РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ ИММАНУИЛА КАНТА

С. В. Мациевский

ИНФОРМАТИКА КАК РЕШЕНИЕ ЗАДАЧ ЕГЭ

Учебное пособие

Издательство
Российского государственного университета им. И. Канта
2009

УДК 681.3.06(075.3)
ББК 32.973.26я721
М 367

Мациевский С. В.

М 367 Информатика как решение задач ЕГЭ: учебное пособие.— Калининград: Изд-во РГУ им. И. Канта, 2009.— 419 с.: ил., табл.

ISBN 978-5-88874-943-2

Автор — председатель комиссии по проверке решений заданий по ЕГЭ по информатике в Калининградской области.

Данное издание призвано решить две задачи: подготовить учащегося к сдаче ЕГЭ по информатике и обучить его основам информатики. Оно отличается от аналогичных изданий расположением материала не по заданиям ЕГЭ и не по программе по информатике, а по методам решения заданий.

Показано, как нужно подходить к решению заданий.

Материал книги включает все официальные демонстрационные задания ЕГЭ за четыре года (2006—2009) и теоретический материал, необходимый для их решения, причем этот материал приведен в полном объеме, а не ограничен теми частями, по которым составлены задания ЕГЭ.

Содержание издания покрывает теоретическую часть государственной программы по информатике для школ и гуманитарных направлений вузов и ссузов.

Книга предназначена как для подготовки учащихся 11-х классов к сдаче ЕГЭ по информатике, так и для обучения информатике школьников и учащихся гуманитарных направлений вузов и ссузов.

Замечания и предложения направляйте по электронному адресу автора matsievsky@newmail.ru.

УДК 681.3.06(075.3)
ББК 32.973.26я721

ISBN 978-5-88874-943-2

Оглавление

Роль учителя в том, чтобы сделать себя лишним для ученика.
Идрис Шах. Учиться как учиться

Идея принципиальной важности состоит в том, *что* мы включаем в наше изучение и *что* из него исключаем.
Идрис Шах. Знать как знать

Предисловие

Данное учебное пособие направлено на подготовку к ЕГЭ по информатике.

Данное учебное пособие существенно отличается от всех других стандартных пособий по подготовке к ЕГЭ по информатике по четырем следующим причинам.

1. Задачи сгруппированы не по темам информатики и не по разделам ЕГЭ, а по методам их решения.

2. Приведены полноценные теоретические сведения как по разделам информатики, так и по методам решения задач в виде алгоритмов, которые необходимы для решения представленных заданий.

3. Методы решения некоторых задач, предложенные в других изданиях, существенно упрощены. На самом деле для решения предложенных заданий не требуются ни логарифмы, ни преобразования логических формул. Складывается впечатление, что в обычных пособиях не решения составляются по задачам, а задачи, иногда неудачно, подбираются под нужную тему. Поскольку при решении заданий частей 1 и 2 (A и B) необходимо указать только ответы, такая позиция авторов непонятна.

4. Для решения всех собранных в этом пособии задач, *кроме задач из последнего параграфа (глава 3, § 3)*, не нужно разбираться в компьютерном программировании. Поэтому те учащиеся, которые слабо или совсем не разбираются в компьютерном программировании, могут опустить § 3 главы 3. В нем содержатся три из четырех задач из третьей части С на построение компьютерных алгоритмов (в предыдущих параграфах разобрана только задача на перебор вариантов при игре двух игроков С3, а также начало задачи С1). Для решения этих трех заданий необходимо знание практического программирования на каком-нибудь алгоритмическом языке.

В данном учебном пособии приведены только задания из демонстрационных вариантов ЕГЭ по информатике за четыре года с 2006 по 2009, открыто и общедоступно размещенные на официальном сайте Федерального института педагогических измерений www.fipi.ru.

В заголовках каждого задания указаны год демонстрационного варианта, откуда взято задание, буква части (А, В или С) и номер задания.

Если не рассматривать последний параграф пособия, то учащийся сможет научиться решать все задания частей 1 и 2 (А и В), а из части 3 (С) — полностью только одно задание С3, за которое можно получить три балла. Другое задание С1 из части 3 (С) сможет решить лишь частично, что позволит ему набрать еще один балл.

Таким образом, при решении всех заданий, разобранных в данном учебном пособии, кроме заданий на практическое программирование из последнего параграфа, можно набрать 32 балла, что позволяет учащемуся получить итоговую отметку «отлично». Но при потере даже одного балла отметка уже будет «хорошо», как следует из таблицы, приведенной во введении.

Замечания и пожелания можно направлять по электронному адресу matsievsky@newmail.ru.

Часть учебы, которую люди ценят,— это именно та часть, которая не приносит им никакой пользы, подобно сладостям — масса удовольствия, но не заменит еды.

Именно из-за того, что ученик столь низкого качества, учителю приходится повторять, расширять и увеличивать в размере то, что иначе ученик не заметил бы вовсе.

Идрис Шах. Наблюдения за покровом

Методические указания

Поскольку материалы книги могут быть использованы и для самостоятельного освоения грамотности, уместно сначала разметить непродолжительные методические указания.

Приведем несколько общих рекомендаций по выполнению заданий ЕГЭ по информатике. Эти рекомендации могут принимать различные формы для различных тем.

1. Работа с текстом задания.
2. Выбор варианта решения.
3. Изучение вариантов ответов.

В тексте выделены все определения и алгоритмы.

Определения для экономии места выделены чертой справа. Кроме того, определяемое понятие выделено курсивом. Определения не нумеруются.

Алгоритмы имеют выделенный заголовок. Описание алгоритма заканчивается горизонтальной чертой. Алгоритмы нумеруются.

Нумерация в тексте сплошная: рисунки, таблицы и алгоритмы нумеруются подряд одной сплошной нумерацией. В каждом разделе своя нумерация.

Данное учебное пособие состоит из девяти параграфов, объединенных в три главы. Дадим некоторые рекомендации о порядке работы с этими параграфами.

Каждый параграф посвящен одной теме по методике решения заданий ЕГЭ по информатике и имеет одну и ту же структуру. Каждый параграф желательно изучать целиком, сразу от начала и до конца.

1. В первом разделе «Теория» каждого параграфа приведены достаточно подробные теоретические сведения по теме параграфа и тем заданиям, которые вошли в данный параграф. Поэтому работу с параграфом следует начинать с изучения этого теоретического материала.

2. Первый раздел каждого параграфа может заканчиваться пунктом с упражнениями. Желательно выполнить все упражнения после изучения теоретического материала.

3. Второй раздел «Алгоритмы» носит вспомогательный характер. В нем собраны основные технологии, которые будут использоваться при решении заданий данного параграфа.

4. В третьем разделе «Задачи» собраны задания ЕГЭ по информатике, которые решаются по алгоритмам, отвечающим данному параграфу. Однотипные задачи собраны в отельные пункты.

5. Четвертый раздел «Ответы» носит вспомогательный характер и содержит ответы на задания из третьего раздела.

6. Наконец, в пятом разделе «Решения» приводятся ответы и решения задач из третьего раздела. Если задача решается аналогично предыдущей, то ее решение не приводится.

В каком же порядке лучше проходить параграфы? С одной стороны, параграфы расположены в некотором естественном порядке трех глав, в которые они объединены: «Числа», «Логика» и «Алгоритмы».

С другой стороны, желательно как можно раньше начать работать с главой «Алгоритмы» для возможно более раннего формирования у учащихся алгоритмического мышления. В то же время материалы одних параграфов могут использоваться в последующих параграфах.

Поэтому автор рекомендует проходить девять параграфов в следующем порядке, причем изучение последнего параграфа, третьего в третьей главе, можно опустить либо из-за нехватки времени, либо по причине очень слабой алгоритмической подготовленности учащихся.

1. Глава 2 «Логика», § 3 «Поиск всех вариантов».
2. Глава 3 «Алгоритмы», § 1 «Управление исполнителем».
3. Глава 1 «Числа», § 3 «Электронная таблица».
4. Глава 2 «Логика», § 1 «Логические операции».
5. Глава 3 «Алгоритмы», § 2 «Выполнение алгоритмов».
6. Глава 2 «Логика», § 2 «Поиск закономерностей».
7. Глава 3 «Алгоритмы», § 3 «Составление алгоритмов».
8. Глава 1 «Числа», § 1 «Измерения количества информации».
9. Глава 1 «Числа», § 2 «Перевод чисел из одной системы счисления в другую».

Упражнения призваны расширить практическую базу учащихся по решению задач по основным положениям теории, в них включены достаточно простые задачи, которые не вошли в задания по ЕГЭ.

Упражнения используются для более глубокого проникновения в теорию.

Кроме того, в связи со сложностью и объемностью материала, представляемого в третьей главе «Алгоритмы», более сложные задания по ЕГЭ в этой главе также являются частью теории, дополняя ее на примерах. Применимость такого подхода обусловлена тем, что объяснение нового материала на примерах крайне эффективна в теории алгоритмов.

Задания по ЕГЭ, решение которых не приведено в данном пособии и которые решаются аналогично решенным заданиям, рекомендуются для тренинга учащихся.

Введение
Методическое письмо

Письмо подготовлено
членами федеральной предметной комиссии по информатике
канд. пед. наук В. Р. Лещинером, канд. пед. наук П. А. Якушкиным
на основе аналитического отчета «Результаты единого государственного экзамена 2008 года», размещенного на сайте ФИПИ
(http://www.fipi.ru). Письмо согласовано
с председателем научно-методического совета ФИПИ по информатике,
д-ром физ-мат. наук, профессором Л. Н. Королевым,
утверждено директором ФИПИ А. Г. Ершовым.

Методическое письмо Об использовании результатов единого государственного экзамена 2008 года в преподавании информатики и ИКТ в образовательных учреждениях среднего (полного) общего образования

Цель единого государственного экзамена по информатике и ИКТ — оценка общеобразовательной подготовки по информатике выпускников XI (XII) классов общеобразовательных учреждений и абитуриентов с целью отбора для зачисления в учреждения высшего и среднего специального профессионального образования.

ЕГЭ представляет собой экзамен с использованием заданий стандартизированной формы — контрольных измерительных материалов (далее — КИМ), выполнение которых позволяет установить уровень освоения участниками ЕГЭ федерального государственного стандарта среднего (полного) общего образования.

Содержание экзаменационной работы по информатике определялось на основе утвержденного Министерством образования Российской Федерации обязательного минимума содержания среднего (полного) общего образования по информатике (Приказ от 30.06.99 №56) с учетом тенденций развития предмета, заложенных в образовательном стандарте 2004 г.

За 4 года ЕГЭ по информатике сформировался как профильный экзамен, сдаваемый преимущественно абитуриентами вузов и ссузов.

Характеристика контрольных измерительных материалов ЕГЭ по информатике

Структура экзаменационной работы по информатике была определена еще в 2006 г. и с тех пор не изменялась. Работа состояла из трех частей. Часть 1 содержала 20 заданий из всех тематических блоков содержания предмета, кроме заданий по технологии телекоммуникаций и технологии программирования. Задания части 1 предполагали выбор одного ответа из четырех предложенных. Часть 2 включала задания по темам: «Информация и её кодирование», «Основы логики», «Алгоритмизация и программирование», «Телекоммуникационные технологии» — всего 8 заданий с кратким ответом. Задания части 3 были направлены на проверку сформированности важнейших умений записи и анализа алгоритмов, предусмотренных требованиями к обязательному уровню подготовки по информатике учащихся общеобразовательных учреждений. В этой части также проверялись умения на повышенном и высоком уровне сложности по теме «Технология программирования». Решения заданий третьей части работы записывались в развернутой форме и проверялись экспертами региональных предметных комиссий. За выполнение каждого задания давалось определенное количество баллов, в зависимости от полноты и качества выполнения. Так, часть 3 включает 4 задания, что составляет 12,5% от общего количества заданий. При успешном их выполнении экзаменуемый может получить максимально 12 первичных баллов (т. е. 30% общего количества первичных баллов за всю работу). С другой стороны, эти задания были самыми сложными и самыми трудоемкими: рекомендованное время их выполнения в два раза превосходило время, отводимое на выполнение первых двух частей работы.

