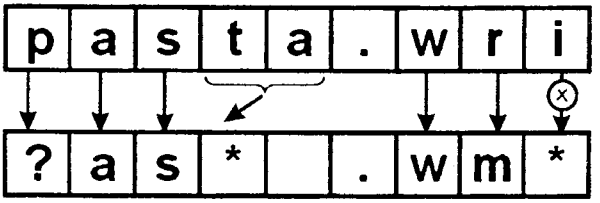
Вывод: файл last.wma маске удовлетворяет.

Файл 4. pasta.wmvx



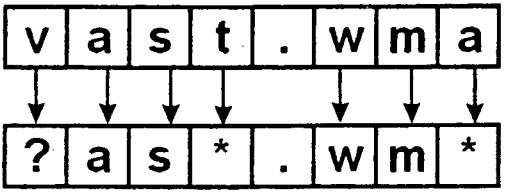
Вывод: файл pasta.wmvx маске удовлетворяет.

Файл 5. pasta.wri



Вывод: файл pasta.wri маске **не удовлетворяет**.

Файл 6. vast.wma



Вывод: файл vast.wma маске удовлетворяет. Заметим, между прочим, что этот случай ничем не отличается от случая last.wma.

Итог. Маска ?as*.wm* отбирает из шести файлов:
asc.wma
casting.wmv
last.wma
pasta.wmvx
pasta.wri
vast.wma,

ровно четыре файла:
casting.wmv
last.wma
pasta.wmvx
vast.wma
как и задано в условии. Поэтому она и является правильным ответом.

Правильным ответов является вариант 3). ◀

A5 Автомат преобразования двухзначных шестнадцатеричных чисел

? Автомат получает на вход два двухзначных шестнадцатеричных числа. В этих числах все цифры не превосходят цифру 6 (если в числе есть цифра больше 6, автомат отказывается работать). По этим числам строится новое шестнадцатеричное число по следующим правилам.

1. Вычисляются два шестнадцатеричных числа – сумма старших разрядов полученных чисел и сумма младших разрядов этих чисел.

2. Полученные два шестнадцатеричных числа записываются друг за другом в порядке возрастания (без разделителей).

Пример. Исходные числа: 66, 43. Поразрядные суммы: A,9. Результат: 9A.

Определите, какое из следующих чисел может быть результатом работы автомата.

1) 9F 2) 911 3) 42 4) 7A

√ Прежде чем приступить к решению, отметим следующее.

Есть *два условия*, которые позволяют нам выбрать правильный вариант ответа и, соответственно, отвергнуть неправильные.

Одно из этих условий записано явно: **полученные числа записываются друг за другом *в порядке возрастания* (без разделителей).**

Другое условие «зашифровано» в условии. Если две цифры не превосходят **6**, то их сумма не может быть больше

шестнадцатеричной цифры **C**. Это касается и суммы старших разрядов, и суммы младших разрядов. Поясним для тех, кто еще не понял: цифра **6** имеется и в десятичных цифрах, и в шестнадцатеричных. В десятичной системе счисления сумма двух таких цифр даст **12**, и это будет двузначное число, но в шестнадцатеричной системе счисления это будет не двузначным, а однозначным числом **C** (в данном случае слово «однозначным» не очень подходящее — лучше сказать «одноразрядным числом» или «числом, составленным из одной цифры).

Конечно, сумма таких «небольших» шестнадцатеричных цифр не может быть двузначной. Тогда сумма двух *двузначных (то есть состоящих из двух цифр)* шестнадцатеричных чисел с таким ограничением как раз будет именно двузначной, то есть содержать ровно две цифры.

▶ Итак, ограничение «все цифры не превосходят цифру **6**» позволяет нам отбросить первый и второй варианты ответов.

Третий вариант не удовлетворяет условию: полученные суммы записываются в порядке возрастания.

Правильным вариантом является *вариант 4.*

Проверка.

Такая сумма получится, например, если на вход автомату подать шестнадцатеричные числа 46 и 54. ◀

А6 | *Получение информации из базы данных*

? Ниже приведены две таблицы из базы данных. Каждая строка таблицы 2 содержит информацию о ребенке и об одном из его родителей. Информация представлена

значением поля ID в соответствующей строке таблицы 1. В фрагменте базы данных представлены сведения о родственных отношениях. Определите на основании приведенных данных фамилию и инициалы внучки Петровой С.М.

Таблица 1		
ID	Фамилия_И.О.	Пол
25	Жвания К.Г.	Ж
49	Черняк А.П.	М
62	Петрова М.Н.	Ж
76	Ильченко Т.В.	Ж
82	Петрова С.М.	Ж
96	Басис В.В.	Ж
102	Ильченко В.И.	М
123	Павлыш Н.П.	Ж
134	Черняк П.Р.	М
...	...	...

Таблица 2	
ID_Родителя	ID_Ребёнка
25	134
76	49
76	123
82	76
82	96
102	76
102	96
134	49
134	123
...	...

1) Басис В.В.
2) Ильченко Т.В.
3) Павлыш Н.П.
4) Петрова М.Н.

√ В таблице 1 дана информация (точнее, часть информации) о нескольких лицах, в таблице 2 дана частичная информация об отношениях родитель–ребенок. У нас нет прямых данных об отношении (бабушка/дедушка)–внучка, но из житейских представлений мы понимаем, что (бабушка/дедушка) — это мать матери или мать отца внучки.

Формально говоря, (X — внучка для Y) означает, что (X имеет пол Ж) И (X — ребенок для Z) И (Z — ребенок для Y).

Искать придется в обратном порядке. В задании Y — это Петрова С.М. Придется искать всех ее детей Z, а затем внуков X по правилу «дети детей», и еще не забыть, что нам годится только внучка и не годится внук.

В заданиях используется сокращение ID (*identifier* — идентификатор), которое обычно используется в базах данных для так называемых первичных ключей — числовых кодов, однозначно характеризующих объект. Это прежде всего означает, что в таблице 1 нет строк с повторяющимися значениями поля ID. Использование первичных ключей позволяет забыть про фамилию, а говорить: нам нужна внучка человека с ID=82.

▶ Итак, в таблице 1 находим код (ID) Петровой С.М. Он равен 82. В таблице 2 ищем это значение в поле ID_Родителя. Это дает нам информацию, что у Петровой С.М. есть два ребенка, которые имеют коды ID=76 и ID=96. В этой задаче нам даже неважно, какого они пола. Важно только найти, кто у них является ребенком. Для этого эти коды 76 и 96 ищем теперь в поле ID_Родителя. Для кода 96 таких строк нет, а вот для кода 76 найдется две строки, в которых ID_ребенка равен соответственно 49 и 123. Но по таблице 1 выясняем, что 49 — это лицо мужского пола и внучкой быть не может.

А вот ID=123 нам вполне подходит. Итак, мы нашли цепочку: (Петрова С.М., ID=82 — родитель для ID=76, а ID=76 — родитель для ID=123 Павлыш Н.П., пол=Ж). В этом итоговом факте мы записали только ту часть информации из таблиц, которая нам нужна для ответа. Между прочим — ID=76 — это Ильченко Т.В., пол=Ж, как говорят в таких случаях Павлыш Н.П. — это внучка по матери, но для решения задания это неважно.

Можно конспектировать решение в виде схемы.

Таблица 1		
ID	Фамилия_И.О.	Пол
25	Жвания К.Г.	Ж
49	Черняк А.П.	М
62	Петрова М.Н.	Ж
76	Ильченко Т.В.	Ж
82	Петрова С.М.	Ж
96	Басис В.В.	Ж
102	Ильченко В.И.	М
123	Павльш Н.П.	Ж
134	Черняк П.Р.	М
...	...	...

Таблица 2	
ID_Родителя	ID_Ребёнка
25	134
76	49
76	123
82	76
82	96
102	76
102	96
134	49
134	123
нет в родителях...	

не

Итак, *правильный вариант ответа — 3).*

A7 Автоматическая корректировка формул при копировании в электронных таблицах

? Дан фрагмент электронной таблицы.

	A	B	C	D
1	1	2	3	
2	5	4	=$A$2+B$3	
3	6	7	=A3+B3	

Чему станет равным значение ячейки D1, если в нее скопировать формулу из ячейки C2?

Примечание: знак $ используется для обозначения абсолютной адресации.

1) 18
2) 12
3) 14
4) 17

√ Это вопрос очень простой для тех, кто часто работает с электронными таблицами. Однако эта замечательная программа, (вернее, класс программ), на мой взгляд, до сих пор недооценена. Очень немногие учащиеся используют эту программу, даже не подозревая, как ускоряет она решение многих задач, которые им приходится решать на разных уроках, и даже не на уроках, а в жизни!

Для тех же, кто не знаком с ними, следует пояснить следующий минимум сведений.

Электронная таблица состоит из клеток (другое название — ячейки). Каждая ячейка имеет координаты — буквенный номер столбца и числовой номер строки. В ячейках таблицы можно располагать числа, тексты и формулы. Формула всегда начинается со знака =, что и является признаком того, что это формула.

Приведем типичный пример. Он не связан с заданием из демо-варианта ЕГЭ, а просто иллюстрирует механизм корректировки формул.

	A	B	C	D
1	**Продукт**	**Цена**	**Количество**	**Сумма**
2	Хлеб	12	20	=B2*C2
3	Молоко	26,2	12	
4	Сыр	280	2,5	
5				

Формула в ячейках обычно не отображается, а вместо этого в ней отображается результат вычисления, но мы включили специальный режим, чтобы увидеть на изображении как раз формулы, а не результаты.

Было бы нерационально в каждую клетку столбца D вписывать новую формулу — ведь они аналогичны друг другу.

С другой стороны, копирование «буква в букву» привело бы к тому, что при копировании формулы вниз в ячейки D3, D4, ... и так далее во всех ячейках столбца D одинаково было бы =B2*C2, что нам не подходит.

Поэтому в электронных таблицах принят принцип автоматической корректировки формул при копировании.

Благодаря этому принципу при копировании формулы =B2*C2 в клетку D3, она будет иметь вид =B3*C3, для клетки =B4*C4 для клетки D4 и так далее. То есть электронная таблица «понимает», что каждый раз умножаются числа из соседних слева столбцов, но каждый раз из соответствующей строки. Итак, после копирования формулы будут такими.

	A	B	C	D
1	**Продукт**	**Цена**	**Количество**	**Сумма**
2	Хлеб	12	20	=B2*C2
3	Молоко	26,2	12	=B3*C3
4	Сыр	280	2,5	=B4*C4

Аналогично, если бы формула копировалась по горизонтали, менялись бы буквы — номера столбцов (Правда, в данном примере это бы не имело смысла!).

Однако иногда бывает, что, наоборот, корректировать какую-то из координат, участвующих в формуле, не следует.

Например, введем параметр — надбавка за доставку товаров, которая время от времени меняется. Тогда ее следует ввести в какую-то клетку и ссылаться на клетку при подсчетах.

	A	B	C	D	E
1	**Продукт**	**Цена**	**Количество**	**Сумма**	**Сумма с надбавкой**
2	Хлеб	12	20	=B2*C2	=D2*(1+C7)
3	Молоко	26,2	12	=B3*C3	
4	Сыр	280	2,5	=B4*C4	
5					
6					
7	Надбавка		0,2		
8					

Теперь при копировании формулы =D2*(1+C7) таблица будет автоматически изменять D2 на D3, D4, и это правильно, но таблица будет изменять и C7 на C8, C9,..., а эта корректировка нам не нужна, она приведет нас к неправильному результату.

Для таких случаев в электронных таблицах есть особый инструмент. Знак $ в формуле, поставленный перед координатой, числовой или буквенной, никак не влияет на результат вычисления, но «замораживает» при копировании ту координату, перед которой он поставлен. Поэтому правильный вариант формулы в примере: =D2*(1+C$7). При копировании она будет превращаться в =D3*(1+C$7), =D4*(1+C$7) и так далее, как и требуется в рассматриваемом нами постороннем примере.