Содержание экзамена включало основные темы курса информатики и информационных технологий, объединенных в

следующие тематические блоки: «Информация и её кодирование», «Алгоритмизация и программирование», «Основы логики», «Моделирование и компьютерный эксперимент», «Программные средства информационных и коммуникационных технологий», «Технология обработки графической и звуковой информации», «Технология обработки информации в электронных таблицах», «Технология хранения, поиска и сортировки информации в базах данных», «Телекоммуникационные технологии».

Важной проблемой при разработке контрольных измерительных материалов для ЕГЭ, является поиск оптимального способа проверки знаний и определения уровня сложности КИМ. Необходимо определить, какова доля заданий на простое воспроизведение материала, в какой ситуации проверяется умение применять полученные знания. В КИМ по информатике сознательно не включены задания, требующие простого воспроизведения знания терминов, понятий, величин, правил (такие задания слишком просты для выполнения, кроме того, при использовании заданий на простое воспроизведение очень трудно обеспечить параллельность вариантов). При выполнении любого из заданий КИМ от экзаменуемого требуется решить какую-либо задачу: либо прямо использовать известное правило, алгоритм, умение, либо выбрать из общего количества изученных понятий и алгоритмов наиболее подходящее и применить его в известной либо новой ситуации. Таким образом, на уровне *воспроизведения знаний* (6 заданий) выполняются несложные задания в одно-два действия, требующие формального следования изученному алгоритму или правилу. Таковы задания на перевод чисел из одной системы счисления в другую, построение таблицы истинности для заданного выражения, установление соответствия между различными типами моделей, чтение и формальное исполнение блок-схем, формирование URL по описанию адреса документа и т.п. Задания на воспроизведение знаний входят в первую и вторую часть работы.

Материал на проверку сформированности *умений применять свои знания в стандартной ситуации,* входящий во все три части экзаменационной работы, предполагает использование комбинации правил или алгоритмов, совершение последовательных действий, однозначно приводящих к верному результату. Предполагается, что экзаменуемые в процессе изучения школьного курса информатики приобрели достаточный опыт в решении подобных задач. К числу заданий такого рода относятся задания на подсчет информационного объёма сообщения (применяются правила вероятностного и алфавитного подсчета объемов сообщений, требуется правильно выбрать единицу измерения объема информации, перевести все данные в эти единицы). К этой же группе примыкают задания на осуществление арифметических действий в двоичной, восьмеричной и шестнадцатеричной системах счисления (требуется применить комбинацию алгоритмов вычисления «столбиком» для двоичной либо кратных систем счисления и алгоритмов перевода из двоичной в восьмеричную (по триадам) и в шестнадцатеричную (по тетрадам) системы счисления); а также задания на создание и преобразование логических выражений и так далее. К этому типу относится одно из заданий третьей части работы, требующее формальной записи изученного в школе алгоритма обработки массива на языке программирования либо естественном языке. Это наиболее часто встречающийся в экзаменационной работе тип заданий, общее их количество заданий и доля от максимального первичного балла составляет чуть менее половины от соответствующих показателей за всю работу.

Задания на проверку сформированности *умений применять полученные знания в новой ситуации,* входящие во вторую и третью часть работы, предполагают решение экзаменуемыми своеобразной творческой задачи: какие изученные правила и алгоритмы следует применить, в какой последовательности это необходимо сделать, какие данные использовать. К данному типу относятся текстовые логические задачи, задания на поиск и устранение ошибок в алгоритмах, на написание программ.

О распределении в экзаменационной работе заданий по уровням сложности можно судить по данным таблицы 1.

Таблица 1

Распределение заданий по уровням сложности

Уровень сложности заданий	Число заданий	Максимальный первичный балл	Процент максимального первичного балла за задания данного вида деятельности от максимального первичного балла за всю работу (40)
Базовый	16	16	40
Повышенный	12	14	35
Высокий	4	10	25
Итого:	32	40	100

Таким образом, экзамен проверяет знания и умения выпускников на различных уровнях. Базовый уровень представляет собой задания на проверку знаний и умений инвариантной составляющей курса информатики, преподающегося в классах и учебных заведениях всех профилей. Такие задания составляют 50% от всех заданий экзаменационной работы. Задания повышенного уровня были связаны с содержанием профильных курсов информатики, требующих более углубленного изучения. Задания высокого уровня призваны выделить учащихся, хорошо овладевших содержанием учебного предмета, ориентированных на получение высшего профессионального образования в областях, связанных с информатикой и компьютерной техникой.

Характеристика участников ЕГЭ по информатике 2008 года

В 2008 г. число регионов, принявших участие в ЕГЭ по информатике, выросло почти в три раза по сравнению с прошлым годом, при этом количество самих участников возросло почти в четыре раза: в нем участвовало 10347 выпускников из 36 регионов (2007 г.— 2694 участников из 13 регионов, 2006 г.— только г. Санкт-Петербург).

71,3% экзаменуемых по информатике 2008 года — юноши. Данный факт свидетельствует о том, что ЕГЭ по информатике приобрел характер профильного экзамена, который выбирают выпускники, собирающиеся поступать в вузы на специальности, связанные с информационными технологиями, традиционно считающиеся «мужскими».

Среди участников ЕГЭ доминируют выпускники общеобразовательных учреждений: школ, гимназий, лицеев. В то же время статистика этого года позволяет говорить о разнообразии типов образовательных учреждений, в которых обучались участники экзамена.

Существенные изменения произошли в 2008 г. в распределении участников по типам населенных пунктов. Резко сократилась доля больших городов, увеличились доли населенных пунктов меньшего размера. Это связано с тем, что экзамен приобрел по-настоящему всероссийский характер. В 2008 г. около трети участников экзамена проживало в сельской местности (11%), рабочих поселках (7,5%) или малых городах (13,9%). Для этих учащихся возможность сдать профильный экзамен, открывающий путь к высшему образованию в области современных компьютерных технологий, в пределах своего района представляется очень значимой.

Основные результаты экзамена по информатике 2008 года

Результаты экзамена 2008 года в целом соответствуют итогам экзамена 2007 г., хотя заметен некоторый сдвиг в сторону понижения результатов. При пересчете в пятибалльную шкалу практически совпадает доля учащихся, получивших отметку «4» (40% экзаменовавшихся, в 2007 г.— 39,9%). Доля отличников сократилась до 12% (в 2007 г.— 19,4%), доля учащихся, получивших неудовлетворительную оценку, возросла с 9,7% в 2007 г. до 11,3% в 2008 г., доля троечников также возросла до 36,8% (в 2007 г.— 31,2%). Снижение результатов экзамена, выразившееся в увеличении доли учащихся, получивших низкие отметки, проявляется также и в том, что пороги отметки «3» и «4» были в 2008 г. снижены на один первичный балл. Распре-

деление участников экзамена по уровням подготовки показано в таблице 2. Высший балл (40 первичных, 100 тестовых) получили в 2008 г. 25 человек (0,24% всех экзаменуемых), в 2007 г. таких было 12 человек (0,45%).

Таблица 2

Распределение участников экзамена (процент от общего числа) по уровням подготовки

Отметка	Интервал первичного балла		Процент экзаменуемых	
	ЕГЭ 2007	**ЕГЭ** 2008	**ЕГЭ** 2007	**ЕГЭ** 2008
2	0—11	0—10	9,7	11,27
3	12—22	11—21	31,2	38,76
4	23—31	22—31	39,8	37,95
5	32—40	32—40	19,4	12,02

В 2006 году июньский экзамен проходил только в одном городе — Санкт-Петербурге, что обусловило высокий уровень результатов, особенно в условиях профильного экзамена, в котором принимали участие преимущественно выпускники гимназий, лицеев и школ с углубленным изучением предмета. В 2007 г. и 2008 г. экзамен становится более массовым, среди участников 2008 года больше, чем в прошлом году, выпускников, проживающих в сельской местности и в малых городах. Данные факторы сказываются на результатах экзамена в сторону небольшого снижения результатов.

Результаты выполнения экзаменационной работы по темам (разделам) курса

На проверку знаний и умений по разделу **«Информация и ее кодирование»** было отведено 8 заданий, из которых шесть — с выбором ответа и два — с кратким ответом. Пять заданий относятся к базовому уровню сложности, три — к повышенному. Средний процент выполнения колеблется от 84% (задание А13 базового уровня проверяет умение кодировать и

декодировать информацию) до 42% (задание А3 повышенного уровня сложности проверяет умение подсчитывать информационный объем сообщения). Помимо задания А13 не вызвали серьезных затруднений следующие задания: А1, проверяющее знание принципов кодирования текста; А4 на знание двоичной системы (средний процент выполнения — от 63% до 80% в зависимости от варианта). Из заданий базового уровня сложности особые затруднения у экзаменуемых вызвало задание А5, проверяющее умение выполнять арифметические операции в двоичной и кратных системах счисления. Задание было отнесено к базовому уровню, но в ряде вариантов процент выполнения был близок к 50%.

Задания повышенного уровня сложности в среднем выполнило от 33% до 68% экзаменуемых (в 2007 г. от 55% до 70%). Наиболее сложным оказалось задание А3, проверяющее умение подсчитывать информационный объем сообщения путем последовательного определения количества бит, необходимого для записи одного сигнала, умножения полученного числа на число записываемых сигналов, а затем перевод результата в требуемые единицы (байты). Это задание, в зависимости от варианта, правильно выполнило от 33% до 50% экзаменуемых (в 2007 г.— 56%).

В 2007 году наибольшее затруднение вызвало задание на знание математических основ записи чисел в позиционных системах счисления В1, которое предполагает применение знаний в новой ситуации. В 2008 г. это задание верно выполнили от 35% до 68% экз. Наиболее часто встречающейся ошибкой является перечисление не всех чисел, отвечающих заданным требованиям. Задание в обобщенном виде формулируется следующим образом: перечислите в порядке возрастания все натуральные числа не больше n, которые при записи в системе счисления с основанием m заканчиваются на k.

По-прежнему наибольшие затруднения вызывает задание В5 на определение пропускной способности канала связи: результат 2008 г. (57% правильных ответов) оказался выше 2006 г. (46%), но ниже результата 2007 г.(63%).

Несмотря на то, что 2008 году проявилось некоторое снижение результатов по сравнению с 2007 г. в целом выполнение заданий по этой теме можно признать удовлетворительным.

Раздел **«Алгоритмизация и программирование»** был представлен в экзаменационной работе наиболее подробно: в общей сложности 9 заданий базового, повышенного и высокого уровня сложности во всех трех разделах работы. Знания и умения, связанные с использованием основных алгоритмических конструкций, выявлялись как заданием на исполнение и анализ отдельных алгоритмов, записанных в виде блок-схемы, на алгоритмическом языке или на языках программирования, так и заданиями на составление алгоритмов для конкретного исполнителя (задание с кратким ответом) и анализ дерева игры.

Экзаменуемые отлично справились с заданием А14 базового уровня сложности на воспроизведение знаний и умений, проверяющее умение исполнить алгоритм, записанный на естественном языке. Средний процент его выполнения в 2008 г. составил 85% при 83% в 2007 г. Традиционно хорошо выполняется задание А6 на анализ и исполнение алгоритма, записанного в виде блок-схемы (средний процент успешного выполнения однотипных заданий в 2008 г.: 80%; в 2007 г.— 83%, в 2006 — 77%). Задание В3 на запись фрагмента алгоритма для исполнителя с фиксированным набором команд выполнили в среднем 74% экзаменуемых 2008 г. (85% в 2007 г., 87% — в 2006 г.). Некоторое снижение результатов вызвано сознательным усложнением задания при разработке КИМ в 2008 г. Задание А7 на использование переменных также не вызвало затруднений — средний процент выполнения в 2008 г. составил 75% при 82% выполнения в 2007 г. Задание повышенного уровня А8, проверяющее знание алгоритмов работы с массивами, в части вариантов было в 2008 г. сформулировано по-новому, в результате чего процент выполнения колеблется по вариантам от 33% (новые формулировки) до 65% (задания прошлых лет). В 2007 г. это задание правильно выполнили 64% экзаменовавшихся при 66% в 2006 г. и 49% в 2005 г. Тот же результат проявился и при выполнении задания повышенного

уровня сложности А20 (оно было отнесено к типу заданий на применение знаний в новой ситуации). В следующем году подобное задание будет отнесено к категории высокого уровня. Это довольно сложное задание, которое можно выполнить либо трудоемким и затратным по времени способом, либо потратив время на анализ условия с целью поиска решения достаточно коротким путем. Задание А20 при общем невысоком уровне процента выполнения (в среднем 28%) хорошо дифференцирует учащихся по уровню их подготовки.