	A	B	C	D	E
1	**Продукт**	**Цена**	**Количество**	**Сумма**	**Сумма с надбавкой**
2	Хлеб	12	20	=B2*C2	=D2*(1+C$7)
3	Молоко	26,2	12	=B3*C3	=D3*(1+C$7)
4	Сыр	280	2,5	=B4*C4	=D4*(1+C$7)
5					
6					
7	**Надбавка**		0,2		

Теперь можно приступать к решению задания. Все очень просто. Формула копируется из ячейки C24 в ячейку D1. Смещение по вертикали — на одну строку выше, смещение по горизонтали — на один столбец правее. Но в формуле **=$A$2+B$3**, которую мы собираемся копировать только координата B не имеет перед собой слева знака **$.** Поэтому в формуле она будет изменяться, а прочие координаты останутся при копировании без изменений. При копировании формулы на один столбец правее (это существенно) и на одну строку выше (это в данном конкретном случае никакой роли не играет) координата B превратится в координату C, а вся формула превратится в **=$A$2+C$3.** Теперь остались только расчеты.

$A$2+C$3 = 5+(A3+B3) = 5+13 = 18,

Правильный ответ — вариант ответа 1).

Можно предложить краткую схему решения.

	A	B	C	D
1	1	2	3	=$A$2+~~B~~$3 C
2	5	4	=$A$2+B$3	
3	6	7	=A3+B3	

A8 Расчет размера файла аудиозаписи

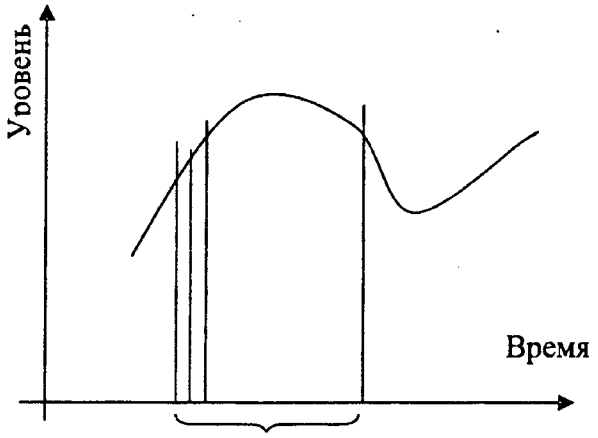
? Производится одноканальная (моно) звукозапись. Значение сигнала фиксируется 48 000 раз в секунду, для записи каждого значения используется 32 бит. Запись длится 4 минуты, её результаты записываются в файл, сжатие данных не производится. Какая из приведенных ниже величин наиболее близка к размеру полученного файла?

1) 44 Мбайт 2) 87 Мбайт 3) 125 Мбайт 4) 175 Мбайт

√ Грубо схему кодирования звука можно представить себе так: много раз в секунду (в задании дано — 48000 раз в секунду,) измеряется уровень звуковой волны. По условию, на каждое измерение тратится 32 бита, то есть при измерении уровня звуковой волны может получаться 232 разных значение.

Общая закономерность: чтобы получить звукозапись более высокого качества, следует увеличить частоту измерений и производить измерения с большей точностью, то есть тратить на каждое измерение больше бит, но и то и другое потребует больше памяти.

Примечание: следует быть внимательным! Если речь идет о стереозвукозаписи, то есть двухканальной, то потребуется памяти в два раза больше.



Частота измерений в секунду (Гц)

Этих минимальных сведений достаточно, чтобы приступить к вычислениям.

▶ На секунду звука потребуется 48000*32 бита, то есть 48000*32/8=192000 байт. Ну а на минуту — в 60 раз больше, и еще не забудем про то, что запись длилась 4 минуты.

Получаем 192000*60*4=46 080 000 байт. Поэтому выбираем вариант ответа 1).

Конечно, на самом деле потребуется чуть больше — в файле будет размещено еще некоторое количество служебной информации. Поэтому вопрос в задании сформулирован именно так: какая из величин наиболее близка к размеру полученного файла?

Заметим еще, что в таких задачах часто не требуется вычислять точно. Округлим 48000 до 50 000, и тогда вычислять будет легче: 50 000*32/8 = 200 000 байт и далее, 200 000 байт *60*4 = 48 000 000 байт. Точно так же 1Мбайт можно считать приближенно равным 1 000 000 байт. Российский академик Крылов (еще «той» России) говорил: «Недостаток математического образования проявляется прежде всего в избыточной точности производимых вычислений». Обратите также внимание на группировку по 3 цифры — это позволит избежать «глупых» ошибок. ◀

A9 | *Однозначное кодирование*

? Для кодирования некоторой последовательности, состоящей из букв А, Б, В, Г и Д, решили использовать неравномерный двоичный код, позволяющий однозначно декодировать двоичную последовательность. Вот этот код: А – 00, Б – 01, В – 100, Г – 101, Д – 110. Можно ли сократить для одной из букв длину кодового слова так, чтобы код по-прежнему можно было декодировать однозначно? Коды остальных букв меняться не должны.

Выберите правильный вариант ответа.

1) для буквы Д – 11
2) это невозможно

3) для буквы Г – 10
4) для буквы Д – 10

√ При равномерном кодировании, то есть когда все коды одинаковой длины, вопроса, где граница кода очередного символа, не возникает. Например, используй таблицу ASCII, и нужно будет лишь каждый раз отсчитывать 8 бит — вот тебе и очередной код символа. Однако, когда используются коды разной длины для разных букв, появляется возможность для двусмысленности. Например, если закодировать М как 000, а N как 00, то сообщение 000000 можно раскодировать двумя способами — либо как MM, либо как NNN.

Но если действовать аккуратно, то можно и при неравномерном кодировании обеспечивать однозначность декодирования.

Условие Фано: чтобы кодирование было однозначным, никакой код не должен быть начальной частью другого кода.

▶ Условие Фано позволяет быстро забраковать вариант 3) — код буквы Г становится начальной частью кода буквы В. Аналогично не подойдет и вариант 4).

А вот вариант 1) вполне годится, измененный код буквы Д не будет начальной частью кода других букв, и другие коды не будут начальной частью кода буквы Д или какой-то другой.

Из схемы видно, что при заданных кодах букв А, Б, В и Г для еще одной буквы Д код должен либо совпадать с 11, либо начинаться с 11, поскольку все другие ветки «закрыты» и попытка дописать к ним нижестоящий узел приведет к нарушению условия Фано.

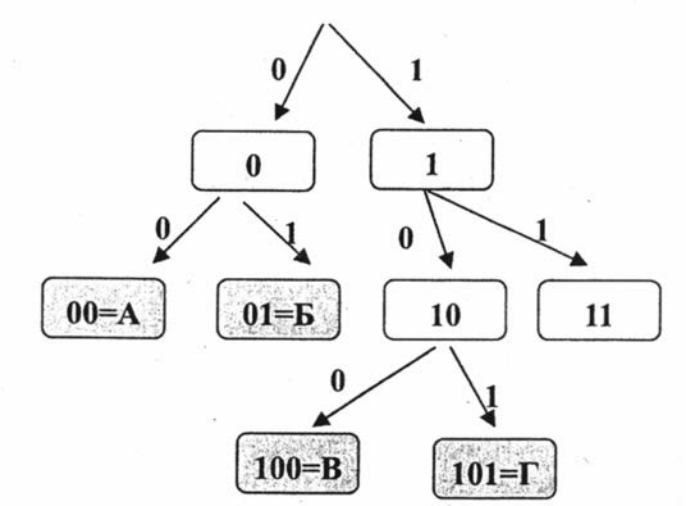


Рис. 3. Пример однозначного кодирования

Правильным вариантом из предложенных является вариант ответа 1). ◀

√ Заметим, что если именно код 11 использовать для буквы Д, не остается возможностей для кодирования других букв, и в сообщениях можно будет использовать только эти пять букв.

A10 *Импликации в условиях*

? Для какого из приведенных чисел X истинно логическое условие: $\neg((X \text{ кратно } 2) \rightarrow (X \text{ кратно } 4))$?

1) 7 2) 8 3) 10 4) 12

√ Опыт показывает, что даже неплохие учащиеся часто путаются в обозначениях, названиях и правилах

логических действий. Особенно много ошибок при анализе формул с импликациями.

Приведем таблицу истинности для импликации.

Таблица истинности импликации

A	B	A→B
0	0	1
0	1	1
1	0	0
1	1	1

Для учащихся, которые путаются с таблицей истинности, можно предложить следующий способ запоминания.

Таблица истинности импликации, объяснение

Истинность A	Истинность B	Мы вывели из утверждения A утверждение B. Как оценить наши действия?	A→B
1	1	Нам дали верные сведения, мы из них вывели верные заключения – все, в том числе и мы – молодцы!	1
0	0	Если мы вывели из неверных данных новое неправильное утверждение, **мы не допустили ошибки** — виноваты неправильные данные и те, кто их предоставил!	1

Истинность А	Истинность В	Мы вывели из утверждения А утверждение В. Как оценить наши действия?	А? В
0	1	**Мы не допустили ошибки**, если из неправильных данных вывели правильное заключение – мало ли что можно получить из неверных данных!	1
1	0	А вот если мы из верных данных получили неверные, **мы точно допустили ошибку,** и виноваты в ошибке только мы!	0

Отсюда, между прочим, понятно, что отбраковывать варианты следует с помощью единственной строки, которая дает нуль — обычно это сделать проще, чем подтвердить импликацию для трех строк, котрые дают 1.

▶ Понятно, что для истинности ¬ ((*X* кратно 2) → → (*X* кратно 4)) должно быть ложно ((*X* кратно 2) → (*X* кратно 4)). Единственная возможность для этого — это когда (*X* кратно 2) истинно, а (*X* кратно 4) ложно. Это позволяет начать отбраковку вариантов. Вариант 1 не годится, так как *X*=7 не кратно 2, то есть не делится на 2. А варианты 2) и 4) не годятся, так как числа 8 и 12 кратны 4. Остается вариант 3), как раз в нем *X*=10 кратно 2 и при этом не кратно 4.

Вариант ответа 3) и является правильным решением задачи.

Проверим его прямыми подсчетами логических значений.

X	(X кратно 2)	(X кратно 4)	(X кратно 2) → (X кратно 4)	¬((X кратно 2) → (X кратно 4))
логические значения условий				
7	0	0	1	0
8	1	1	1	0
10	1	0	0	1
12	1	1	1	0

√ Еще один способ решения основан на известном тождестве, которое *следует обязательно запомнить:*

$$A \to B = \neg A \vee B.$$

Еще лучше при этом использовать для обозначения логического отрицания черту сверху:

$$A \to B = \overline{A} \vee B.$$

Знак ¬ просто изобрели типографы, чтобы все размещать в одну строку. Но верхняя черта позволяет избавиться от многих запутывающих скобок.

Итак,

$$\neg((X\ кратно\ 2) \to (X\ кратно\ 4)) =$$
$$= \overline{(X\ кратно\ 2) \to (X\ кратно\ 4)} =$$
$$= \overline{\overline{(X\ кратно\ 2)} \vee (X\ кратно\ 4)} =$$
$$= \overline{\overline{(X\ кратно\ 2)}} \wedge \overline{(X\ кратно\ 4)} =$$
$$= (X\ кратно\ 2) \wedge (X\ не\ кратно\ 4).$$

При этом, конечно, мы использовали еще два известных логических тождества. Во-первых, тождество, позволяющее

убрать двойное отрицание $\overline{\overline{A}} = A$. Во-вторых, закон де Моргана: $\overline{A \vee B} = \overline{A} \vee \overline{B}$.

Зато теперь нам совершенно очевидно, что нас интересуют числа, которые делятся на 2 и при этом не делятся на 4. Из предложенных вариантов это только 10!

A11 Минимальный расход памяти

? В некоторой стране автомобильный номер длиной 5 символов составляют из заглавных букв (задействовано 30 различных букв) и любых десятичных цифр в любом порядке.