Задание В6, проверяющее умение исполнить алгоритм, записанный на естественном языке, в 2008 году было также усложнено по сравнению с 2007 г., что привело к снижению результата до 34% в среднем при 61% выполнения в 2007 г. и 55% в 2006 г.

Два задания высокого уровня сложности с развернутым ответом оказались выполнены в 2008 г. лучше, чем в 2007: 33% экзаменуемых получили высший балл (в среднем по вариантам) выполняя задания на запись алгоритма на естественном языке или языке программирования (при среднем проценте выполнения 28% в 2007 г.). 43% экз. в среднем справились с заданием на анализ дерева игры (в 2007 г. 36%). Высший бал в этом году получили 27% выпускников, приступивших к этой задаче, что является очень хорошим результатом для задачи высокого уровня сложности. Результат выполнения этих двух заданий с развернутым ответом показывает, что в текущем году учителя школ уделили значительное внимание подготовке выпускников к решению этих задач.

В целом выполнение заданий этого раздела экзаменационной работы показало хорошее знание экзаменуемыми темы, что объясняется центральным положением данной темы в школьном курсе информатики и методически отработанным за долгие годы развития предмета содержанием обучения. Следует отметить позитивные результаты целенаправленной подготовки абитуриентов к ЕГЭ по информатике с использованием опубликованных вариантов КИМ и учебно-тренировочных материалов.

По разделу **«Основы логики»** в экзаменационной работе содержалось пять заданий: три с выбором ответа и два с кратким ответом. Два задания базового, два повышенного и одно – высокого уровня сложности. Экзаменуемые хорошо справились с заданием А11 базового уровня на проверку умения строить таблицы истинности и логические схемы: 79% выполнения в среднем (результат практически эквивалентен 2006 и 2007 годам) а также с заданием А10 базового уровня на преобразование логических выражений: 83% выполнения в среднем при 79% в 2007 г. и 73% в 2006 г. Результат выполнения задания А9 повышенного уровня на проверку знания основных понятий и законов математической логики также выше результатов прошлых лет: 74% при 57% в 2007 г. и 69% в 2006 г.

Как и в прошлые годы задание В2 на решение логического уравнения дало результат не соответствующий высокому уровню сложности задания, в среднем 49% при 51% в 2007 г. Задание В4 повышенного уровня с кратким ответом представляет собой текстовую логическую задачу. В этом году результат оказался ниже прошлых лет: 52% при 64% в 2007 г. и 57% в 2006 г. В целом в 2008 году по теме «основы логики» результаты полностью соответствуют и иногда даже превосходят результаты, прогнозировавшиеся комиссией. Можно сделать окончательный вывод о том, что повышенное внимание, уделенное этому разделу при разборе результатов ЕГЭ предыдущих лет, дало свои плоды: результат усвоения этой темы не выбивается из общего ряда.

По теме **«Моделирование»** в экзамене 2008 г. было только одно задание базового уровня с выбором ответа, которое учащиеся выполнили вполне удовлетворительно: средний процент выполнения составил 71%. Это ниже результатов прошлых лет (93% выполнения в 2007 г., 87% — в 2006 г., 91% — 2005 г.). Этот результат был спрогнозирован при разработке КИМ, так как во всех предыдущих аналитических отчетах отмечалось, что задание было слишком легким. В 2008 г. задание было усложнено и давалось в принципиально новой формулировке.

К разделу **«Основы информационных технологий»** в вариантах КИМ относятся 7 заданий в первой и второй частях работы. Три задания базового уровня и четыре задания повышенного уровня. Анализ этой части работы показывает, что учащиеся имеют хорошее представление о файловой системе организации данных. С заданием А15 базового уровня на воспроизведение знаний в среднем в 2008 году справилось 93% экзаменуемых (82% в 2007 г., 85% в 2006г., 68% — в 2005 г.). Это единственный элемент, хорошо усвоенный даже учащимися, получившими неудовлетворительную оценку. Задание В8 повышенного уровня из раздела «Телекоммуникационные технологии» на прогнозирование результатов поиска информации в Интернете в 2007 г. дало 63% выполнения, что выше результатов 2007 г. (55%) и июня 2006 г. (выпускники Санкт-Петербурга — 61%). Сказывается рост доступности Интернета в регионах (результат выполнения национальных проектов). Эти два задания выполнены экзаменуемыми лучше, чем в предыдущие годы. По другим заданиям темы заметно некоторое снижение результатов.

Задание А19 на чтение данных, представленных в виде диаграмм, в 2008 году было повышенного уровня сложности, что дало снижение процента выполнения с 88% в 2007 г. до 84% в 2008 г. (2006 г.— 74%). Два задания были выполнены с похожими на прошлые годы результатами: 68% экз. справились с заданием А16 (базы данных): в 2007 г.— 71%, в 2006 г.— 62%. 69% экз. успешно выполнили задание А18 (электронные таблицы): результат 2007 г.— 71%, 2006 г.— 76%, Задание повышенного уровня А17 из раздела «Технология обработки графической и звуковой информации» было дано в новой формулировке и проверяло иной, чем в прошлые годы элемент содержания (кодирование Web-страниц в RGB модели). Оно вызвало определенные затруднения, но в целом разброс процента выполнения по вариантам (от 42% до 56%) соответствует уровню сложности (в 2007 г. средний результат был 69% и тогда он соответствовал базовому, а не повышенному уровню сложности). Задание В7 также было новым, так как в про-

шлые годы результаты выполнения этого задания устойчиво росли и в прежней форме задание уже не соответствовало повышенному уроню сложности, предусмотренному спецификацией. Средний результат выполнения этого задания 45%. Публикация задания прошлого года приводит к потере его новизны задания и повышает результат его выполнения при проведении экзамена следующего года.

Два задания с развернутым ответом (С1 и С4) были нацелены на проверку знаний учащихся по **технологии программирования**. Одно задание повышенного уровня сложности предполагало поиск и устранение ошибок в уже имеющейся программе и ее доработку, другое предполагало самостоятельное написание программы для решения оригинальной задачи (высокий уровень сложности). Задачи именно этого раздела информатики традиционно являются одними из самых важных при определении уровня подготовки выпускников в вузах, где практикуют вступительные испытания по предмету. В составе КИМ эти два задания позволяют экзаменуемому при успешном их решении набрать до 7 баллов (из 40 первичных).

Средний процент выполнения задания на поиск ошибок составил в этом году 37% (при 39% в 2007 г. и 48% в 2006 г.), в то время как средний процент выполнения задачи на самостоятельное программирование (С4) – 30% (при 10% в 2007 г. и 12% в 2006 г.). Вместе с тем следует помнить, что высшую оценку в 3 балла за задачу на поиск ошибок получили только 16% приступивших к решению, а 3 или 4 балла за задачу на самостоятельное программирование получили только 563 человека из 10347, участвовавших в экзамене, то есть только 5,4%. Самостоятельное программирование по-прежнему может выполняться только наиболее подготовленными выпускниками.

Общий уровень подготовки участников ЕГЭ по информатике можно признать удовлетворительным с учетом специфики преподавания этого предмета в общеобразовательных учреждениях Российской Федерации. Однако, как и в прежние годы, экзамен показал разрыв в требованиях школьной про-

граммы по информатике и приемных комиссий вузов. Задачи третьей группы на программирование (С1 и С4), а также на формализованную запись изученных алгоритмов (С2) на уровне, более-менее соответствующем запросам вузов, выполняет незначительная группа участников экзамена (не более 20% абитуриентов), хотя процент выполнения заданий этой группы в 2008 г. повысился по сравнению с предшествующими годами. Это подтверждает сделанный в еще в 2006 г. вывод об имеющемся противоречии между уровнем подготовки выпускников массовой школы, определяющимся существующими учебными планами и программами, и требованиями, предъявляемыми вузами к абитуриентам, поступающим на специальности компьютерного профиля. Фактически эти требования невозможно реализовать без профильной, дополнительной к базовому школьному курсу информатики, подготовки.

Рекомендации по совершенствованию методики преподавания информатики с учетом результатов ЕГЭ 2008 года

Результаты экзамена в целом показали преемственность с результатами экзаменов 2006 и 2007 годов. Еще раз следует подчеркнуть, что на сегодняшний день наиболее слабым звеном в школьном преподавании курса информатики является обучение программированию, требуемое вузами и в недостаточном объеме реализуемое массовой школой.

При переходе ЕГЭ в штатный режим, профильный характер экзамена по информатике станет очевидным и, как следствие, возрастет внимание будущих абитуриентов к разделу, связанному с программированием. Вряд ли в ближайшее время будет резко увеличено количество задач на составление программ (они достаточно трудоемки), но возможно введение новых элементов содержания в первую и вторую часть работы. При моделировании содержания экзаменационной работы от года к году будет увеличиваться значение профильного стандарта по информатике 2004 г. как основополагающего нормативного документа.

Экзамен 2008 г. (как и экзамен 2007 г.), показал позитивное влияние на результаты ЕГЭ целенаправленной подготовки выпускников к сдаче экзамена в форме ЕГЭ. В то же время он показал и значимость этой подготовки на примере заданий, оказавшихся для участников экзамена незнакомыми. В этом году негативное влияние новизны формы задания на результат его выполнения было особенно заметно. При подготовке учащихся к ЕГЭ следует использовать не только демоверсию текущего (2009) года, но и демонстрационные версии экзамена прошлых лет и опубликованные задания открытого сегмента Федерального банка тестовых заданий. Следует также помнить, что формулировки заданий могут различаться довольно значительно при одном и том же проверяемом содержании.

При выполнении заданий второй части очень важно обратить внимание учащихся на то, что правильным является только ответ, где перечислены все удовлетворяющие условию элементы Учителя математики хорошо знают, что неполный ответ на вопрос (например, потеря корня уравнения) является неверным. Это в полной мере относится и к заданиям по информатике, так как дихотомическая оценка (верно-неверно) заданий с выбором ответа и с кратким ответом требует полного и исчерпывающего ответа на вопрос. В частности, выше уже отмечалось, что основная причина низкого результата выполнения задания B1 в 2008 году — перечисление неполного списка чисел, удовлетворяющих условию. Если вопрос формулируется «сколько элементов удовлетворяет условию?», то правильный ответ предполагает обязательную проверку всех элементов (или нахождения определенных свойств нужных элементов и мысленного доказательства «для себя», почему нужно искать именно такие элементы, для сокращения пространства поиска). Причиной плохого выполнения задания A20 в экзамене 2008 г. (вопрос был сформулирован: «сколько клеток данного поля удовлетворяют условию?») было то, что учащиеся и не проверяли каждую клетку поля (трудоемкий, но надежный способ решения задания), и не смогли вычленить отличительные признаки искомых клеток. При подготовке уча-

щихся к экзамену учитель должен разобрать с учащимися задания данного типа (соответствуют заданиям А18 и В3 демонстрационной версии 2009 года), обращая особое внимание на доказательство верности найденного решения (при выполнении задания необходимо убедиться в том, что ни одно удовлетворяющее условию значение не было потеряно).

При подготовке учащихся к выполнению заданий третьей части работы следует познакомить учащихся с указаниями для экспертов по проверке и оцениванию работ. Это поможет учащимся предотвратить возможные ошибки, проверить полноту своего решения.

На сайте Федерального института педагогических измерений (http://www.fipi.ru) публикуются все документы, связанные с разработкой КИМ для ЕГЭ 2009 г., в том числе демонстрационные версии экзаменационных работ и учебно-тренировочные материалы. Этот сайт содержит официальные документы, соответствующие будущему экзамену и именно на его материалы следует ориентироваться при подготовке к экзамену.

Искусство хитрое цифири
Нас учит: дважды два — четыре,
И помогает нам понять,
Как ложь за истину принять.

Гете. Фауст

Глава 1
Числа

§ 1. Измерение количества информации

1. Теория

1°. Число и цифра. Системы счисления

Числа пронизывают всю нашу повседневную жизнь. Только этого мало: число — это такой же основной элемент нашего сознания и интеллекта, как естественный язык. Человек может не уметь читать и писать, но считать он умеет всегда.

Видимо, человечество открывало значения чисел в следующем порядке.

1. Сначала число имело только небольшие количественные значения. Чтобы сравнить два больших стада овец, считать не нужно, достаточно прогнать их через ворота парами: одна овца берется из одного стада, другая — из остального.