Каждый такой номер в компьютерной программе записывается минимально возможным и одинаковым целым количеством байтов (при этом используют посимвольное кодирование, и все символы кодируются одинаковым и минимально возможным количеством бит).

Определите объем памяти, отводимый этой программой для записи 50 номеров.

1) 100 байт
2) 150 байт
3) 200 байт
4) 250 байт

√ Для решения задачи следует знать формулу: если у нас есть N позиций и на каждую позицию мы можем поставить M различных знаков, то всего можно получить $\mathbf{M^N}$ различных вариантов сообщений, паролей, номеров, чисел — в общем, неважно чего.

Чтобы не перепутать, куда ставить M, куда N, можно всегда вспомнить контрольный пример, который должен твер-

до сидеть в памяти у того, кто хоть чуть изучал информатику: имея в байте 8 позиций и на каждую ставя 0 или 1 (2 знака), мы можем получить 256 различных вариантов.

Если и это запомнить трудно, можно попробовать вариант: две цифры и одна позиция и все варианты перечислить: 0 и 1. Два варианта это, конечно, 2^1, ведь 1^2, конечно, равна 1.

Однако использовать эту формулу сразу, например, вычисляя **$(10+30)^5$**, — ложный путь, который не приведет к ответу.

Еще, конечно, следует внимательно читать задачу. Типичная ошибка, как я говорю своим ученикам, «в спешке придумать себе из тех же слов совсем другую задачу и успешно её решить».

▶ Внимательно читаем задачу. В каждую из 5 позиций пароля (Внимание! На самом деле 5 — это не то число, которое нам нужно сейчас, оно нам потребуется позже!) можно поставить символ. Разнообразие пригодных для использования символов равно **30 буквам в заглавном регистре+10 цифр, итого 40 различных вариантов.**

Переформулируем задачу.

Имеется набор символов, состоящий из 30 букв в заглавном регистре +10 цифр, итого 40 различных вариантов.

Каждый символ кодируется минимально возможным количеством бит, одинаковым для всех символов. Вопрос первый: сколько потребуется бит?

С помощью таких символов кодируется пароль из 5 символов. Битовые коды всех 5 символов, входящих в пароль, объединяются в одну цепочку, которая размещается в байтах памяти. Вопрос второй: сколько потребуется (минимально) байт?

Нужно хранить 50 паролей. Вопрос третий — сколько на них потребуется байт?

Вспомним, что кодировка ASCII дает нам 256 различных символов. Почему 256? Потому что на каждый символ тратится 8 бит, а используя две цифры 0 и 1, можно записать 28=256 различных восьмибитовых чисел — от 0000 0000 до 1111 1111 (разбили по четыре цифры для удобства!). А если на каждый символ тратить только 7 бит, мы можем закодировать 27=128 различных символов, если только 6 бит, то 26=64 варианта>40, если 5 бит, то только 25=32 и так далее. Видим, что 5 бит не хватит, придется брать 6 бит, которых хватит с небольшим излишком для кодирования каждого символа в автомобильном номере. Это ответ на первый вопрос.

Для 5 символов, входящих в автомобильный номер, потребуется 5*6=30 бит. Но поскольку номер записывается в компьютерной программе с помощью целого количества байт, придется взять 4 байта=32 бита>30 бит, то есть тоже с небольшим вынужденным излишком.

Наконец, теперь мы сможет ответить на вопрос, сколько байт потребуется на 50 автомобильных номеров — 200 байт. *Правильным будет вариант ответа 3).* ◀

√ Воспоминание из когда-то прочитанного по ассоциации с этой задачей. Коротко даже не могу объяснить смысл ассоциации, но, по-моему, подходит.

Во времена парусного флота каждый корабль украшался красивой резьбой. Когда же наступила эпоха пароходов, в последних попытках конкурировать с ними парусники стали делать по упрощенной схеме, как на конвейере, главное — быстро и дешево! Старые мастера бурчали: «Их строят милями, а перед спуском на воду режут на кабельтовы» (примечание: миля, кабельтов — морские меры длины, миля равна 1852 метра, кабельтов — 1/10 часть мили).

A12 Что делает программа с массивом?

? В программе описан одномерный целочисленный массив A с индексами от 0 до 10. Ниже представлен фрагмент этой программы, записанный на разных языках программирования, в котором значения элементов сначала задаются, а затем меняются.

Паскаль[1]

```
for i:=0 to 10 do
  A[i] := i-1;

for i:=1 to 10 do
  A[i-1] := A[i];
A[10] := 10;
```

Как изменятся элементы этого массива после выполнения фрагмента программы ?

1) все элементы, кроме последнего, окажутся равны между собой;
2) все элементы окажутся равны своим индексам;
3) все элементы, кроме последнего, будут сдвинуты на один элемент вправо;
4) все элементы, кроме последнего, уменьшатся на единицу.

▶ Первые две строки программы производят начальное заполнение массива. Чтобы понять, какие значения получат элементы массива, можно выполнить несколько проб:

[1] Приводится вариант программы только на Паскале.

первый раз тело цикла будет выполняться при i=0, при этом A[0] получит значение –1, при втором проходе i=1, A[1] получит значение 0, ..., а когда тело цикла будет выполняться последний раз, то будет i=10, и A[10] получит значения 9.

Таблица 2

Значения в массиве после выполнения первого цикла

i	0	1	2	...	8	9	10
A[i]	–1	0	1	...	7	8	9

Второй цикл выполняется, начиная с **i=1** и до **i=10**. Мы видим в теле цикла оператор **A[i-1] := A[i]**. При i=1 это даст A[0] := A[1], при i=2 соответственно A[1] := A[2] и т.д. Таким образом заполнятся все элементы, кроме последнего. Последний элемент A[10] получит свое значение уже вне цикла — в отдельном операторе A[10] := 10;.

Таблица 3

Значения в массиве после выполнения второго цикла

i	0	1	2	...	8	9	10
A[i]	0	1	2	...	8	9	10

Вывод. *Правильный вариант ответа 2).* ◀

A13 Робот в лабиринте

? Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости, включает в себя 4 команды-приказа и 4 команды: проверки условия.

вверх	**вниз**	**влево**	**вправо**

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

Если робот начнет движение в сторону находящейся рядом с ним стены, то он разрушится, и программа прервется.

Другие 4 команды проверяют истинность *условия* отсутствия стены у каждой стороны той клетки, где находится РОБОТ.

сверху свободно	**снизу свободно**	**слева свободно**	**справа свободно**

Цикл

ПОКА < *условие* >
последовательность команд
КОНЕЦ ПОКА

выполняется, пока условие истинно.

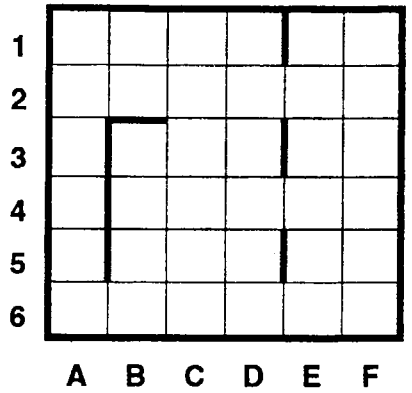
В конструкции

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

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

Сколько клеток лабиринта соответствуют требованию, при котором, выполнив предложенную программу, РОБОТ уцелеет и остановится в закрашенной клетке (клетка А1)?

```
НАЧАЛО
   ПОКА < слева свободно ИЛИ сверху свободно >
   ЕСЛИ < слева свободно >
   ТО влево
   ИНАЧЕ вверх
   КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```



1) 8 2) 12 3) 17 4) 21

√ В каждый момент у робота, выполняющего программу, есть три вариант действий.

1. Если есть возможность сделать шаг влево, то сделать шаг влево.

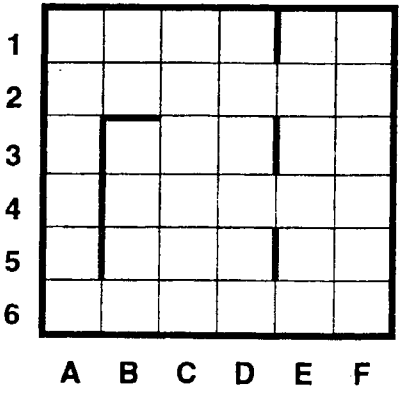
2. Если нет возможности сделать шаг влево, но есть возможность сделать шаг вверх, то сделать шаг вверх.

3. Если нет возможности сделать шаг влево и нет возможности сделать шаг вверх, то оставаться на месте и завершить программу.

Причем указанные действия выполняются именно с такой приоритетностью — второй вариант пробуется, только если первый не прошел, а третий вариант пробуется только в том случае, если не прошли первые два.

Программа движения РОБОТа проста, и можно пробовать ее непосредственно для каждой клетки, однако сделать это для 36 клеток потребует много времени и, можно поручиться, где-нибудь да ошибешься. Поэтому следует как-то сокращать рассуждения.

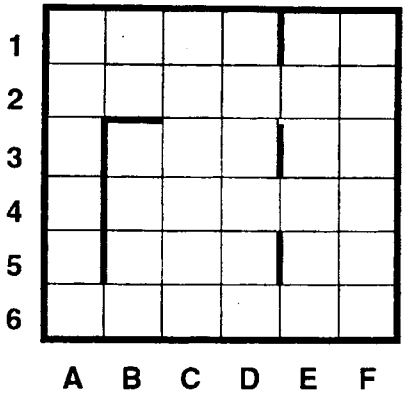
▶ Будем последовательно наращивать количество клеток, которые приведут нас к нужному результату.



Прежде всего заметим, что из клеток B1, C1, D1 робот, действуя по варианту 1, придет в A1.

Из клеток A2...A6, действуя по варианту 2, робот придет в A1.

Наконец, если он первоначально находится в самой клетке A1, он по варианту 3 останется на месте при выполнении программы, поэтому клетка A1 формально также годится.



К перечисленным ранее клеткам добавим еще клетки, стартуя из которых робот придет в A1. Например, двигаясь из клеток B6...F6, робот сначала дойдет до клетки A6, а затем дойдет из этой клетки до клетки A1. При этом на первом этапе он действует по варианту 1 из вышеуказанных, а затем — по варианту 2. Такая же ситуация со стартовыми клетками B2...B6 — робот сначала дойдет по горизонтали до A2, а затем до A1.

Самая сложная траектория — для стартовых клеток E3...F3. Робот сначала попытается пойти влево и для клетки F3 действительно сделает шаг, затем сделает шаг вверх в E2, а затем, как говорилось выше, в два приема дойдет до A1.

Подсчитав закрашенные клетки, мы тем самым найдем правильный ответ.

Правильный ответ — вариант 4) ◀

РЕШЕНИЕ ЗАДАЧ ЧАСТИ В

Общие замечания

Как известно, часть В отличается от части А прежде всего тем, что нет предлагаемых четырех вариантов ответа, из которых следует выбрать правильный. Есть только общие указания, что должно быть ответом: число или цепочка символов. Многие спорят по поводу того, насколько усложняет решение заданий отсутствие возможных ответов. На мой взгляд — не очень!

При записи ответов обратите внимание как раз на их форму. Ответы должны быть записаны именно так, как указано в условии задания. Следует помнить, что часть В проверяется машинным способом, и, если вы запишите, например, N=10 вместо просто 10, как требовалось, проверяющая программа сочтет ваш ответ неверным. Будет очень обидно!

B1 | *Исполнитель Арифметик*

? У исполнителя Арифметик две команды, которым присвоены номера:

1. прибавь 2,

2. умножь на 3.

Первая из них увеличивает число на экране на 2, вторая — утраивает его.

Например, **21211** — это программа:

умножь на 3

прибавь 2
умножь на 3
прибавь 2
прибавь 2,
которая преобразует число 1 в 19.

Запишите порядок команд в программе преобразования **числа 3 в число 69**, содержащей не более 5 команд, указывая лишь номера команд.

(Если таких программ более одной, то запишите любую из них)

√ Задача простая, но требует холодной головы. Сколько раз сам наступал на грабли — вместо программы, которая из 3 делает число 69, придумывал программу, которая из 1 делает число 19 (ясно, что 69 и 19 — это условные числа, с помощью них я кратко показываю суть ошибки)!

Еще один вывод из этой задачи состоит в том, что иногда задачу удобно решать с конца:

«Начинать лучше с конца, чем с начала. Тогда всё уже становится намного понятней!» (Юрий Татаркин. Персональная страница на сайте афоризмов http://www.aphorism.ru)

▶ Интуитивное соображение: число 69 довольно велико, значит, надо стремиться использовать больше команд умножения — они быстрее наращивают число. Наверное, последней командой будет команда умножения числа 23 на 3. Число 23 умножением не получить, значит оно получено командой прибавь 2 из числа 21. Число 21, наверное, получено из 7 умножением на 3. У нас осталось не более двух команд, и далее уже можно догадаться, что число 7 получено из исходного числа 3 прибавлением дважды числа 2.

Эти соображения быстро подсказывают результат: команды **11212** преобразуют числа следующим образом:

$$3 \to 5 \to 7 \to 21 \to 23 \to 69.$$

Ответ: **11212** ◀

В бланке ответов это будет выглядеть примерно так:

Кстати, обратите внимание на образцы написания букв и цифр, которые даны в бланке ответов, и старайтесь им следовать. Например, обратите внимание на то, что у единички нет «пяточки» внизу. При использовании какого-нибудь оригинального написания символов возможна ситуация, когда проверяющая машина не распознает символ правильно, хотя разработчики постоянно совершенствуют аппаратуру и программное обеспечение и вероятность ошибки незначительна.

B2 | *Результат работы программы*

? Определите значение переменной с после выполнения следующего фрагмента программы *(записанного ниже на разных языках программирования)*[1]. Ответ запишите в виде целого числа.

Паскаль

```
a := 30;
b := 14;
a := a - 2 * b;
```

[1] Мы приводим программы только на языке Паскаль.

```
if a > b then
    c := b + 2* a
else
    c := b - 2 * a;
```

√ Для успешного решения такой задачи требуется некоторые минимальные представления о программировании. И аккуратность!

▶ Приведем один из вариантов решения. Это, скорее, конспект того, как следует мыслить при решении задачи, чем конспект оформления. На самом деле, достаточно карандашом прямо на листе с заданиями сделать пометки о числах и условиях.

Результат работы строки программы

a := 30;	**a** получит значение **30**
b := 14;	**b** получит значение **14**
a := a - 2 * b;	**b** получит значение **30 – 2*14 = 2**
if a > b then	Условие **2 >14** имеет значение **false**, поэтому переходим на ветвь **else**
c := b + 2*a	
else	
c := b - 2*a;	**c** получит значение **14-2*2 = 10**

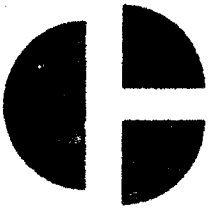
Ответ: 10 ◀

В3 Электронные таблицы

? Дан фрагмент электронной таблицы.

	A	B	C
1	2	4	
2	=(B1-A1)/2	=2-A1/2	=(C1-A1)*2-4

Какое число должно быть записано в ячейке C1, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:C2 соответствовала рисунку?



▶ Вычисляя значения тех ячеек, которые мы можем вычислить, то есть те формулы, в которых не участвует еще неизвестная нам клетка C1, получаем:

для ячейки A2 по формуле =(B1-A1)/2 значение 1,

для ячейки B2 по формуле =2-A1/2 значение 1.

Анализируя диаграмму, легко понять, что эти единички дают нам два правых сектора диаграммы. Значит, левый сектор соответствует числовому значению 2, которое должно быть значением формулы =(C1-A1)*2-4. Учитывая, что A1 имеет значение 2, получим простенькое уравнение (C1-2)*2-4=2, из которого найдем C1=5.

Ответ: 5 ◀

B4 | *Азбука Морзе*

? Азбука Морзе позволяет кодировать символы для сообщений по радиосвязи, задавая комбинацию точек и тире. Сколько различных символов (цифр, букв, знаков пунктуации и т.д.) можно закодировать, используя код азбуки Морзе длиной **не менее четырёх** и **не более пяти** сигналов (точек и тире)?

√ Как мы уже говорили в задаче А11, если у нас есть *N* позиций и на каждую позицию мы можем поставить 2 различных знака, то всего можно получить $\mathbf{2^N}$ различных вариантов сообщений, паролей, номеров, чисел — в общем, неважно чего.

▶ Из условия 4<=N<=5. Поэтому мы можем иметь $\mathbf{2^4=16}$ различных цепочек сигналов длиной в 4 сигнала и $\mathbf{2^5=32}$ различных цепочек сигналов длиной в 5 сигналов.

Ответ: 48 ◀

B5 | *Определить результаты работы программы*

Определите, что будет напечатано в результате работы следующего фрагмента программы.

Паскаль[1]

```
Var n, s : integer;
BEGIN
  n := 0;
```

[1] Приведен код программы только на языке Паскаль.

```
  s := 0;
  while s<=35 do
    begin
      n := n+1;
      s := s+4
    end;
  write(n);
END.
```

√ Для успешного решения такой задачи требуется некоторые минимальные представления о программировании.

▶ Заметим, что переменная n не может оказать влияние на условие продолжения/окончания цикла, а также и на значение переменной s. Это просто счетчик, который отмечает, сколько раз выполнится тело цикла.

Задачу можно переформулировать так: сколько раз нужно к начальному значению 0 переменной s прибавить 4, чтобы результат стал больше 35.

Такую задачу решить легко. После 8 раз получим s=32<=35, а вот после 9 раз получим s=36>35, что нам и нужно.

Ответ: 9 ◀

√ Будьте внимательны к условию продолжения/окончания цикла. Например, s<32 и s<=32 дадут разные ответы при начальном s:=0 и приращении s:=s+4;.

B6 Рекурсивное вычисление функций

? Алгоритм вычисления значения функции $F(n)$, где n — натуральное число, задан следующими соотношениями:

$F(1) = 1$

$F(n) = F(n - 1)*n$, при $n > 1$.

Чему равно значение функции $F(5)$?

В ответе запишите только натуральное число.

√ В этом демо-варианте используется широко известная формула вычисления факториала. Следует быть готовым к тому, что в реальных вариантах будет что-нибудь посложнее.

▶

F(1) = 1
F(2) = F(1)*2 = 2
F(3) = F(2)*3 = 2*3 = 6
F(4) = F(3)*4 = 6*4 = 24
F(5) = F(4)*5 = 24*5 = 120

Ответ: 120 ◀

B7 Число в различных системах счисления

? Запись десятичного числа в системах счисления с основаниями 3 и 5 в обоих случаях имеет последней цифрой 0. Какое минимальное натуральное десятичное число удовлетворяет этому требованию?

▶ Мы хорошо знаем, что в десятичной системе счисления числа, оканчивающиеся на 0 — это те, которые делятся на 10. Точно так же в троичной системе счисления числа, оканчивающиеся на 0 — это те, которые делятся на 3. И так же для 5.

Теперь переформулируем задачу. Требуется найти минимальное натуральное число, которое одновременно делится на 3 и на 5, и записать его в десятичной системе счисления.

Теперь легко догадаться, что это 15.

Проверка. $15_{10}= 120_3=30_5$ Мы хотя бы проверили, что это число делится на 3 и на 5. В том, что оно минимальное из таких, можно убедиться так: в пятеричной системе счисления из меньших чисел на 0 заканчиваются только $20_5=10_{10}=101_3$ и $10_5=5_{10}=12_3$, но эти числа на 3 не делятся.

Ответ: 15 ◀

B8 | *Результат работы программы*

? Ниже на четырех языках записан алгоритм[1]. Получив на вход число x, этот алгоритм печатает два числа: a и b. Укажите наибольшее из таких чисел x, при вводе которых алгоритм печатает сначала 2, а потом 21.

Паскаль