2. С осмыслением понятия времени как линейно упорядоченных событий стала развиваться порядковая сторона числа.

3. Вначале специальных обозначений для чисел не было, и их записывали буквами алфавита. Затем появляются более удобные *цифры*.

4. В высшей математики есть числовые системы, отражающие разнообразные взаимосвязи математических и физических объектов, такие, как комплексные числа.

Понятия числа — одно из неопределяемых математических понятий. Опишем число через способы его применения.

Число — основное понятие математики и информатики, которое отражает представление об окружающем мире отдельных объектов (вещей) в форме:

1) количества (измерения) объектов;
2) счета (порядка) объектов;
3) обозначения (кодирования) объектов;
4) соотношений между объектами (их взаимосвязи).

Значение числа — смысл числа, т. е. информация, скрытая в числе. Например, количество предметов.

В этой книге будем использовать только первые три способа применения чисел для измерения, счета и кодирования объектов. Причем предпочтение будет отдаваться самым элементарным числам — *натуральным*.

Понятие натурального числа в информатике и математике отличаются: в математике натуральные числа начинаются с 1, а в информатике — с 0.

Натуральное число — одно из чисел

0, 1, 2, 3, 4, …

Число — понятие абстрактное, и требует своего названия в устной речи и обозначения на письме. Числа на письме обозначают двумя способами:

1) буквами различных алфавитов;

2) специальными знаками — *цифрами*.

Цифра — специальный знак для графического обозначения чисел.

> В обыденной речи *число* называют *цифрой*. Вот определение из кроссворда: «Число — цифра на листке календаря» («цифра месяца» пока еще не говорят). Это следует учитывать при разговоре.

В этой книге термины «число» и «цифра» используются только в их научном значении.

Таблица 1

Различные обозначения чисел от 1 до 10

Название обозначения	Обозначение чисел										
Современные арабские цифры	0	1	2	3	4	5	6	7	8	9	10
Цифры арабской письменности	٠	١	٢	٣	٤	٥	٦	٧	٨	٩	١٠
Современные римские цифры	—	I	II	III	IV	V	VI	VII	VIII	IX	X
Древнегреческий алфавит	—	α	β	γ	δ	ε	ς	ζ	η	θ	ι
Старославянский алфавит	—	**А**	**В**	**Г**	**Д**	**Є**	**Ѕ**	**З**	**И**	**Ѳ**	**І**
Арабский алфавит	—	ا	ب	ج	د	ه	و	ز	ح	ط	ي

В дальнейшем будем записывать числа только цифрами.

При записи чисел цифрами недостаточно знать, как выглядят цифры. Необходимо владеть правилами конструирования чисел из значков-цифр.

Система счисления — способ записи чисел, включающий две части:

1) цифры;

2) правила записи чисел этими цифрами.

Систему счисления называют также *нумерацией*.

Чтобы понять, как устроена числовая система, рассмотрим различные системы, которые отличаются друг от друга по следующим трем признакам:

1) разное *начертание* цифр;

2) разные *способы записи* чисел цифрами;

3) разное *количество* цифр.

1. Рассмотрим две системы счисления, которые отличаются только начертанием цифр и больше ничем.

Современная цивилизация в использует общепринятое начертание арабских цифр, изучаемое в школе. Но в странах арабской письменности принят совершенно другой рисунок десяти цифр (некоторые напоминают общепринятые цифры), см. две первые строки таблицы 1.

> Обе эти системы возникли в ходе исторического развития двух частей одной и той же прасистемы начертания цифр. Первая система развивалась в Западной Европе, вторая — на Ближнем и Среднем Востоке.
>
> Правила составления чисел в этих системах совершенно совпадают, «вес» разряда уменьшается в обоих системах слева направо (хотя по-арабски пишут справа налево). Например, число 1828 в арабской письменности записывается как ١٨٢٨.

2. По способу записи чисел цифрами, системы счисления делятся на два класса:

1) *позиционные системы счисления*;

2) *непозиционные системы счисления*.

Непозиционная система счисления — система счисления, в которой каждая цифра имеет всегда одно и то же значение, независимо от ее местоположения в записи числа.

Непозиционную систему называют также *символьной*.

Пример непозиционной системы — это *римская система счисления*.

Приведем примеры записи чисел в этой системе.

а. I = 1.

б. II = 1 + 1 = 2.

в. III = 1 + 1 + 1 = 3.

Как видим, римская цифра I всегда обозначает количество 1 на любом месте.

Позиционная система счисления — система счисления, в которой значение цифры *зависит* от ее положения в записи числа.

Примером позиционной системы является обычная школьная *десятичная система счисления*.

Приведем примеры.

а. 1 = 1.

б. 11 = 1 + 10.

в. 111 = 100 + 10 + 1.

Итак, значение цифры 1 зависит от ее места в записи числа.

Русский язык, как и большинство других естественных языков, является десятичным: мы говорим, по известному выражению, не просто прозой, но десятичной прозой. Эта естественная ограниченность языка мешает называть числа из других систем.

3. Позиционные системы отличаются друг от друга количеством цифр.

Основание системы счисления — это количество цифр этой системы.

Позиционные системы называются по своему основанию, например, *десятичная*, *двоичная*, *восьмеричная*, *шестнадцатеричная*.

2°. Двоичная система счисления

Двоичная система счисления — позиционная система счисления, состоящая из двух цифр 0 и 1. *Двоичное число* — число, записанное двоичными цифрами.

Это самая простая система счисления с минимальным основанием 2. Она используется в компьютерах по следующим причинам:

1) простота аппаратной реализации: 1 — есть сигнал, 0 — нет;

2) самое сложное действие — это действие таблицы сложения $1_2 + 1_2 = 10_2$.

Одна и та же запись числа в разных системах счисления может иметь разное *количественное значение*. Например, $10_2 = 2_{10}$. Для их различения основание системы пишется как нижний индекс в *десятичной системе*. В контексте индекс обычно опускается.

Из таблицы 2 видно, что таблица сложения — самая сложная; возможно, слово «сложение» происходит от «сложный».

Таблица 2

Таблицы умножения, сложения и вычитания двоичных чисел

×	0	1
0	**0**	**0**
1	**0**	**1**

+	0	1
0	**0**	**1**
1	**1**	**10_2**

0 − 0 = 0
1 − 0 = 1
1 − 1 = 0
10_2 − 1 = 1

Основное свойство натуральных чисел: следующее натуральное число больше предыдущего на 1.

Применяя это свойство, получаем (не забывайте, что $1_2 + 1_2 = 10_2$):

$1_2 + 1_2 = 10_2$ «десять»;

$10_2 + 1_2 = 11_2$ «одиннадцать»;

$11_2 + 1_2 = 100_2$ «сто» (см. рис. 3).

а	б	в
^{1} 1 + 1 10_2	10_2 + 1 11_2	$^{1\,1}$ 11_2 + 1 100_2

Рис. 3. Вычисление первых двоичных чисел

Таблица 4

Первые натуральные двоичные числа от 0 до 16

Десятичное число	Двоичное число	Десятичное число	Двоичное число
0	0		
1	1	9	1001_2
2	10_2	10_{10}	1010_2
3	11_2	11_{10}	1011_2
4	100_2	12	1100_2
5	101_2	13	1101_2
6	110_2	14	1110_2
7	111_2	15	1111_2
8	1000_2	16	10000_2

Короткие двоичные числа называют по аналогии с десятичными, хотя это не соответствует их количественному значению: 10_2 «десять» — два предмета, 11_2 «одиннадцать» — три, а длинные называют по цифрам: 11001 — «один, один, нуль, нуль, один».

Чередование цифр в двоичных числах удобно представлять как смену показаний механического двоичного счетчика такси. На каждом колесике написаны только две цифры 0 и 1, за один шаг колесико поворачивается на одну цифру, а после полного оборота поворачивается на одну цифру следующее колесико (см. рис. 5).

0 0 0 0 → 0 0 0 1 → 0 0 1 0 → 0 0 1 1 → 0 1 0 0 → 0 1 0 1

Рис. 5. Двоичный счетчик такси

3°. Разрядность чисел. Бит, байт

Разрядность числа — количество цифр в записи числа.

Однозначное число записывается одной цифрой. Их количество — 10 — совпадает с количеством цифр: 0, 1, 2, …, 9.

Двузначное число записывается двумя цифрами. Заметим, что все однозначные числа становятся двузначными, если приписать к ним слева незначащий нуль: числа 00, 01, 02, …, 09 двузначны. Всего получаем 100 двузначных чисел, начиная с самого маленького 00 и заканчивая самым большим 99.

Трехзначное число записывается тремя цифрами. Однозначные и двузначные числа — частный случай трехзначных. Трехзначных десятичных чисел всего 1000 от 000 до 999. И т. д.

n-значное число записывается с помощью n цифр.

Из таблицы 6 получаем, что количество n-значных десятичных чисел равно 10^n при любом натуральном $n \geqslant 1$.

Таблица 6

Количество однозначных, двузначных и т. д. до 10-значных десятичных чисел

Разрядность чисел	Количество чисел с такой разрядностью
1	$10 = 10^1$
2	$100 = 10^2$
3	$1000 = 10^3$
4	$10\,000 = 10^4$
5	$100\,000 = 10^5$
6	$1\,000\,000 = 10^6$
7	$10\,000\,000 = 10^7$
8	$100\,000\,000 = 10^8$
9	$1\,000\,000\,000 = 10^9$
10	$10\,000\,000\,000 = 10^{10}$

При изучении таблицы 4 получаем, что имеется:

1) только 2 однозначных двоичных числа 0 и 1;

2) $4 = 2^2$ двузначных двоичных числа: 00, 01, 10_2 и 11_2;

3) $8 = 2^3$ трехзначных двоичных чисел от 000 до 111_2;

4) $16 = 2^4$ четырехзначных двоичных чисел от 0000 до 1111_2.

Рассуждая далее по аналогии и учитывая опыт десятичной системы (табл. 6), получим таблицу 7 (все числа в которой записаны в десятичной системе).

Однозначными двоичными числами можно пронумеровать от 1 до 2 объектов, двузначными — от 1 до 4, трехзначными — 1—8 и т. д., *n*-значными — от 1 до 2^n объектов.

> Не путайте двоичные числа, записываемые любым количеством двух цифр 0 и 1, и двузначные числа, записываемые двумя цифрами в любой системе счисления!

Таблица 7

Количество однозначных, двузначных и т. д. до 10-значных двоичных чисел

Разрядность чисел	Количество чисел с такой разрядностью
1	$2 = 2^1$
2	$4 = 2^2$
3	$8 = 2^3$
4	$16 = 2^4$
5	$32 = 2^5$
6	$64 = 2^6$
7	$128 = 2^7$
8	$256 = 2^8$
9	$512 = 2^9$
10	$1024 = 2^{10}$

В случае, когда объекты нумеруются с помощью двоичных чисел, то количества цифр, которые входят в состав этих двоичных чисел, носят специальные названия *бит* и *байт*.

Бит — единица измерения количества информации, определяемая как количество информации в испытании с двумя исходами.

> Название «бит» происходит от английского сокращения "bit" либо от слов "BInary elemenT" — «двоичный элемент», либо от слов "BInary digiT" — «двоичный разряд», либо неизвестно откуда.
>
> Бит — выбор ответа «да» или «нет» на вопрос, его электронное представление — «есть сигнал/нет сигнала». В информатике «да» обычно обозначается 1, «нет» — 0.

Бит — это однозначное двоичное число, один бит кодирует два объекта.

Но бит очень уж мал. Наиболее распространен более большой *байт*.

Байт — наименьшая *составная* единица измерения информации, равная 8 битам. Получаем *первый компьютерный инвариант* 8: 1 **байт** = *8* **бит**.

> Кодовые таблицы содержат количество кодов, равное два в степени восемь.
>
> Почему один байт равен 8 битам, не меньше и не больше? Потому что у здания Большого театра в Москве 8 колонн.

Байт — это восьмизначное двоичным числом.

Одним байтом можно закодировать 256 объектов, задав каждый объект одним из 256 восьмизначных двоичных чисел от $0000\ 0000_2 = 0_{10}$ до $1111\ 1111_2 = 255_{10}$.

Получаем *второй компьютерный инвариант* 256: $256 = 2^8$.