```
var x, a, b: integer;
begin
  readln(x);
  a := 0; b := 1;
```

[1] Мы приводим код программы только на языке Паскаль.

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

√ Здесь ситуация посерьезней, чем в задании **B5**. Требуются навыки отладки программ. Именно такие накопленные навыки позволяют анализировать работу сложных программ. Хотя в пределах, необходимых для решения такой задачи, наверное, этому можно научить и любого толкового школьника.

▶ Как видно из последних операторов, в конце работы программы переменная *a* должна иметь значение 2, а переменная *b* должна иметь значение 21.

Далее заметим, что переменная *a* нигде не фигурирует в правых частях операторов присваивания или в условиях. Значит, она никак не влияет на другие переменные и на проверяемые условия. Это просто счетчик, фиксирующий, сколько раз выполнилось тело цикла. Каждый раз в теле цикла от **x** «откусывается» правая цифра (**x := x div 10;**), и длина **x** уменьшается на 1 символ. Цикл завершается, когда **x** станет равным 0. Так как счётчик (переменная *a*) при этом насчитал 2 цикла, значит, переменная **x** должна иметь две десятичные цифры, обозначим их M_1M_0.

x mod 10 — это каждый раз последняя цифра числа. Когда тело цикла выполняется первый раз, это будет цифра M_0, при втором выполнении тела цикла это будет цифра M_1. Каждый раз в теле цикла переменная *b* умножается

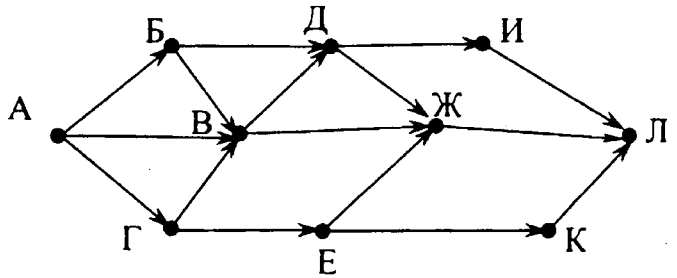
на эту цифру. Обратите внимание, что начальное значение b=1; если бы было 0, то умножение дало бы нам всегда 0. Делаем вывод, что после завершения оператора цикла b будет равно произведению этих цифр и равно 21.

Понятно, что 21 может быть только произведением 3 и 7. Число ***x*** может быть равно 37 или 73. По условию нам нужно наименьшее из них, то есть 37.

Ответ: 37 ◀

B9 | *Подсчитать количество дорог*

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



▶ У каждого узла будем указывать число, показывающее, сколькими вариантами можно добраться до этого узла из пункта А. Поставим у узла А условную единицу. Для всех других узлов **правило** таково: **нужно суммировать числа из тех узлов, от которых есть стрелки к данному узлу**. При этом придется это делать в определенной

последовательности — мы сможем «обсчитать» некоторый узел только тогда, когда будут «обсчитаны» все узлы, из которых к нему приходят стрелки. Узлы Б и Г обсчитываем сразу после узла А. А вот расчет для узла В можно делать только после Б и Г, и, естественно, А.

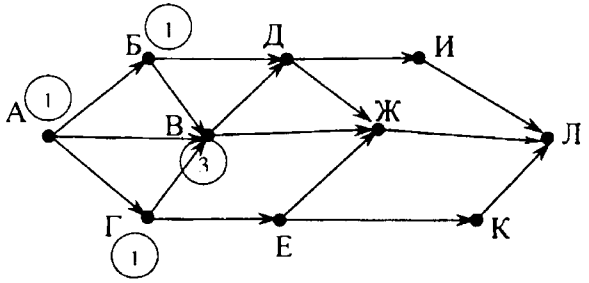


Рис. 4. Расчёт количества путей для первых 4-х узлов

После этого можно вести расчеты для узлов Д и Е. Объясним себе это так: к узлу Д можно прийти из узла Б по одной дороге и из узла В также по одной дороге. Но до узла В можно добраться из А тремя способами, и до узла Б можно добраться из А одним способом. Значит, всего разных путей из А в Д будет четыре.

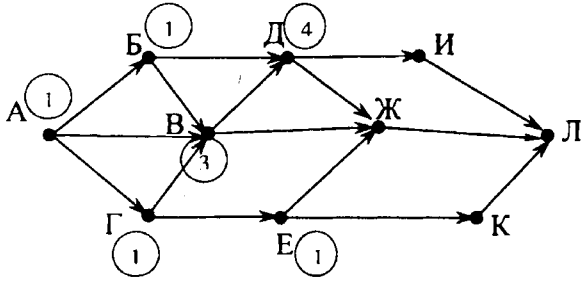


Рис. 5. Расчёт количества путей (очередной этап)

Далее таким же образом вычислим такие числа и для оставшихся узлов. Например, в узел Ж приходят дороги

из узлов В, Д и Е. Поэтому общее количество разных путей из А в Ж будет равно 3+4+1=8.

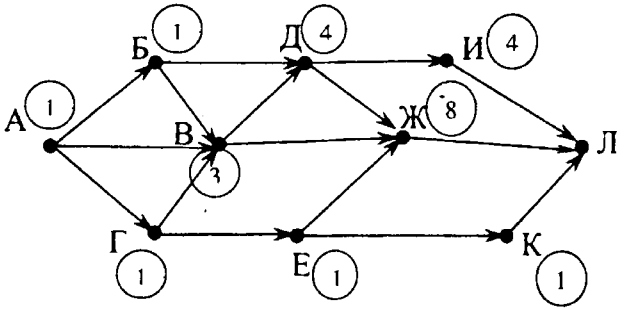


Рис. 6. Расчёт количества путей (очередной этап)

Только теперь мы можем подсчитать число различных путей из А в Л.

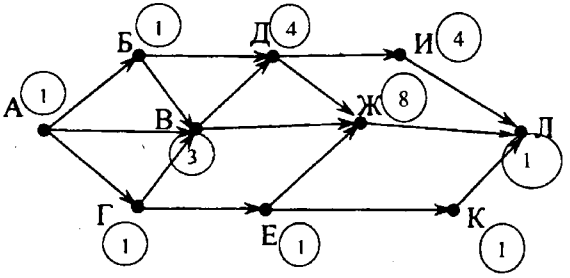


Рис. 7. Расчёт количества путей (завершение)

Перечислим все возможные пути.

1. АБДИЛ 2. АБДЖЛ 3. АБВДИЛ 4. АБВДЖЛ 5. АБВЖЛ
6. АВДИЛ 7. АВДЖЛ 8. АВЖЛ 9. АГВДИЛ
10. АГВДЖЛ 11. АГВЖЛ 12. АГЕЖЛ 13. АГЕКЛ

Ответ: 13 ◀

B10 Время скачивания

? Документ размером 20 Мбайт можно передать с одного компьютера на другой двумя способами.

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

Б. Передать по каналу связи без использования архиватора.

Какой способ быстрее и насколько, если:

- средняя скорость передачи данных по каналу связи составляет 2^{20} бит в секунду;
- объем сжатого архиватором документа равен 20% исходного;
- время, требуемое на сжатие документа, — 5 секунд, на распаковку — 1 секунда?

В ответе напишите букву А, если быстрее способ А, или Б, если быстрее способ Б. Сразу после буквы напишите число, обозначающее, на сколько секунд один способ быстрее другого.

Так, например, если способ Б быстрее способа А на 23 секунды, в ответе нужно написать Б23.

Единиц измерения «секунд», «сек.», «с.» к ответу добавлять не нужно.

▶ Расчет по способу А.

Объем сжатого документ 20 Мбайт*20/100=4 Мбайта.

Время передачи сжатого документа:

$$\frac{4\text{М байта}}{2^{20}\text{ бит/сек}} = \frac{4 * 1024 * 1024\text{ байт}}{2^{20}\text{ бит/сек}} = \frac{4 * 2^{10} * 2^{10} * 8\text{ бит}}{2^{20}\text{ бит/сек}} = 32\text{ сек.}$$

Добавим еще время на сжатие и распаковку: 32+5+1=38 сек.

Расчет по способу Б:

$$\frac{20\,\text{Мбайт}}{2^{20}\,\text{бит/сек}} = \frac{20 * 1024 * 1024\,\text{байт}}{2^{20}\,\text{бит/сек}} = \frac{20 * 2^{10} * 2^{10} * 8\,\text{бит}}{2^{20}\,\text{бит/сек}}$$

$= 160$ сек.

Вывод: способ А быстрее способа Б на 122 сек. Осталось аккуратно записать ответ.

Ответ: А122 ◀

B11 | *IP-адрес и маска в сети*

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

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

IP-адрес узла: 217.19.128.131

Маска: 255.255.192.0

При записи ответа выберите из приведенных в таблице чисел четыре элемента IP-адреса и запишите в нужном порядке соответствующие им буквы. Точки писать не нужно.

A	B	C	D	E	F	G	H
0	16	19	64	128	131	192	217

Пример.

Пусть искомый IP-адрес 192.168.128.0, и дана таблица.

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

В этом случае правильный ответ будет записан в виде: HBAF.

▶ Число 255 в двоичной записи состоит из 8 единиц 1111 1111. Поэтому при поразрядной конъюнкции этого числа с любым другим числом из 8 бит это другое число останется без изменений. Итак, первые два поля в IP-адресе узла при поразрядной конъюнкции с полями маски так и останутся 217 и 19.

Число 0 в двоичной записи состоит из 8 нулей 0000 0000. При поразрядной конъюнкции этого числа с любым другим числом из 8 бит оно производит противоположный эффект — обнуляет результат. То есть самое правое поле в IP-адресе узла при конъюнкции с таким же полем маски даст в точности 0.

Осталось только выполнить поразрядную конъюнкцию для третьих частей адреса узла и маски, то есть для чисел 128 и 192.

128	1	0	0	0	0	0	0	0
192	1	1	0	0	0	0	0	0
Результат поразрядной конъюнкции	1	0	0	0	0	0	0	0

В десятичном виде это число 128 (нам повезло, никаких дополнительных суммирований не потребовалось).

Итак, адрес получен 217.19.128.0 . Осталось только собрать его из доступных кусочков.

Ответ: НСЕА ◀

√ Так как нам попался очень простой случай, рассмотрим выдуманный пример поразрядной конъюнкции, где чуть большей сложностей.

180	1	0	1	1	0	1	0	0
240	1	1	1	1	0	0	0	0
Результат поразрядной конъюнкции	1	0	1	1	0	0	0	0
Вес разряда	128	64	32	16	8	4	2	1

Результатом поразрядной конъюнкции будет

128+32+16=176.

B12 Запросы к поисковому серверу

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

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

Запрос	Найдено страниц (в тысячах)
Фрегат | *Эсминец*	3400
Фрегат & *Эсминец*	900
Фрегат	2100

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

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

▶ Приведенная схема поясняет условие и подсказывает решение.

Эсминец без фрегата = **3400–2100 =1300.**
Эсминец = **900+1300 =2200.**

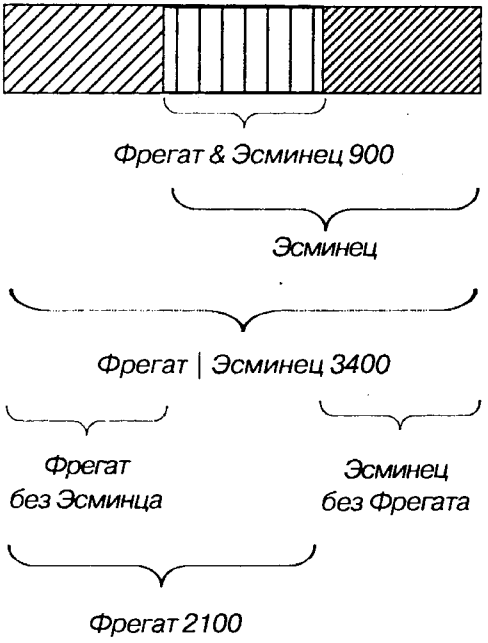


Рис. 8. Данные для расчета количества страниц в ответе поискового сервера

Ответ: 2200 ◀

B13 | *Исполнитель Удвоитель*

? У исполнителя Удвоитель две команды, которым присвоены номера:

1. прибавь 1,
2. умножь на 2.

Первая из них увеличивает на 1 число на экране, вторая удваивает его.

Программа для Удвоителя — это последовательность команд. Сколько есть программ, которые число 3 преобразуют в число 23?

▶ Обозначим K(N) — количество разных программ получения числа N из исходной 3 для исполнителя Удвоитель.

Число 4 можно получить из 3 одним способом — программа состоит из одной команды **прибавь 1**. Запишем это так K(4)=1.

Число 5 можно получить также единственным способом — из 4 командой **прибавь 1.** Так как способ получения 4 из 3 тоже 1, то существует только одна программа, позволяющая получить 5 из исходного 3. Эта программа состоит из двух команд: **прибавь 1; прибавь 1.** Запишем K(5) =1.

Но число 6 можно получить уже двумя путями — либо из пятерки прибавлением 1, либо из тройки умножением на 2. K(6)=2.

В дальнейшем же можно вывести простую формулу:

K(N) = K(N-1), если число N не делится на 2 и

K(N) = K(N-1)+K(N/2), если число N делится на 2.

Теперь все ясно:

K(7)=K(6)=2;

K(8)=K(7)+K(4)=2+1=3;

K(9)=K(8)=3;

K(10)=K(9)+K(5)=3+1=4;

K(11)=K(10)=4;

K(12)=K(11)+K(6)=4+2=6;;

K(13)=K(12)=6;

K(14)=K(13)+K(7)=6+2=8;

K(15)=K(14)=8;

K(16)=K(15)+K(8)=8+3=11;
K(17)=K(16)=11;
K(18)=K(17)+K(9)=11+3=14;
K(19)=K(18)=14;
K(20)=K(19)+K(10)=14+4=18;
K(21)=K(20)=18;
K(22)=K(21)+K(11)=18+4=22;
K(23)=K(22)=22.

Итак, можно написать 22 разные программы для исполнителя Удвоитель, которые исходную 3 преобразуют в число 23. ◀

√ K(24)=K(23)+K(12)=22+6=28;
Стоп, это уже лишнее. Надо помнить конечную цель!

Ответ: 22.

B14 *Результат работы сложной программы*

Определите, какое число будет напечатано в результате выполнения следующего алгоритма (для вашего удобства алгоритм представлен на четырех языках):

Паскаль

```
var a,b,t,M,R :integer;
Function F(x:integer):integer;
begin
 F := 3*(x-8)*(x-8);
end;
BEGIN
a := -20; b := 20;
M := a; R := F(a);
```

```
for t:=a to b do
 begin
 if (F(t)<R) then begin
   M := t;
   R := F(t);
 end;
 end;
write(M);
END.
```

√ Задание очень трудное. Требуется в полной мере проявить свое математическое и программистское образование. Но и тренировка в решении таких задач может дать хорошие результаты.

▶ Рассматривая функцию F, мы видим, что это квадратичная функция. Иногда говорят, что у нее один корень x=8, а иногда говорят, что она имеет два совпадающих корня x_1=8 и x_2=8. Нам это не особенно важно, для нас важно, что в этой точке x=8 располагается вершина параболы. Это — по оси X, а по оси Y ее значение будет равно в вершине в точности 0. Так как коэффициент 3 положительный, ветки параболы направлены вверх и в вершине будет минимальное значение. Это все нам хорошо известно из школьной алгебры. Для дальнейшего важно, что при $x < 8$ функция F убывает, то есть $F(x+1) < F(x)$, а при $x > 8$ — возрастает, то есть $F(x+1) > F(x)$.

Перед началом цикла переменные получат значения M=–20; R=F(–20); (сколько это будет, считать не надо — хлопотно!).

При t =–20 в теле цикла будет проверяться условие F(–20)<R, причем в этот момент R=F(–20). Условие ложно, значит M и R не изменят свои значения.

При t=−19 в теле цикла будет проверяться условие F(−19)<R при R=F(−20). Как сказано выше, из свойств функции F следует, что условие истинно, значит, M получит новое значение —19, R получит новое значение F(−19).

При t=−18 в теле цикла будет проверяться условие F(−18)<R при R=F(−19). Из свойств функции F снова следует, что условие истинно, значит, M получит новое значение —18, R получит новое значение F(−18).

Так будет повторяться до t=8.

При t=8 в теле цикла будет проверяться условие F(8)<R, причем в этот момент R=F(7). Условие по-прежнему истинно, значит, M получит новое значение 8, R получит новое значение F(8).

При t=9 ситуация изменится. В теле цикла будет проверяться условие F(9)<R (причем R=F(8)). Но теперь из свойств функции F уже следует, что условие ложно, значит M и R не изменят свои значения и останется M=8 и R=F(8).

Это будет продолжаться при всех t до конца цикла.

Напомним, что программа печатает в качестве результата значение M и нам нужно было узнать, что она напечатает.

Ответ: 8 ◀

√ Оказывается, вычислять сами значения функции в каких-то там точках вовсе и не потребовалось. Однако надо проявить бдительность — в данном условии требуется напечатать именно значение M, но вдруг в реальном варианте потребуется напечатать значение R!

B15 Сколько решений у системы логических уравнений

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

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

$$(\neg y_1 \vee y_2) \wedge (\neg y_2 \vee y_3) \wedge (\neg y_3 \vee y_4) = 1$$

$$(y_1 \to x_1) \wedge (y_2 \to x_2) \wedge (y_3 \to x_3) \wedge (y_4 \to x_4) = 1$$

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

▶ Система выглядит страшной, но есть поговорка «Глаза боятся, а руки делают!».

Рассмотрим первую строку. Конъюнкция истинна, когда истинно каждое из составляющих ее выражений. Это знают все, кто берется сдавать ЕГЭ по информатике (или я не прав?). Но многие не знают один полезный факт про импликацию: импликация A → B =1 тогда и только тогда, когда A <= B, если 0 и 1 рассматривать как числа. Взгляните на таблицу истинности импликации в разборе задания A10 и убедитесь! Этот факт может иногда здорово помочь. В данном случае он показывает нам, что $(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge (x_3 \to x_4) = 1$ равносильно требованию $(x_1 \le x_2)$ *и* $(x_2 \le x_3) \wedge (x_3 \le x_4)$.

Для анализа второй строки применим еще один факт про импликацию (он известен больше): ¬A∨B = A → B.

Но это сразу указывает нам на то, что вторая строка утверждает про «игреки» то же самое, что первая строка говорит про «иксы». Итак, вторая строка равносильна требованию:

$$(y_1 \le y_2) \text{ и } (y_2 \le y_3) \wedge (y_3 \le y_4).$$

Теперь легко понять, что третья строка равносильна требованию, чтобы $(y_1 \le x_1)$ и так же для других индексов.

Теперь можно переформулировать задачу: сколько существует наборов логических переменных $x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4$, при которых цепочка переменных x_1, x_2, x_3, x_4 — неубывающая и цепочка переменных y_1, y_2, y_3, y_4 — неубывающая, и при этом еще:

$$(y_1 \le x_1) \text{ и } (y_2 \le x_2) \text{ и } (y_3 \le x_3) \text{ и } (y_4 \le x_4).$$

Последнее условие можно выразить словами так: «иксы» возрастают раньше, чем «игреки».

Можно уже попытаться выписать все такие цепочки переменных. Но можно еще упростить себе расчеты, если каждую неубывающую цепочку переменных x_1, x_2, x_3, x_4 указать одним числом — самым левым из индексов, при котором переменная первый раз равна 1, а левее — они равны нулю. Например, цепочку $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 1$ можно кратко обозначить индексом 2. Надо только еще договориться, что цепочку $x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0$ будем обозначать индексом 5. Таким образом, оказывается, что существует ровно 5 неубывающих цепочек переменных x_1, x_2, x_3, x_4 и каждая из них может быть однозначно описана указанным индексом, всего индексы могут принимать значения от 1 до 5. Совершенно аналогичная ситуация для «игреков».

Условие «“иксы” возрастают раньше, чем “игреки”» можно переформулировать так: индекс для «иксов» должен быть меньше или равен индексу для «игреков».

Это позволяет приступить к расчетам.

Если индекс «игреков» равен 1, то индекс «иксов» может быть только тоже единицей.

Если индекс «игреков» равен 2, то индекс «иксов» может быть равен 0 или 1.

Если индекс «игреков» равен 1, то индекс «иксов» может быть только тоже единицей.

Если индекс «игреков» равен 2, то индекс «иксов» может быть равен 1 или 2.

Если индекс «игреков» равен 3, то индекс «иксов» может быть равен 1,2 или 3.

Если индекс «игреков» равен 4, то индекс «иксов» может быть равен 1,2,3 или 4.

Если индекс «игреков» равен 5, то индекс «иксов» может быть равен 1,2,3,4 или 5.

Например, если $x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1$ (индекс равен 3), то возможны случаи:

$y_1 = 0, y_2 = 0, y_3 = 1, y_4 = 1$ (индекс равен 3),

$y_1 = 0, y_2 = 0, y_3 = 0, y_4 = 1$ (индекс равен 4),

$y_1 = 0, y_2 = 0, y_3 = 0, y_4 = 0$ (индекс равен 5).

А вот вариант $y_1 = 0, y_2 = 1, y_3 = 1, y_4 = 1$ (индекс равен 2) для таких «иксов» годится — не выполняется условие $(y_2 \le x_2)$.

Всего имеем 1+2+3+4+5=15 различных наборов «иксов» и «игреков», удовлетворяющих всем условиям.

Ответ: 15 ◀

√ Задача, следует признать, непростая. Однако только умение решать такие сложные задачи и позволит набрать максимальный балл на ЕГЭ по информатике!

Мы завершаем анализ части В. Обратите внимание на то, что в конце части В дано указание: «Не забудьте перенести все ответы в бланк ответов №1». В самом деле, представьте, как будет обидно, когда вы решите эти задачи,

но, отложив внесение ответов в бланк на последние полчаса, в спешке последних минут забудете это сделать! Советуем все-таки сделать это не в последний момент, а хотя бы на исходе третьего часа работы.

РЕШЕНИЕ ЗАДАЧ ЧАСТИ С

Общие замечания

Как известно, часть С содержит задания, которые требуют ответа в развернутой форме в письменном виде. Здесь уже ситуация очень похожа на школьную контрольную работу.

Однако различия все-таки есть. В школе работу проверяет учитель, который отлично знает возможности учащегося, поэтому может простить ему некоторую небрежность в тексте, иногда простит даже ошибку, если видит ее случайность. Работы, выполненные на ЕГЭ, проверяются также людьми, но по некоторым формальным правилам. Поэтому решения-ответы должны быть оформлены строго определенным образом.

Я всегда призываю учащихся при оформлении решений из части С «наступить на горло собственной песне» и строго соблюдать все формальные правила. Например, отступы в программах ставить строго «по правилам», а не как вам удобно.

Избегайте всяких оригинальных решений! Балл за это не добавят!

В отличие от частей А и В, для которых приведены только правильные ответы, для части **С** в демо-варианте даны рекомендуемые его разработчиками способы оформленного решения. В принципе, к этому нечего добавить. Однако ведь до решения еще нужно дойти. Поэтому я привожу ниже некоторые соображения и свои решения, которые в некоторой степени являются еще и указанием на то, как найти решение задания.

C1 Анализ программы и исправление кода

? Требовалось написать программу, при выполнении которой с клавиатуры считывается координата точки на прямой (x — действительное число) и определяется принадлежность этой точки одному из выделенных участков B и D (включая границы). Программист торопился и написал программу неправильно.

Паскаль[1]

```
var x: real;
begin
 readln(x);
 if x>=-3 then
   if x<=9 then
     if x>1 then
       write("не принадлежит")
     else
       write("принадлежит")
end.
```

Последовательно выполните следующее.

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D и E). Границы(точки −3,1,5 и 9) принадлежат заштрихованным областям (B и D соответственно).

[1] Приведена программа только на Паскале.

Область	Условие 1 (x>=–3)	Условие 2 (x<=9)	Условие 3 (x>1)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					

В столбцах условий укажите «Да», если условие выполнится, «Нет» если условие не выполнится, «—» (прочерк), если условие не будет проверяться, «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.)

√ Для пояснения ошибки, имеющейся в программе, запишем ее в более понятном и эквивалентном с точки зрения результатов работы, но более громоздком виде.

```
if x>=-3
  then
   begin
      if x<=9
        then
```

```
          begin
            if x>1
              then write("не принадлежит")
              else write("принадлежит")
          end
        else
          begin
            {блок 2}
          end
    end
  else
    begin
      {блок 3}
    end;
```

Теперь видно, что существенный недостаток программы состоит в том, что вывод сообщений «принадлежит»/ «не принадлежит» выполняется только внутри одного из трех ветвлений. Если введенное значение x взято из зоны A, не выполнится условие x>=–3, поэтому программа должна выполнить переход в то место, которое мы обозначили блок 3. Но там нет никаких операторов, поэтому программа ничего не выведет в ответ, и это тоже означает неправильную обработку. Итак, зона A обрабатывается неверно.

Аналогично, если значение x взято из зоны E, условие x>=–3 выполнится, но не выполнится условие x<=9, поэтому программа должна выполнить переход в то место, которое мы обозначили блок 2, и опять ничего не выведет. Итак, зона E также обрабатывается неверно.

Еще одна ошибка в том, что нигде в программе не упоминается разграничивающая зоны C и D точка x=5. Для обеих этих зон выполняются условия x>=–3, x<=9, x>=1b, и программа выведет сообщение «не принадлежит», что верно для зоны C и неверно для зоны D.

Зона В обрабатывается верно.

После такого анализа мы можем заполнить таблицу, а также внести исправления в программу.

Типичный способ исправления — написать составное сложное условие принадлежности зонам В или D, и использовать при этом одно ветвление. «Сложным» оно называется только с точки зрения правил языка Паскаль, поскольку в нем используются логические связки AND и OR. На самом деле программисты предпочитают на практике именно такие «надежные» способы, где легко понять, что программный код соответствует условию задачи.

Вообще говоря, в некоторых подобных задачах возможен еще вариант (при заданных условиях задачи он невозможен), когда программа для какой-то зоны в некоторых точках x давала бы один ответ, а в некоторых — другой.

Далее приведен в неизменном виде вариант оформления правильного решения, представленный разработчиками.

▶ 1.

Область	Условие 1 (y>=x)	Условие 2 (y>=0)	Условие 3 (y<=2-x*x)	Программа выведет	Область обрабатывается верно
A	Нет	–	–	–	Нет
B	Да	Да	Нет	Принадлежит	Да
C	Да	Да	Да	Не принадлежит	Да
D	Да	Да	Да	Не принадлежит	Нет
E	Да	Нет	–	–	Нет

2. Возможная доработка (Паскаль):

```
if (x>=-3) and (x<=1) or (x>=5) and (x<=9) then
   write("принадлежит")
 else
   write("не принадлежит")
```
◀

От себя я бы рекомендовал записать еще понятнее:

```
if ((x>=-3) and (x<=1)) or
   ((x>= 5) and (x<=9))
 then
   write("принадлежит")
 else
   write("не принадлежит")
```

Лишние скобки и разбивка на две строки резко увеличивают читаемость, то есть понятность кода. Поскольку проверяющие — это обычно самые квалифицированные учителя информатики или преподаватели программирования в вузах, им понравится такое красивое оформление кода. Вы привлечете их симпатии на свою сторону, и, возможно, в какой-то промежуточной ситуации, где балл можно добавить, а можно и не добавлять, они примут решение в вашу пользу! Не упускайте такую возможность!

Далее мы приводим еще один вариант исправления кода, указанный теми же разработчиками. Но мы его не рекомендуем — в нем вероятность ошибки гораздо выше!

▶ Возможны и другие способы доработки. Например:

```
if x>=-3 then
  if x<=1 then
    write("принадлежит")
  else
    if x>=5 then
      if x<=9 then
        write("принадлежит")
      else
        write("не принадлежит")
```

```
    else
      write("не принадлежит")
else
  write("не принадлежит")

Другой пример:
if abs(abs(x-3)-4)<=2 then
   write("принадлежит")
 else
   write("не принадлежит")
```

C2 Написать программный код

? Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от 0 до 100. Опишите на русском языке или на одном из языков программирования алгоритм, позволяющий найти и вывести произведение элементов массива, которые имеют нечетное значение и делятся на три. Гарантируется, что в исходном массиве есть хотя бы один элемент, значение которого нечетно и делится на 3.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается использовать часть из них. Исходные данные подобраны так, что результат произведения не выходит за пределы объявленных типов данных.

Паскаль

```
const
  N = 30;
var
```

```
  a: array [1..N] of longint;
  i, j, p: longint;
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 *Free Pascal* 2.4) 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).