> При развитии вычислительной техники потребовались производные единицы байта, аналогичные единицам, как километры, которые в 1000 раз больше метра, или килограммы, которые в 1000 раз больше грамма.

При составлении *производных единиц* байта используется множитель не 1000, а степень числа 2 — *третий компьютерный инвариант* 1024: $1024 = 2^{10}$.

Почему 1024? Возможно, потому, что число 1024 имеет три хорошие свойства: 1) оно на три порядка больше единицы; 2) это число является степенью двойки; 3) 1024 подозрительно почти равно 1000.

Перечислим известные производные единицы байта.

1 килобайт = 1 Кбайт = 1 Кб = 1 К = 1024 байта.

1 мегабайт = 1 Мбайт = 1 Мб = 1 М = 1024 К.

1 гигабайт = 1 Гбайт = 1 Гб = 1 Г = 1024 М.

1 терабайт = 1 Тбайт = 1 Тб = 1 Т = 1024 Г.

1 петабайт = 1 Пбайт = 1 Пб = 1 П = 1024 Т.

1 экзабайт = 1 Эбайт = 1 Эб = 1 Э = 1024 П.

Воспользовавшись близостью чисел 1024 и 1000, немецкий компьютерный журнал напечатал изображение новой денежной единицы — одной киломарки, равной 1024 маркам. К сожалению, затем произошел переход не к киломаркам, а к евро.

Переведем некоторые из этих единиц в другие.

а. 1 Мб = 2^{10} Кб. б. 1 Кб = 2^{-10} Мб.

в. 1 Мб = 2^{10} Кб = $2^{10} \times 2^{10}$ б = 2^{20} б.

г. 1 б = 2^{-10} Кб = $2^{-10} \times 2^{-10}$ Мб = 2^{-20} Мб.

Приведем таблицу перевода из любой единицы в любую.

Таблица 8

Коэффициенты перевода производных единиц байта

Ед-цы	Байт	Килобайт	Мегабайт	Гигабайт	Терабайт	Петабайт	Экзабайт
б	1	2^{-10}	2^{-20}	2^{-30}	2^{-40}	2^{-50}	2^{-60}
Кб	2^{10}	1	2^{-10}	2^{-20}	2^{-30}	2^{-40}	2^{-50}
Мб	2^{20}	2^{10}	1	2^{-10}	2^{-20}	2^{-30}	2^{-40}
Гб	2^{30}	2^{20}	2^{10}	1	2^{-10}	2^{-20}	2^{-30}
Тб	2^{40}	2^{30}	2^{20}	2^{10}	1	2^{-10}	2^{-20}
Пб	2^{50}	2^{40}	2^{30}	2^{20}	2^{10}	1	2^{-10}
Эб	2^{60}	2^{50}	2^{40}	2^{30}	2^{20}	2^{10}	1

4°. **Шестнадцатеричная система счисления**

Двоичная система имеет количество цифр, меньше десяти, а шестнадцатеричная — больше десяти. Поэтому рассмотрим эту систему отдельно.

Шестнадцатеричная система счисления — позиционная система счисления, состоящая из шестнадцати цифр

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Шестнадцатеричное число записывается шестнадцатеричными цифрами.

Шестнадцатеричную систему используют для более удобной, содержательной и короткой записи двоичных чисел или объектов, ими закодированных.

Например, один байт кодируется *восьмизначным* двоичным и всего лишь *двузначным* шестнадцатеричным числом.

Как и при работе с двоичными числами, выпишем сначала первые шестнадцатеричные числа от 0 до 32. Снова воспользуемся основным свойством натуральных чисел: следующее число равно предыдущему плюс 1. Это и позволит достаточно легко последовательно вычислить шестнадцатеричные числа от 0 до 32.

Следует иметь в виду, что при вычислении первых шестнадцатеричных чисел в четырех местах могут возникнуть для новичка характерные трудности, «пороги», связанные с тем, что количество шестнадцатеричных цифр больше десяти. Перечислим эти четыре «порога»:

1) $9 + 1 = A$;

2) $F + 1 = 10_{16}$;

3) $19_{16} + 1 = 1A$;

4) $1F + 1 = 20_{16}$.

Получаем таблицу 9. Отметим важную закономерность: с ростом основания системы разрядность чисел снижается (см. также приложение)!

Таблица 9

Двоичные, десятичные и шестнадцатеричные числа от 0 до 32

Двоичное число	Десятичное число	Шестнадцатеричное число
0	0	0
1	1	1
10_2	2	2
11_2	3	3
100_2	4	4
101_2	5	5
110_2	6	6
111_2	7	7
1000_2	8	8
1001_2	9	9
1010_2	10_{10}	A
1011_2	11_{10}	B
1100_2	12_{10}	C
1101_2	13_{10}	D
1110_2	14_{10}	E
1111_2	15_{10}	F
$1\ 0000_2$	16_{10}	10_{16}
$1\ 0001_2$	17_{10}	11_{16}
$1\ 0010_2$	18_{10}	12_{16}
$1\ 0011_2$	19_{10}	13_{16}
$1\ 0100_2$	20_{10}	14_{16}
$1\ 0101_2$	21_{10}	15_{16}
$1\ 0110_2$	22_{10}	16_{16}
$1\ 0111_2$	23_{10}	17_{16}
$1\ 1000_2$	24_{10}	18_{16}
$1\ 1001_2$	25_{10}	19_{16}
$1\ 1010_2$	26_{10}	$1A_{16}$
$1\ 1011_2$	27_{10}	$1B_{16}$
$1\ 1100_2$	28_{10}	$1C_{16}$
$1\ 1101_2$	29_{10}	$1G_{16}$
$1\ 1110_2$	30_{10}	$1E_{16}$
$1\ 1111_2$	31_{10}	$1F_{16}$
$10\ 0000_2$	32_{10}	20_{16}

Хотя с шестнадцатеричными числами мы не будем производить арифметических действий, выпишем для общего развития шестнадцатеричную таблицу сложения. Изучая таблицу 9 с шестнадцатеричными числами от 0 до 32, легко составить таблицу 10 сложения шестнадцатеричных чисел.

Таблицу умножения шестнадцатеричных чисел можно найти в приложениях.

Таблица 10

Шестнадцатеричная таблица сложения

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
10	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20

Аналогично десятичной и двоичной системам подсчитаем количество однозначных, двузначных и т. д. шестнадцатеричных чисел. Однозначных шестнадцатеричных чисел будет 16, двузначных — от 256 и т. д., n-значных — 16^n. Получим таблицу 10 (все числа в которой записаны в десятичной системе).

В отличие от двоичных чисел при кодировании объектов шестнадцатеричными числами количество цифр, входящее в числа, никак не называется.

Таблица 11

Количество однозначных, двузначных и т. д. до 10-значных шестнадцатеричных чисел

Разрядность чисел	Количество чисел с такой разрядностью
1	$\mathbf{16^1}$
2	$\mathbf{16^2}$
3	$\mathbf{16^3}$
4	$\mathbf{16^4}$
5	$\mathbf{16^5}$
6	$\mathbf{16^6}$
7	$\mathbf{16^7}$
8	$\mathbf{16^8}$
9	$\mathbf{16^9}$
10	$\mathbf{16^{10}}$

5°. Система счисления с основанием n

Система счисления с основанием n — позиционная система счисления, состоящая из n цифр, где $n \geqslant 2$:

$$0, 1, 2, \ldots, \overline{n-1}.$$

n-ричное число записывается n-ричными цифрами.

Например, *троичная система* состоит из трех цифр 0, 1 и 2, *четверичная* — из четырех 0, 1, 2 и 3, а *восьмеричная* — из восьми 0, 1, 2, 3, 4, 5, 6 и 7.

Троичное число записывается троичными цифрами, а *четверичное* — четверичными.

Мы показали, что значение минимальной цифры n-ричной системы счисления равна 0, а максимальная — $n - 1$.

Как и ранее, выпишем первые троичные, четверичные и восьмеричные числа от 0 до 32. Снова воспользуемся основным свойством натуральных чисел: следующее число равно предыдущему плюс 1. Это и позволит достаточно легко последовательно вычислить троичные, четверичные и восьмеричные числа от 0 до 32. Все полученные здесь и ранее числа объединим в таблицу 13 (см. также приложение).

Отметим важную закономерность: с ростом основания системы разрядность чисел снижается (см. также приложение)!

Аналогично десятичной, двоичной и шестнадцатеричной системам подсчитаем количество однозначных, двузначных и т. д. троичных, четверичных, восьмеричных и n-ричных чисел. Получим таблицу 12 (все числа в которой записаны в десятичной системе).

В отличие от случая двоичных чисел при кодировании объектов во всех других системах счисления количество цифр, входящее в эти числа, никак не называется.

Таблица 12

Количество однозначных, двузначных и т. д. до 10-значных троичных, четверичных, восьмеричных и n-ричных чисел

Разрядность чисел	Количество 3-чных чисел	Количество 4-чных чисел	Количество 8-чных чисел	Количество n-чных чисел
1	3^1	4^1	8^1	n^1
2	3^2	4^2	8^2	n^2
3	3^3	4^3	8^3	n^3
4	3^4	4^4	8^4	n^4
5	3^5	4^5	8^5	n^5
6	3^6	4^6	8^6	n^6
7	3^7	4^7	8^7	n^7
8	3^8	4^8	8^8	n^8
9	3^9	4^9	8^9	n^9
10	3^{10}	4^{10}	8^{10}	n^{10}

Таблица 13

2-чные, 3-чные, 4-чные, 8-чные, 10-чные и 16-чные числа

2-чное	3-чное	4-чное	8-чное	10-чное	16-ричное
0	0	0	0	0	0
1	1	1	1	1	1
10	2	2	2	2	2
11	10	3	3	3	3
100	11	10	4	4	4
101	12	11	5	5	5
110	20	12	6	6	6
111	21	13	7	7	7
1000	22	20	10	8	8
1001	100	21	11	9	9
1010	101	22	12	10	A
1011	102	23	13	11	B
1100	110	30	14	12	C
1101	111	31	15	13	D
1110	112	32	16	14	E
1111	120	33	17	15	F
1 0000	121	100	20	16	10
1 0001	122	101	21	17	11
1 0010	200	102	22	18	12
1 0011	201	103	23	19	13
1 0100	202	110	24	20	14
1 0101	210	111	25	21	15
1 0110	211	112	26	22	16
1 0111	212	113	27	23	17
1 1000	220	120	30	24	18
1 1001	221	121	31	25	19
1 1010	222	122	32	26	1A
1 1011	1000	123	33	27	1B
1 1100	1001	130	34	28	1C
1 1101	1002	131	35	29	1G
1 1110	1010	132	36	30	1E
1 1111	1011	133	37	31	1F
10 0000	1012	200	40	32	20

6°. Пиксель. Разрешение. Глубина цвета

Компьютер имеет дело с конечным количеством чисел и объектов (с бесконечностью имеют дело только математики и богословы). Поэтому в компьютерах данные представлены в виде конечного числа маленьких объектов.

В частности, в наиболее распространенных графических устройствах ввода/вывода,— мониторе, сканере и принтере,— графическое изображение представляется в виде матрицы «атомов», маленьких элементарных объектов.

Получается, в отличие от векторной, *растровая графика*.

Пиксель (pixel, PICture ELement) — элементарная единица изображения.

Матрица пикселей — представление пиксельного изображения в виде *матрицы* — рядов и столбцов пикселей (см. рис. 1).

Пиксель еще называют *зерном изображения*, или просто *зерном*, а матрицу пикселей — просто *матрицей*.

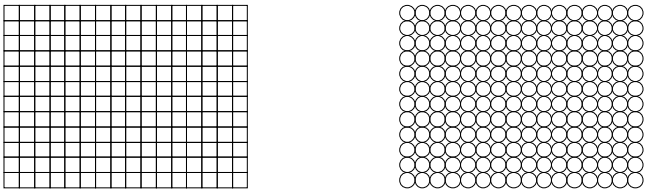


Рис. 14. Схематические изображения матрицы пикселей

Изображение состоит из пикселей. Из пикселей составлены как сами объекты изображении, так и контуры этих объектов.

> На самом деле отделить цветовую составляющую изображения от контурной, формообразующей можно только условно, за счет контраста. Кстати, биологический глаз устроен довольно сложно: одними рецепторами глаз воспринимает цветовую составляющую изображения, а другими — контурную, контрастную.

Четкость и детальность изображения улучшается при: 1) уменьшении размера пикселей; 2) увеличении количества пикселей на единице длины.

Разрешение изображения — количество пикселей на единице длины. Точнее, количество *центров* пикселей, приходящийся на американский дюйм, причем измерение проводится вдоль горизонтальной или вертикальной стороны матрицы пикселей. *Дюйм* — единица длины, часто используемая в компьютерах. *Американский, или английский, дюйм* равен 2,54 см.

Разрешение аппаратуры — разрешение изображений, которое может осуществлять это оборудование.

Разрешение изображения называется просто *разрешением*.

Точка по-английски «dot» («дот»), дюйм — «inch» («инч»), точек на дюйм — «dot per inch», поэтому выражение «точек на дюйм» сокращается как «*dpi*» («дэ-пэ-и», или «ди-пи-ай»).

Разрешение современной аппаратуры обычно одинаково по горизонтали и вертикали и обозначается одним числом в единицах dpi. Все пиксели в изображении одного размера.

Упаковка пикселей — расположение пикселей в изображении так, что расстоянию между их центрами равно их размеру.

Перекрытие пикселей — расположение пикселей в изображении так, что расстояния между их центрами больше их размера.

Схематически разрешение, например, 10 dpi для картинки 30×20 пикселей, можно изобразить, как на рисунке 2.

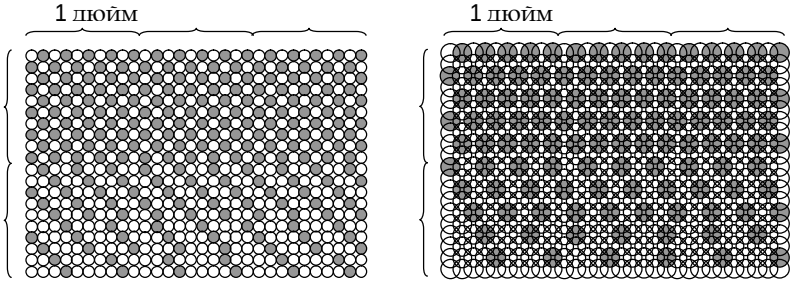


Рис. 15. Разрешение 10 dpi для упаковки (слева) и перекрытия (справа) пикселей

Пиксель — это также цветовая единица изображения: цвет изображения получается из цветов пикселей.

С цветами не все так просто, как кажется. Например, в зависимости от ситуации черный цвет может быть самым темным оттенком любого цвета, а белый — самым светлым.

В смысле количества цветов изображения бывают следующих трех видов.

Цветное изображение (color, colour, цвет, читается «ка́лэ») — изображение из пикселей, которые могут в принципе принимать любые цвета.

Монохромное изображение (monochrome, читается «монохром») — изображение, состоящее из пикселей разной яркости одного цвета: от черного или темного до белого или светлого данного цвета.

Черно-белое изображение (black and white, черный и белый, читается «блэк энд уайт») — изображение, состоящее из пикселей только двух цветов: черного и белого.

За счет чего получается цвет пикселя? Вместе с пикселем в данных хранится его цвет, точнее, код цвета. Под это число может отводиться различный объем памяти.

Глубина цвета — количество битов в коде цвета. Количество цветов, приходящихся на один пиксель, равно два в степени глубины.

Глубина цвета также называется *разрядностью, или битностью, цвета, или просто палитрой*.

Меньше всего памяти требуется пикселям черно-белого или любого двуцветного изображения. Тогда пиксель может быть только 2 цветов, и для его кодировки достаточно 1 бита.

Стандартное количество цветов на первых цветных компьютерах и в стандартных языках программирования — 16, т. е. просто удвоенное количество основного набора компьютерных цветов. 16 цветов кодируются, конечно, 4 битами.

16 цветами передать цветную фотографию невозможно, а 256 — можно (но плохо). Это следующий стандарт цветного компьютера. 256 цветов кодируются 8 битами, или 1 байтом.

Следующий цветной стандарт — это 16-битный, или 2-байтный, цвет. Здесь получается 2^{16} цветов, что достаточно для вполне адекватной передачи цветной фотографии.

Современный стандарт компьютера и Интернета — 24-битный, или 3-байтный цвет (по байту на каждый из трех основных цветов), которого более чем достаточно для передачи цветных фотографий. На мониторах вместо 24-битного цвета может присутствовать 32-битный, что не мешает показывать 24-битный.

1-битный цвет — глубина $2^1 = 2$-цветного пикселя.

4-битный цвет — $2^4 = 16$-цветного.

8-битный цвет — $2^8 = 256$-цветного.

16-битный цвет, или High Color (читается «хай калэ», хороший цвет) — глубина 2^{16}-цветного пикселя.

24-битный, или 32-битный цвет, или True Color (читается «тру калэ», истинный цвет) — глубина 2^{24}- или 2^{32}-цветного пикселя.

7°. **Восприятие цвета человеком**. **Пиксель**, **подпиксель**

> Важнейший факт, что всякий цвет может быть получен надлежащим смешением трех спектральных цветов, выбранных так, чтобы два из них при смешении не могли дать третий, был впервые указан великим русским естествоиспытателем М. В. Ломоносовым

Человек воспринимает *цвета* рецепторами восприятия цвета.

Имеется 3 таких рецептора: один воспринимает в основном *красный* цвет, другой — *зеленый* и третий — *синий* (см. рис. 16). Если на глаз падает свет одного из этих 3 цветов, то работает один рецептор из трех (схему этого процесса см. на рис. 17).

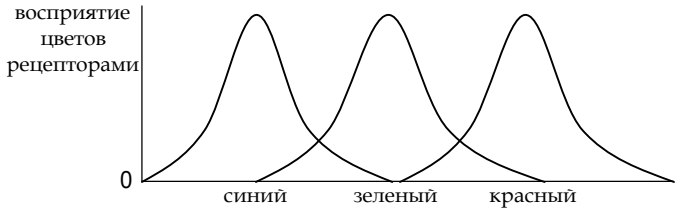


Рис. 16. Степень восприятие человеком цветов рецепторами глаза

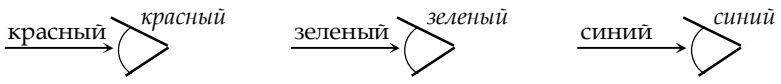


Рис. 17. Схема восприятия трех основных цветов человеком

Не все цвета воспринимаются глазом одинаково чувствительно. Способность различать соседние цвета зависит от цвета (рис. 18). А чувствительность к интенсивности синего на два порядка выше, чем к другим цветам!

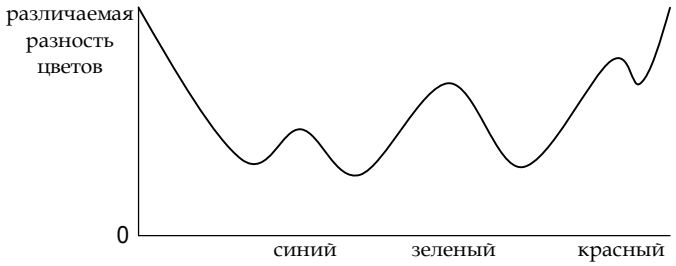


Рис. 18. Изменение способности глаза различать разность цвета

Вышесказанное делает понятным, почему за три основных цвета восприятия цветов выбраны следующие цвета (а не какие-нибудь другие, например, которые предлагал Ломоносов).

Основные цвета восприятия цветов — следующие три цвета:

1) *красный, R* (*red*, читается «рэд»);
2) *зеленый, G* (*green*, читается «грин»);
3) *синий, B* (*blue*, читается «блю»).

Пиксель на экране цветного монитора состоит из этих трех цветов, на монохромном — из одного цвета.

Пиксель монитора состоит из трех точек трех основных цветов (см. рис. 19).

Подпиксель — составная часть цветного пикселя. Цветной пиксель монитора имеет три подпикселя трех основных цветов: красного, зеленого и синего.

Как и мониторы, сканеры работают с теми же тремя цветами, имея три рецептора, перед которыми стоят соответствующе фильтры. Красный, зеленый и синий цвета можно назвать основными цветами монитора и сканера.

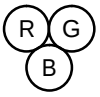


Рис. 19. Схема цветного пикселя и его подпикселей

8°. **Цветовые модели**

Цветовая модель (color model) — математическое описание получения всех цветов из нескольких основных.

Аддитивная цветовая модель — цветовая модель, основанная на сложении цветов, когда цвета при смешивании осветляются.

Цветовая модель RGB (Red, Green, Blue, читается «эр-гэ-бэ») — аддитивная модель с тремя цветами: красным, зеленым и синим.

Проще всего смоделировать *красный, синий и зеленый цвета*. Для этого нужно просто зажечь соответствующие подпиксели и погасить остальные два.

Белый цвет пикселя (*white*) моделируется сложением всех трех основных цветов восприятия, *черный цвет пикселя* (*black*) — отсутствием цветов.

Осталось рассмотреть восприятие пар цветов (см. рис. 20).

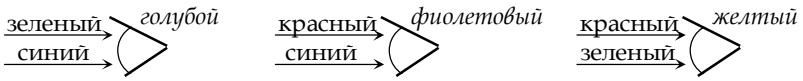


Рис. 20. Схема восприятия пар основных цветов восприятия

Голубой цвет пикселя, C (*синезеленый, бирюзовый, циан, cyan*) моделируется сложением зеленого и синего цветов, *фиолетовый цвет пикселя, M* (*пурпурный, лиловый, магента, magenta*) — сложением красного и синего цветов, *желтый цвет пикселя, Y* (*yellow*) — красного и зеленого.

Восемь основных цветов схемы RGB изображают в следующем виде (см. рис. 21).

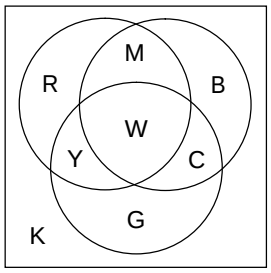


Рис. 21. Цветовая схема RGB (монитора и сканера)

При работе с изображениями в графических редакторах и подготовке веб-страниц накладывают изображения друг на друга или на фон. Удобно, когда некоторые пиксели накладываемого изображения *прозрачные*. Прозрачный пиксель замещает на изображении пиксель любого указанного цвета.

Прозрачный пиксель, или альфа-пиксель (alpha) — это пиксель изображения, который имеет цвет того пикселя, на который он попал при наложении этого изображения на другое или на цветной фон.

Цветовая модель RGBA (Red, Green, Blue, Alpha, читается «эр-гэ-бэ-а») — цветовая модель RGB с прозрачным пикселем.

Все разнообразие цветов, которое человек видит на экране монитора или вводит в компьютер со сканера, получается как комбинация трех основных цветов разной *яркости*.

Яркость подпикселя — степень свечения подпикселя, когда его цвет может меняться от черного до яркого цвета подпикселя.

Разнообразие цветов — следствие глубины цвета: чем больше глубина цвета подпикселей, тем точнее можно моделировать цвет. Стандарт Интернета является глубина цвета 24 бита, когда на каждый подпиксель приходится 1 байт, т. е. цвет пикселя кодируется шестизначным 16-ричным числом.

Шесть цветов цветовой модели, кроме белого и черного, образуют *цветовой круг* перехода цветов (см. рис. 22).

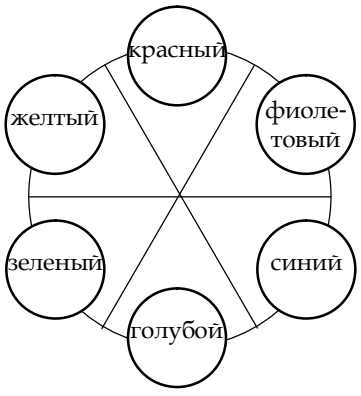


Рис. 22. Цветовой круг

Приведенный цветовой круг универсален и работает не только для цветовых моделей монитора, но также и для цветовых моделей принтера. Все разнообразие цветов вписывается в этот круг.

Цвет изображения на бумаге получается поглощением, вычитанием составляющих падающего белого цвета. При смешивании цветов на бумаге складываются *поглощаемые* цвета. Поэтому цвета печати состоят из цветов, полученных при поглощении минимального количества цветов, воспринимаемых рецепторами человека, т. е. одного основного цвета (рис. 23).