√ Найти произведение значений элементов массива — одна из типовых задач обработки массивов. Эту задачу, точнее, программный код для ее решения, следует адаптировать к заданию. Отметим, что в отличие от программ, которые выполняют суммирование элементов, начальным значением «накопительной» переменной следует взять 1. Кроме того, конечно, следует добавить проверку дополнительных условий, чтобы выбирать только нужные элементы массива. Делимость на заданное число в Паскале обычно проверяют так:

(a[i] mod 2 <> 0) (**a[i]** нечетно, то есть не делится на 2, то есть остаток от деления на 2 не равен 0)

и

(a[i] mod 3=0) (**a[i]** кратно трем, то есть делится на 3).

Небольшая тонкость программы на Паскале состоит в том, что при использовании логических связок каждое элементарное условие, содержащее знаки = > <, приходится заключать в отдельные скобки.

Теперь мы готовы написать нужный фрагмент программы. Чтобы заслужить уважение проверяющего, записать его следует максимально аккуратно и ясно. Речь идет не только о почерке. Регистр букв (заглавные/строчные) следует использовать тот же, что и в программе. Кроме того, следует соблюдать правила отступов, которые подчеркивают подчиненность (вложенность) одних конструкций в другие, или, наоборот, их равноправность.

Напоминаем, что это задание части С, поэтому ответом в данном случае будет программный код, который представлен ниже и который будет проверяться специально выделенными работниками.

```
p := 1;

 for i := 1 to N do
   if (a[i] mod 2<>0) and (a[i] mod 3=0)
     then p := p*a[i];
```

С3 | *Игра в камушки*

? Два игрока, Петя и Ваня, 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, первый ход делает Петя. Ход состоит в том, что игрок или утраивает число камней в какой-либо куче, или

добавляет 2 камня в какую-либо кучу. Игра завершается в тот момент, когда количество камней в одной из куч становится не менее 19. Если в момент завершения игры общее число камней в двух кучах не менее 35, то выиграл Ваня, в противном случае — Петя. Кто выигрывает при безошибочной игре обоих игроков? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

√ Представим следующее собственное решение с обоснованием, которое, впрочем, не сильно отличается от предложенного разработчиками.

Решение, которое предлагается в демо-варианте, нас не удовлетворяет тем, что оно не показывает сам процесс поиска ответа. Лучше всего это делать в виде такой же таблицы, только строк будет больше, чем в «чистовике», ведь мы пока не знаем, кто выигрывает при правильной игре обоих игроков и должны еще найти это.

Обозначим варианты хода, например, так: вариант 1 — утраивать количество камней в первой куче, 2 — утраивать количество камней во второй куче, 3 — добавить два камня в первую кучу, 4 — добавить два камня во вторую кучу.

Оговорим правила записи информации в клетках. Первая цифра — вариант хода, далее в скобках через запятую — количество камней в первой куче и количество камней во второй куче, ПП или ПВ — если достигнут конец игры и выиграл Петя или Ваня соответственно. Пустые клетки слева означают то же самое, что и выше. Пустые клетки справа означают, что игра закончена.

При тех данных, которые представлены в демо-варианте, процесс поиска закончился быстро. Но при других данных он может затянуться, к этому надо быть готовым.

Начальное положение по условию (4,3)

Ход 1 Петя	Ход 2 Ваня	Ход 3 Петя	Ход 4 Ваня	Ход 5 Петя	Ход 6 Ваня	Ход 7 Петя
1(12,3)	1(36,3)ПВ					
2(4,9)	1(12,9)ПП					
	2(4,27)ПП					
	3(6,9)	1(18,9)ПП				
	4(4,11)	1(12,11)ПП				

Мы видим, что если Петя первым ходом выберет тот вариант, который мы обозначили как вариант 2, он побеждает на ходе втором или, максимум, на третьем, независимо от того, какой из четырех вариантов второго хода выберет Ваня.

Верхнюю строку, которая залита серым цветом в таблице, в окончательном ответе приводить не нужно.

Общий принцип — в окончательном чистовом варианте за победителя в каждом его ходе указывается только один вариант, который приводит к победе, за проигравшего — все 4 варианта, чтобы показать, что он проигрывает в каждом случае.

Ниже приводится решение, которое следует представить на «чистовике», то есть бланке решения. Это один из вариантов. Разработчики задания предлагают несколько иной вариант решения.

Оформление решения

Вывод: при правильной игре обеих сторон выигрывает Петя. Для победы он может первым ходом утроить количество камней во второй куче и в зависимости от действий Вани действовать согласно следующей таблице.

Обозначим варианты хода, например, так: вариант 1 — утраивать количество камней в первой куче, 2 — утраивать

количество камней во второй куче, 3 — добавить два камня в первую кучу, 4 — добавить два камня во вторую кучу.

Оговорим правила записи информации в клетках. Первая цифра — вариант хода, далее в скобках через запятую — количество камней в первой куче и количество камней во второй куче, ПП или ПВ — если достигнут конец игры и выиграл Петя или Ваня соответственно. Пустые клетки слева означают то же самое, что и выше. Пустые клетки справа означают, что игра закончена.

Начальное положение по условию (4,3)

Ход 1 Петя	Ход 2 Ваня	Ход 3 Петя	Ход 4 Ваня	Ход 5 Петя	Ход 6 Ваня	Ход 7 Петя
2(4,9)	1(12,9)ПП					
	2(4,27)ПП					
	3(6,9)	1(18,9)ПП				
	4(4,11)	1(12,11)ПП				

◀

√ Чем отличается решение, представленное разработчиками, от приведенного выше?

В решении, представленном разработчиками, в таблице указывается только количество камней в каждой куче, то есть те числа, которые в нашей таблице записаны в скобках. Нет указания на то, каким вариантом хода это сделано. Указание на победу Пети дано в стороне от ячеек. В качестве первого выигрышного хода Пети дан другой, который приводит к победе за большее количество ходов.

В целом же идея та же — *обоснование проводится с помощью таблицы, в которой за победителя указывается каждый раз один вариант хода, а за проигравшего — все возможные варианты.*

Тот вариант, который рекомендуем мы, особенно хорош как раз при поиске решения.

С4 Написать программу

? На вход программе подаются сведения о пассажирах, желающих сдать свой багаж в камеру хранения на заранее известное время до полуночи. В первой строке сообщается число пассажиров N, которое не меньше 3, но не превосходит 1000; во второй строке — количество ячеек в камере хранения M, которое не меньше 10, но не превосходит 1000. Каждая из следующих N строк имеет следующий формат.:

<Фамилия> <время сдачи багажа> <время освобождения ячейки>,

где <Фамилия> — строка, состоящая не более чем из 20 непробельных символов; <время сдачи багажа> — через двоеточие два целых числа, соответствующие часам (от 00 до 23 — ровно 2 символа) и минутам (от 00 до 59 — ровно 2 символа); <время освобождения ячейки> имеет тот же формат.

<Фамилия> и <время сдачи багажа>, а также <время сдачи багажа> и <время освобождения ячейки> разделены одним пробелом. Время освобождения больше времени сдачи.

Сведения отсортированы в порядке времени сдачи багажа. Каждому из пассажиров в камере хранения выделяется свободная ячейка с минимальным номером. Если в момент сдачи багажа свободных ячеек нет, то пассажир уходит, не дожидаясь освобождения одной из них.

Требуется написать программу (укажите используемую версию языка программирования, например *Borland Pascal* 7.0), которая будет выводить на экран для каждого пассажира номер ему предоставленной ячейки (можно сразу после ввода данных очередного пассажира). Если ячейка пассажиру не предоставлена, то его фамилия не печатается.

Пример входных данных:
3
1 0
Иванов 09:45 12:00
Петров 10:00 11:00
Сидоров 12:00 13:12

Результат работы программы на этих входных данных:
Иванов 1
Петров 2
Сидоров 1

√ При решении такой программистской задачи следует задать себе следующие важные вопросы.

Следует ли хранить все исходные данные в том виде, в котором они вводятся? Этот вопрос фактически состоит из двух подвопросов. 1) Возможно ли это вообще при заданном в условии объеме исходных данных? И (если возможно), 2) Будет ли такая программа оптимальной?

При анализе условия можно понять, что если багажная ячейка использовалась, то для определения, освободилась ли она или еще занята, достаточно хранить только время освобождения каждой такой ячейки. Время удобно хранить не в виде двух чисел (часы и минуты), а одним числом — число минут от полуночи. Это удобнее и для сравнения времени. Причем такой способ можно распространить и на ячейки, которые не были задействованы, — будем считать время освобождения нулевым, как будто ячейку освободили еще в полночь.

Ниже приведен код программы на Паскале, рекомендуемый разработчиками, с добавленными мною в начале пояснениями смысла используемых переменных.

▶

{

N — из условия — общее количество пассажиров.

K — из условия — общее количество ячеек в камере хранения (здесь разработчики что-то намудрили — в
условии задачи она обозначена M, ничего не мешало ее и в программе так же обозначить!).

p — массив, который хранит для каждой занятой ячейки время ее освобождения в минутах (отсчет минут начинается от полуночи). Вообще-то используется только первые K элементов массива, но приходится брать по максимуму с учетом условия задачи: количество ячеек в камере хранения M, которое не меньше 10, но не превосходит 1000.

c, c1 — временные переменные для чтения очередного символа из потока входных данных.

i, j — переменные для организации циклов.

name — фамилия очередного пассажира.

time1, time2 — для очередного пассажира время в минутах от полуночи сдачи багажа и время освобождения ячейки.

}

Пример правильной программы на языке Паскаль:

```
var p:array[1..1000] of integer;
    c,c1:char;
    i,j,N,K:integer;
    name:string;
    time1,time2:integer;
begin
```

```
    readln(N,K);
    for i:=1 to K do
      p[i]:=0;
    for i:=1 to N do
    begin
      name:='';
      repeat
        read(c);
        name:=name+c
      until c=' "; {считана фамилия}
      read(c,c1); {считаны часы первого времени}
       time1:=60*((ord(c)-ord("0"))*10+ord(c1)-
ord("0"));
      read(c,c,c1); {пропущено двоеточие,
                     и считаны минуты}
     time1:=time1+(ord(c)-ord("0"))*10+ord(c1)-
ord("0");
      read(c,c,c1); {считаны часы второго
                     времени}
      time2:=60*((ord(c)-ord("0"))*10+ ord(c1)-
ord("0"));
      readln(c,c,c1); {пропущено двоеточие,
                       и считаны минуты}
     time2:=time2+(ord(c)-ord("0"))*10+ord(c1)-
ord("0");
      for j:=1 to K do
        if p[j]<=time1 then
          begin
            p[j]:=time2;
            writeln(name,' ",j);
            break;
          end;
    end;
  end. ◀
```

О задаче С4

Задача С4 задумана разработчиками ЕГЭ как своеобразный фильтр, отделяющий самых-самых знатоков информатики. В самом деле, для ее успешного решения требуется обладать весьма приличными знаниями в программировании, да еще и иметь опыт написания программ. Поэтому-то за нее и дается по максимуму — четыре балла. Это так называемые первичные баллы. Во сколько они превратятся при переводе в 100-бальную шкалу — трудно сказать, организаторы из года в год изменяют правила перевода в 100-бальную шкалу. Но можно взять ориентировочно — поскольку первичных баллов максимально 40, можно очень грубо считать, что следует умножать первичный балл на 2,5. Итак, четыре первичных балла превращаются в десять баллов 100-бальной шкалы! Ради них имеет смысл потратить время на изучение основ программирования. В любом случае дополнительные знания не пропадут — в каждом вузе сегодня изучаются несколько учебных предметов, связанных с компьютерами, да и на любом рабочем месте, если оно связано с умственным трудом, вам они, скорее всего, пригодятся.