Рис. 23. Схема образования трех основных цветов печати

Основные цвета печати — следующие три цвета красителей:

1) *голубой, C* (cyan, читается «циан»);

2) *фиолетовый, M* (magenta, читается «магента»);

3) *желтый, Y* (yellow, читается «елоу»).

Офисные принтеры работают с субтрактивными моделями *CMY* и *CMYK*.

Сначала рассмотрим цветовую модель CMY.

Субтрактивная цветовая модель — цветовая модель вычитания цветов, когда цвета при смешивании темнеют.

Цветовая модель CMY (Cyan, Magenta, Yellow, читается «цми») — субтрактивная имитация всех воспринимаемых цветов на основе трех основных цветов красителей: голубого, фиолетового и желтого.

Проще всего смоделировать *голубой, фиолетовый и желтый цвета* — просто напечатаются соответствующие подпиксели и не печатаются остальные два.

Белый цвет пикселя (white) моделируется отсутствием всех цветных красителей, *черный цвет пикселя* (black) — наложением всех трех красителей.

Рассмотрим наложение пар красителей основных цветов. Восемь основных цветов схемы CMY изображают так, как показано на рисунке 25.

Красный цвет пикселя моделируется наложением фиолетового и желтого красителей; *зеленый цвет пикселя* — голубого и желтого красителей; *синий цвет пикселя* — голубого и фиолетового (см. рис. 24).

Рис. 24. Схема восприятия наложения пар трех основных цветов печати

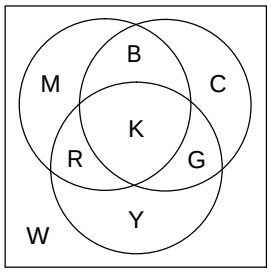


Рис. 25. Цветовая схема CMY (принтера)

Эта идеальная математическая модель не учитывает качества реальных голубого, фиолетового и желтого красителей, которые при наложении мешают друг другу поглощать цвета. Поэтому при смешивании сразу голубого, фиолетового и желтого красителей получается грязно-коричневый, а не черный цвет. Уважающие себя принтеры используют цветовую модель *CMYK*.

Цветовая модель CMYK (Cyan, Magenta, Yellow, blacK, читается «цмик») — цветовая модель CMY, дополненная *черным* цветом.

9°. **Упражнения**

1. Перевести 200 байтов в биты.

2. Перевести 200 битов в байты.

3. Перевести 5 Гб в байты.

4. Перевести 7 Кб в гигабайты.

5. Пересчитать в мегабайты: 10240 Кб; 1024000 Кб. 10 Гб; 1000 Гб. 20 б; 1000 б. 20 Тб; 1000 Тб. Ответ должен легко читаться!

6. К одной телефонной станции подключено 100 номеров, к другой станции — 1000. Двоичными числами какой *минимальной* длины можно закодировать эти номера?

7. Одной компьютерной программе доступны 5000 ячеек памяти, другой — 8000. Двоичными числами какой *минимальной* длины можно закодировать эти ячейки?

8. Одной компьютерной программе на компьютере «Сетунь» с троичной системой счисления доступны 50 ячеек памяти, другой — 100. Троичными числами какой *минимальной* длины можно закодировать эти ячейки?

9. Графическая картинка занимает на экране монитора площадь 15×15 см^2 и содержит 360 тысяч пикселей. Найдите разрешение экрана.

10. Графическая картинка имеет ширину 600 пикселей. При измерении на экране она имеет ширину 15 см, а при измерении на бумаге после печати — 2,5 см. Определите разрешение экрана и принтера.

11. Монитор имеет размер по диагонали 17" и разрешение экрана 1024×768. Найдите разрешение его изображения.

12. Страница формата А4 имеет размер 8×11 дюймов^2. Определите объем файла в Мб, который получился при сохранении рисунка формата А4, сканированного с разрешением 300 точек на дюйм и глубиной цвета 24 бита.

13. Два соседних пикселя на мониторе (рис. 26) имеют зеленый и красный цвета. Какой цвет имеют эти два пикселя вместе?

Тот же вопрос для голубого и красного цветов.

14. Два соседних пикселя на белой бумаге (рис. 27) имеют фиолетовый и голубой цвета. Какой цвет они имеют вместе?

Тот же вопрос для красного и голубого цветов.

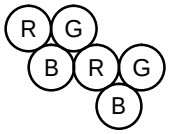


Рис. 26. Схема двух соседних пикселей на мониторе

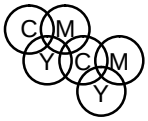


Рис. 27. Схема двух соседних пикселей при печати

Решения

1. 200 б = 8×200 бит = 1600 бит.

2. 200 бит = $\frac{200}{8}$ байт = $\frac{8 \times 25}{8}$ байт = 25 байт.

3. В таблице 8 в строке Гб и столбце байт стоит коэффициент 2^{30}. Поэтому 5 Гб = 5×2^{30} б.

4. В таблице 8 в строке Кб и столбце гигабайт стоит коэффициент 2^{-20}: 7 Кб = 7×2^{-20} Гб.

5. 10240 Кб = 10240×2^{-10} Кб = 10 Мб.

1024000 Кб = 1024000×2^{-10} Кб = 1000 Мб.

10 Гб = 10×2^{10} Мб = 10240 Мб.

1000 Гб = 1000×2^{10} Мб = 1024000 Мб.

20 б = 20×2^{-20} Мб. 1000 б = 1000×2^{-20} Мб.

20 Тб = 20×2^{20} Мб. 1000 Тб = 1000×2^{20} Мб.

6. Найдем по таблице 7 *минимальную* степень двойки, которая больше или равна 100. Это 7: $2^6 = 64 < 100 \leqslant 128 = 2^7$.

Следовательно, 100 номеров телефонной станции можно закодировать двоичными числами минимальной длины 7.

Теперь найдем по таблице 7 *минимальную* степень двойки, большую или равную 1000. Это 10: $2^9 = 512 < 1000 \leqslant 1024 = 2^{10}$.

Следовательно, 1000 номеров телефонной станции можно закодировать двоичными числами минимальной длины 10.

7. Вычислим *минимальную* степень двойки, которая больше или равна 5000. Это 13: $2^{12} = 4096 < 5000 \leqslant 8192 = 2^{13}$.

Следовательно, 5000 номеров телефонной станции можно закодировать двоичными числами минимальной длины 13.

Теперь вычислим *минимальную* степень двойки, большую или равную 8000. Это тоже 13: $2^{12} = 4096 < 8000 \leqslant 8192 = 2^{13}$.

Следовательно, 8000 номеров телефонной станции можно закодировать двоичными числами минимальной длины 13.

8. Вычислим *минимальную* степень тройки, которая больше или равна 50. Это 4: $3^3 = 27 < 50 \leqslant 81 = 3^4$.

Следовательно, 50 ячеек памяти можно закодировать троичными числами минимальной длины 4.

Теперь вычислим *минимальную* степень тройки, которая больше или равна 100. Это 5: $3^4 = 81 < 100 \leqslant 243 = 3^5$.

Следовательно, 100 ячеек памяти можно закодировать троичными числами минимальной длины 5.

9. Разрешение — величина линейная. Поэтому найдем, сколько пикселей картинки приходится на 15 см. Площадь 15×15 см^2 содержит 360 000 пикселей, поэтому пиксели образуют квадрат со стороной $\sqrt{360000} = \sqrt{36}\sqrt{10000} = 6 \cdot 100 = 600$ пикселей. Учитывая, что 15 см — это с большой точностью 6 дюймов, получаем, что разрешение экрана при показе этой картинки равно 600 пикселей / 6 дюймов = 100 dpi.

10. 2,5 см — это практически 1 дюйм. Поэтому разрешение экрана равно 600 пикселей / 6 дюймов = 100 dpi, а разрешение принтера — 600 пикселей / 1 дюйм = 600 dpi.

11. Найдем, сколько пикселей размещается на диагонали монитора. По теореме Пифагора:

$$\sqrt{1024^2 + 768^2} = \sqrt{(256 \cdot 4)^2 + (256 \cdot 3)^2} = \sqrt{256^2}\sqrt{4^2 + 3^2} = 256 \cdot 5 = 1280.$$

Получаем «разрешение по диагонали»: 1280 dot / 17" ≈ 75,3 dpi.

12. Глубина цвета 24 бита = 3 байта. На квадратном дюйме 300·300 пикселей. Итак, объем файла равен 300·300·8·11·3 б = = 23 760 000 б = 23 203,125 Кб ≈ 22,66 Мб.

13. Из модели RGB (рис. 21) следует, что зеленый и красный цвета дадут желтый, а голубой и красный — то же самое, что синий, зеленый и красный, т. е. белый.

14. Из модели CMY (рис. 25) следует, что фиолетовый и голубой цвета дадут синий, а голубой и красный — тот же цвет, что и голубой, фиолетовый и желтый, т. е. черный.

2. Алгоритмы

Итак, подведем итоги. Выпишем алгоритмы, которыми мы пользовались.

При этом обобщим алгоритм, который использовался при решении задачи с троичными числами, с троичной системы счисления на систему счисления с основанием n.

Алгоритм 1. Перевод битов в байты и наоборот.

1. Чтобы перевести число битов в байты, нужно поделить это число на 8.

2. Чтобы перевести число байтов в биты, нужно умножить это число на 8.

Алгоритм 2. Перевод производных единиц байтов.

Чтобы перевести число одних производных единиц байтов в другие, нужно это число умножить на коэффициент, стоящий в таблице коэффициентов перевода на пересечении строки исходных единиц и столбца единиц, которые нужно получить.

Алгоритм 3. Минимальное количество битов для кодирования.

Минимальным количеством битов, которыми можно закодировать m объектов, является минимальная степень числа 2 такая, что 2 в этой степени больше или равно m.

Для небольших m эту степень можно получить подбором, перебирая все степени числа 2, начиная с 0 в возрастающем порядке.

Алгоритм 4. Минимальное количество цифр троичного числа для кодирования.

Минимальным количеством цифр троичного числа, которыми можно закодировать m объектов, является минимальная степень числа 3 такая, что 3 в это степени больше или равно m.

Для небольших m эту степень можно получить подбором, перебирая все степени числа 3, начиная с 0 в возрастающем порядке.

Алгоритм 5. Минимальное количество цифр n-ричного числа для кодирования.

Минимальным количеством цифр n-ричного числа, которыми можно закодировать m объектов, является минимальная степень числа n такая, что n в это степени больше или равно m.

Для небольших m эту степень можно получить подбором, перебирая все степени числа n, начиная с 0 в возрастающем порядке.

Алгоритм 6. Определение результирующего цвета по модели цветов.

Если есть смешанные цвета, переводим их в основные три. После этого определить итоговый цвет смеси цветов не составит труда.

3. Задачи

1°. **Стандартные кодировки символов**

Задача 2006.А1

Считая, что каждый символ кодируется двумя байтами, оцените информационный объем следующего предложения в кодировке Unicode:

Один пуд — около 16,4 **килограмма**.

1) 32 Кбайта 2) 512 бит 3) 64 бита 4) 32 байта

Задача 2007.А1

Считая, что каждый символ кодируется 16 битами, оцените информационный объем следующей пушкинской фразы в кодировке Unicode:

Привычка свыше нам дана: **Замена счастию она**.

1) 44 бита 2) 704 бита 3) 44 байта 4) 704 байта

Задача 2008.А1

В кодировке Unicode на каждый символ отводится два байта. Определите информационный объем слова из 24 символов в этой кодировке.

1) 384 бита 2) 192 бита 3) 256 бит 4) 48 бит

Задача 2009.А1

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

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

2°. **Кодировка различных объектов**

Задача 2006.А3

Сколько мегабайт информации содержит сообщение объемом 2^{23} бит?

1) 1 2) 8 3) 3 4) 32

Задача 2006.А2

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

1) 80 2) 120 3) 112 4) 96

Задача 2007.А2

Световое табло состоит из лампочек, каждая из которых может находиться в двух состояниях («включено» или «выключено»). Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 50 различных сигналов?

1) 5 2) 6 3) 25 4) 50

Задачи 2008.А2, 2009.В1

Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из трех состояний («включено», «выключено» или «мигает»). Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 18 различных сигналов?

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

Задача 2009.А2

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

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

Задача 2007.А3

Метеорологическая станция ведет наблюдение за влажностью воздуха. Результатом одного измерения является целое число от 0 до 100 процентов, которое записывается при помощи минимально возможного количества бит. Станция сделала 80 измерений. Определите информационный объем результатов наблюдений.