Я напоминаю, что рассматриваю все задачи, связанные с программированием, только на примере языка Паскаль. Дублирование информации для всех разрешенных к использованию в ЕГЭ языков слишком бы увеличило объем книги.

Многих учащихся, даже неплохо знающих программирование, ставит с тупик фраза, с которой начинается задание: «На вход программе подаются...». Объясним, что имеется в виду.

Стандартный Паскаль предполагает, что данные могут подаваться программе со стандартного устройства ввода. Такой подход использовался в эпоху «до персональных компьютеров», когда данные могли вводиться с клавиатуры

на терминале большого компьютера или с перфокарты или с какого-то другого устройства ввода. Программе, по большому счету, неважно, с какого устройства поступают данные. Главное — научиться их правильно считывать.

Можно представлять себе, что входные данные — это строки текста. В конце строки размещается специальная метка конца строки. Текстовые редакторы эту метку не отображают, но в позиции метки делают переход на новую строку.

Для чтения входных данных в Паскале используется оператор *READ*( ), где в скобках записываются одна или несколько переменных, в которые считывается информация. Ситуация сильно зависит от типа переменных, записанных в скобках.

1. В скобках записана переменная строкового типа, то есть программа имеет примерно такой вид:

```
Var st : string;
...
Read(st);
```

Мы приводим здесь только те строки кода, которые нам важны для пояснения.

В этом случае в переменную st считывается вся строка входных данных до признака конца строки.

Если таких переменных несколько, потребуется столько строк, сколько переменных. Например, при выполнении кода

```
Read(st1,st2);
Read(st3,st4);
```

в st1 будет считана первая строка входных данных, в st2 — вторая строка входных данных, в st3 — третья, и конечно, в st4 — четвертая. Разумеется, входные данные должны содержать необходимое количество строк.

2. В скобках записана переменная числового типа, то есть программа имеет примерно такой вид:

```
Var a : integer;
...
Read(a);
```

В этом случае в переменную **a** будет считано целое число от текущей позиции до границы числа. Границей числа может быть пробел или признак конца строки.

Например, если входные данные выглядят так:

234 –45 привет

А оператор их чтения выглядит так: read(a,b);, то в переменную **a** будет считано число 234, а в переменную **b** — число –45 и указатель чтения будет после этого находиться после пробела перед символом **п** Поэтому для такой же строки входных данных в случае при выполнении оператора *read*(a,b, st); в st будет считано слово **привет.**

Такая же ситуация в том случае, если переменные не целочисленные, а имеют тип вещественных чисел *real*. В этом случае просто разрешаются еще и дробные числа с разделителем дробной части — точкой. Понятно, что если текст нельзя трактовать как число, то произойдет так называемая «ошибка времени выполнения».

3. Еще один случай — использование переменной типа *char*.

```
Var ch : char;
...
Read( Ch );
```

Тип *Char* служит для хранения ровно одного символа, поэтому в этом случае в Ch будет считан ровно один

символ. Повторяя такое чтение много раз, можно добиться нужного эффекта.

Приведем пример. Пусть каждая строка входных данных содержит информацию об очередном пассажире: фамилию, имя, отчество, место, вес багажа, название документа, разделенные каждый раз одним пробелом, например:

Петров Петр Петрович 47 21.5 паспорт

Прочитать такие данные в соответствующие переменные можно, например, так:

```
Var Ch : char;
    Fam, Im, Ot : string,
    Mesto : integer;
    Ves : real;
    Document : string;
...
Fam := '';
Repeat
   Read(Ch); if Ch<>' ' then Fam ;= Fam+Ch;
Until Ch=' ';

Im := '';
Repeat
   Read(Ch); if Ch<>' ' then  Im := Im+Ch;
Until Ch=' ';

Ot := '';
Repeat
   Read(Ch); if Ch<>' ' then Ot := Ot+Ch;
Until Ch=' ';

Readln( Mesto, Ves, Document);
```

Обратите внимание — нельзя написать просто

Readln(Fam, Im, Ot, Mesto, Ves, Document);,

так как при этом вся строка будет считана в переменную **Fam**, и для других переменных во входных данных «ничего не останется».

Приходится в цикле считывать по одному символу в переменную **Ch** и присоединять этот считанный символ к фамилии, а в следующий раз к имени или отчеству. Границей фамилии, имени, отчества является пробел, поэтому, когда он встретится, цикл заполнения данной переменной заканчивается, и переходим к заполнению следующей.

А вот в конце удается считать **Mesto, Ves, Document** в одном операторе. Первые две переменные — числовые, и для них пробелы в строке входных данных — естественные границы, а переменная **Document** имеет тип **string** и в нее считается все до конца строки, что и требуется.

Приведем для примера еще одну задачу С4 (на самом деле это задача из демо-варианта 2012).

С4 | *Написать программу*

? В командных олимпиадах по программированию для решения предлагается не больше 11 задач. Команда может решать предложенные задачи в любом порядке. Подготовленные решения команда посылает в единую проверяющую систему соревнований. Вам предлагается написать эффективную, в том числе по используемой памяти, программу, которая будет статистически обрабатывать пришедшие запросы, чтобы определить наиболее популярные задачи. Следует учитывать, что

количество запросов в списке может быть очень велико, так как многие соревнования проходят с использованием Интернет. Перед текстом программы кратко опишите используемый вами алгоритм решения задачи.

На вход программе в первой строке 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.

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 упоминается менее трех задач, то выведите информацию об имеющихся задачах. Если несколько задач имеют ту же частоту встречаемости, что и третья по частоте встречаемости задача, их тоже нужно вывести.

Пример выходных данных для приведенного выше примера входных данных.

A+B 2
Простой делитель 2
Крестики-Нолики 1
Прямоугольник 1

√ При решении такой программистской задачи следует задать себе следующие важные вопросы.

Следует ли хранить все исходные данные в том виде, в котором они вводятся? Этот вопрос фактически состоит из двух подвопросов: 1) Возможно ли это? И (если возможно), 2) Будет ли такая программа оптимальной?

В тексте сказано, что число строк во входных данных может быть очень большим. Это означает, что сохранить входные данные в их начальном исходном виде не удастся.

Так, входные данные следует преобразовывать и хранить в «переработанном» виде.

Сами выходные данные задачи подсказывают, что следует хранить после прочтения данных. Нужно хранить название задачи и количество запросов к ней. Для таких случаев подходящей структурой хранения был бы динамически расширяющийся список. Однако в «простых» школьных программах обычно обходятся конструкцией, состоящей из массива с запасом и переменной, которая хранит количество элементов массива, заполненных полезной информацией. Точнее, в данном случае приходится использовать два «параллельных» массива. Один массив для хранения названий задач, второй — количества запросов к соответствующей задаче. В каждом массиве следует заготовить место для 11 элементов — это максимальное количество задач. Названия задач — не более 100 символов. Поэтому общий размер хранимых данных будет вполне приемлем.

После того, как входные данные будут вычитаны, по условию задачи следует взять из массивов три, где значения в массиве *Count* максимальные, но с учетом оговорок, сделанных в условии задачи. Оптимальный способ — отсортировать оба массива параллельно, используя в качестве признака сортировки значения массива *Count* — по убыванию. После этого вывести первые три из них — с учетом оговорок, сделанных в условиях задачи.

Ниже приведен код программы на Паскале, рекомендуемый разработчиками, с добавленными в начале пояснениями смысла переменных.

▶

```
{
N — из условия — общее количество запросов
I, j — переменные для организации циклов
Names — массив для хранения названий задач
Count — массив для хранения количества обращений к каждой задаче
Num — сколько начальных элементов уже заполнено в этих массивах
S — временная переменная для организации обменов значений в массиве Names при сортировке
t — временная переменная для организации обменов значений в массиве Count при сортировке
}
Var N, Num, i, j, t: integer;
  s: string;
  Names: array[1..11] of string;
  Count: array[1..11] of integer;
Begin
  Num:=0; { Число различных задач
                    в списке запросов }
  ReadLn(N); { Считываем количество запросов }
  for i:=1 to N do
  begin
    ReadLn(s); { считали очередную задачу }
  { Осуществляем ее поиск в
              списке уже встретившихся }
    j:=1;
   while (j<=Num) and (s<>Names[j]) do j:=j+1;
   { Если она найдена }
```

```
      if j<=Num then { Увеличиваем  счетчик
                          числа  запросов }
        Count[j]:=Count[j]+1
      else begin { Иначе  добавляем
                     задачу  в  конец  списка }
            Names[j]:=s;
            Count[j]:=1;
            Num:=Num+1
          end
    end;
  { Сортируем  массивы  Names и  Count в порядке
убывания  значений массива  Count}
    for i:=Num downto 2 do
    for j:=2 to i do if Count[j-1]<Count[j] then
    begin
      t:=Count[j]; Count[j]:=Count[j-1];
     Count[j-1]:=t;
      s:=Names[j]; Names[j]:=Names[j-1];
      Names[j-1]:=s;
    end;
    if Num >= 3 then j := 3 else j := Num;
    i := 1;
    while (i <= Num) and (Count[i] >= Count[j]) do
    begin
      WriteLn(Names[i],' ", Count[i]);
      i := i + 1;
    end;
  end.
```

◀

√ Заметим, что используется сортировка обменами (пузырьковая сортировка). Как показывает опыт, она наиболее доступна для учащихся. Точнее, используется некоторая ее вариация — ведь сортируются одновременно два «параллельных массива». Так я их называю потому,

что они относятся к одним и тем же вещам (сущностям). **Names[j]** — это название очередной задачи, а **Count[j]** — количество обращений к этой задаче. И если **Count[j]** перемещаем в другую позицию в массиве, то следует в такую позицию переместить и **Names[j]**. Порядок сортировки для обоих массивов определяет массив *Count*.

Замечу еще, что для сортировки не обязательно использовать именно тот вариант, который используется в приведенном выше коде. Возможно, учитель информатики обучил вас несколько иному варианту той же пузырьковой сортировки, или, возможно, вам больше запомнился другой метод сортировки.

Серия
«Абитуриент»

Чупин Николай Александрович

ПОДГОТОВКА К ЕГЭ ПО ИНФОРМАТИКЕ. ОПТИМАЛЬНЫЕ СПОСОБЫ ВЫПОЛНЕНИЯ ЗАДАНИЙ

Ответственный за выпуск *Кузнецов В.*
Технический редактор *Логвинова Г.*
Верстка *Патулова А.*

ано в набор 25.10.2012 г. Подписано в печать 25.11.2012 г.
Формат 84x108 $^{1}/_{32}$. Бумага типографская.
Гарнитура Pragmatica.
Тираж 2 500. Заказ № 681.

ООО «Феникс»
344082, г. Ростов-на-Дону, пер. Халтуринский, 80.
Тел./факс: (863) 261-89-50, 261-89-59
Сайт издательства: www.phoenixrostov.ru
Интернет-магазин: www.phoenixbooks.ru

Отпечатано с готовых диапозитивов в ЗАО «Книга».
344019, г. Ростов-на-Дону, ул. Советская, 57.
Качество печати соответствует предоставленным диапозитивам.