1) 80 бит 2) 70 байт 3) 80 байт 4) 560 байт

Задача 2008.А3

Для передачи секретного сообщения используется код, состоящий из десятичных цифр. При этом все цифры кодируются одним и тем же (минимально возможным) количеством бит. Определите информационный объем сообщения длиной в 150 символов.

1) 600 бит 2) 750 бит 3) 1200 бит 4) 60 байт

3°. Кодировка растрового изображения

Задача 2006.А17

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

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

Задача 2007.A17

Для хранения растрового изображения размером 64×64 пикселя отвели 512 байтов памяти. Каково максимально возможное число цветов в палитре изображения?

1) 16 2) 2 3) 256 4) 1024

Задача 2008.A17

Для хранения растрового изображения размером 32×32 пикселя отвели 512 байтов памяти. Каково максимально возможное число цветов в палитре изображения?

1) 256 2) 2 3) 16 4) 4

Задача 2009.A15

Для кодирования цвета фона страницы Интернет используется атрибут bgcolor="#XXXXXX", где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели. Какой цвет будет у страницы, заданной тэгом <body bgcolor="#FFFFFF">?

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

4°. **Передача данных**

Задача 2006.B5

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

Задача 2007.B5

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

Задача 2008.B5

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

Задача 2009.B7

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

4. **Ответы**

1. **Стандартные кодировки символов**

Задача 2006.A1. 2) 512 бит.
Задача 2007.A1. 2) 704 бита.
Задача 2008.A1. 1) 384 бита.
Задача 2009.A1. 2) 60.

2. **Кодировка различных объектов**

Задача 2006.A3. 1) 1.
Задача 2006.A2. 4) 96.
Задача 2007.A2. 2) 6.
Задачи 2008.A2, 2009.B1. 3) 3.
Задача 2009.A2. 3) 490 бит.
Задача 2007.A3. 2) 70 байт.
Задача 2008.A3. 1) 600 бит.

3. **Кодировка растрового изображения**

Задача 2006.A17. 4) 4.
Задача 2007.A17. 2) 2.
Задача 2008.A17. 3) 16.
Задача 2009.A15. 1) белый.

4. **Передача данных**

Задача 2006.B5. 2400.
Задача 2007.B5. 3750.
Задача 2008.B5. 625.
Задача 2009.B7. 40.

5. Решения

1°. Стандартные кодировки символов

Замечание. Сложность задач падает с 2006 по 2008 годы, а в 2009 году резко возрастает.

Рекомендации. 1. При работе с текстом задания может возникнуть проблема подсчета количества символов в строке. Возможно, именно из-за этой проблемы в 2008 году текстовую строку совсем убрали из задания.

Тем не менее отметим, что можно сделать так, что неточное количество символов в строке не влияет на выбор правильного ответа. При неточном подсчете количества символов нужно определить информационный объем и выбирать тот ответ, который близок к полученному результату.

2. Возможные варианты решений в этом разделе несущественны.

3. До 2008 года нужно было внимательно изучать варианты ответа, чтобы выразить полученный ответ в правильных единицах — битах или байтах, которые есть в вариантах ответа.

Задача 2006.A1

Ответ: 2) 512 бит.

Решение. Подсчитаем количество символов данной строки с учетом пробелов и всех знаков препинания, имеем: 32.

Как сказано в условии, каждый символ колируется 2 байтами. Получаем $2 \times 32 = 64$ байта. Но такого ответа среди предложенных на выбор нет!

Переведем в биты: 64 байта = 64×8 битов = 512 битов.

Задача 2007.A1

Ответ: 2) 704 бита.

Решение. Полностью аналогично предыдущей задаче.

Задача 2008.A1

Ответ: 1) 384 бита.

Решение. Полностью аналогично предыдущей задаче.

Задача 2009.А1

Ответ: 2) 60.

Решение 1. Обозначим длину сообщения в символах через x. Тогда в 16-битном коде ее информационный объем будет $16x$ бит, а в 8-битном — $8x$. Получаем уравнение $16x - 8x = 480$. Отсюда $8x = 480$, или $x = 60$.

Решение 2. При перекодировке сообщения из 16-битного кода в 8-битный каждый символ стал занимать на 8 бит меньше, а объем сообщения уменьшился на 480 бит. Следовательно, длина сообщения в символах равна 480 бит / 8 бит = 60.

Решение 3. При перекодировке объем информации уменьшится ровно в 2 раза, поэтому в 8-битной кодировке КОИ-8 он составляет 480 бит. Длина сообщения равна 480 бит / 8 бит = 60.

2°. **Кодировка различных объектов**

Замечание. Первая задача на удивление простая. Таких больше не будет.

Уровень сложности задач колеблется. Может быть, самая сложная задача — вторая. Не так прост и переход к троичной системе счисления.

Также следует внимательно изучать предложенные ответы в первой части теста А. В этом смысле вторая часть теста В проще, нельзя ошибиться при выборе ответа.

Рекомендации. 1. При работе с текстом задания нужно определить, в какой системе счисления следует производить подсчеты. Если имеются два варианта состояния, то в двоичной, а если три — то в троичной.

2. Если и возможны варианты решений в этом разделе, то они несущественны.

3. Нужно внимательно изучать варианты ответов, которые могут быть в разных единицах измерения информации, чтобы выразить полученный ответ в правильных единицах — битах или байтах.

Задача 2006.А3

Ответ: 1) 1.

Решение. Просто переведем из битов в мегабайты.

2^{23} бит = $2^{20} \times 2^3$ бит = 2^{20} байт = 1 Мбайт.

Задача 2006.А2

Ответ: 4) 96.

Решение. Азбука Морзе состоит из двух сигналов, поэтому используем двоичную систему счисления.

Максимальное количество кодов Морзе длиной 5 сигналов имеется всего $2^5 = 32$, а длиной 6 сигналов — $2^6 = 64$.

Поэтому максимальное количество различных символов, которые можно закодировать, используя код Морзе длиной не менее пяти и не более шести сигналов, равно сумме этих чисел: $32 + 64 = 96$.

Задача 2007.А2

Ответ: 2) 6.

Решение. Световое табло состоит из лампочек, каждая из которых может находиться в двух состояниях, поэтому используем двоичную систему счисления.

Лампочек должно быть столько, чтобы максимальное количество сигналов, которое может передать табло, было не меньше 50, и в то же время количество лампочек должно быть минимальным.

Наименьшая степень двойки, не меньшая 50, равна 64: $2^6 = 64$ ($2^5 = 32 < 50 \leqslant 64 = 2^6$). Получаем 6 лампочек.

Задачи 2008.А2, 2009.В1

Ответ: 3) 3.

Решение. Аналогично предыдущей задаче, но в этом задании необходимо использовать не двоичную, а троичную систему счисления и рассматривать степени не двойки, а тройки: $3^2 = 9 < 18 \leqslant 27 = 3^3$.

Задача 2009.А2

Ответ: 3) 490 бит.

Решение. Сначала подсчитаем, сколькими битами записывается номер каждого участника. В велокроссе участвуют 119 спортсменов. Поэтому для записи номера необходимо не менее 7 бит: $2^6 = 64 < 119 \leqslant 128 = 2^7$.

Теперь ясно, что информационный объем сообщения, записанного устройством после того, как промежуточный финиш прошли 70 велосипедистов, составляет в битах ровно 70×7 бит = 490 бит.

Задача 2007.А3

Ответ: 2) 70 байт.

Решение. Аналогично предыдущей задаче, только следует учесть, что необходимо записать 101 число: от 0 до 100 (хотя если ошибочно взять 100 чисел, это на ответ не повлияет).

Получаем правильный ответ в битах: 80×7 бит = 560 бит ($2^6 = 64 < 119 \leqslant 128 = 2^7$). Но такого ответа среди четырех предложенных на выбор нет!

Придется перевести ответ из битов в байты.

80×7 бит = 8×70 бит = 70 байт.

Задача 2008.А3

Ответ: 1) 600 бит.

Решение. Аналогично предыдущей задаче. 10 десятичных цифр кодируются 4 битами ($2^3 = 8 < 10 \leqslant 16 = 2^4$.

Поэтому для кодирования 150 символов необходим информационный объем, который подсчитаем в битах:

150×4 бита = 600 бит.

3°. **Кодировка растрового изображения**

Замечание. Задания достаточно простые.

Нужно представлять, что такое пиксель — зерно изображения и глубина цвета — количество возможных цветов пикселя.

Рекомендации. 1. Из текста нужно выудить данные о количестве пикселей и глубине цвета.

2. Возможные варианты решений несущественны.

3. Варианты ответов в безразмерных единицах.

Задача 2006.A17

Ответ: 4) 4.

Решение. Подсчитаем количество пикселей в растровом изображении: $64 \times 64 = 2^6 \times 2^6 = 2^{12}$.

Каждый пиксель имеет из $256 = 2^8$ цветов свой конкретный цвет, который кодируется 8 битами, поскольку количество возможных цветов пикселя равно 2 в степень 8.

Объем памяти изображения равен произведению количества пикселей этого изображения на информационный объем возможных цветов пикселей, то есть составляет 8×2^{12} бит = $= 2^{12}$ байт = $2^2 \times 2^{10}$ байт = 2^2 килобайта = 4 килобайта.

Задача 2007.A17

Ответ: 2) 2.

Решение. Переведем количество байтов памяти, отведенной для изображения, в биты: $512 \times 8 = 2^9 \times 2^3 = 2^{12}$ бит.

Представим количество пикселей в растровом изображении тоже как степень двойки: $64 \times 64 = 2^6 \times 2^6 = 2^{12}$.

Следовательно, информационный объем палитры равен $\frac{2^{12} \text{ бит}}{2^{12}} = 2^{12} \times 2^{-12} \text{ бит} = 2^0 \text{ бит} = 1 \text{ бит}$.

Одним битом можно закодировать ровно $2^1 = 2$ цвета.

Задача 2008.A17

Ответ: 3) 16.

Решение. Полностью аналогично предыдущей задаче.

Задача 2009.A15

Ответ: 1) белый.

Решение. Упростим вопрос, поставленный в задании: какой цвет передается кодом FFFFFF?

В 24-битной RGB-модели первые две шестнадцатеричные буквы кода задают число, равное интенсивности красного цвета, третья и четвертая буквы кода — интенсивности зеленого и две последние — синего.

Получаем, что все три цвета в модели RGB имеют одинаковую максимальную интенсивность FF, поэтому итоговый цвет будет белый.

4°. **Передача данных**

Рекомендации. 1. Нужно внимательно смотреть на данные и на результат. Необходимо следить за единицами измерения. На лишнюю информацию можно внимания не обращать.

2. Возможные варианты решений несущественны.

3. Вариантов ответов здесь нет.

Задача 2006.B5

Ответ: 2400.

Решение. Вычислим длительность подключения в секундах: 10 мин = 10×60 с = 600 с.

Переведем из килобитов в килобайты объем информации, передаваемый модемом за одну секунду: 32 Кбит/с = 4 Кб/с.

Получаем, что максимальный размер файла составляет $600 \text{ с} \times 4 \text{ Кб/с} = 2400$ Кб.

Задача 2007.B5

Ответ: 3750.

Решение. Полностью аналогично предыдущей задаче.

Задача 2009.B7

Ответ: 625.

Решение. Полностью аналогично предыдущей задаче.

Задача 2009.B7

Ответ: 40.

Решение. Перейдем в скорости передачи данных и размере файла к одним единицам: к байтам. Это удобно сделать, перейдя к степеням. Переведем скорость передачи данных:

$128000\ \text{бит/с} = 2^7 \times 10^3\ \text{бит/с} = 2^4 \times 10^3\ \text{б/с}$.

Переведем размер файла: $625\ \text{Кб} = 5^4\ \text{Кб} = 2^{10} \times 5^4\ \text{б}$.

Определим время в секундах передачи файла в секундах, разделив размер файла на скорость передачи файла:

$$\frac{2^{10}\times 5^4\ \text{б}}{2^4\times 10^3\ \text{б/с}} = \frac{2^6\times 5^4}{(2\times 5)^3}\text{с} = 2^3\times 5\,\text{с} = 40\,\text{с}.$$

§ 2. Перевод чисел из одной системы счисления в другую

1. Теория

1°. Значащие цифры в записи числа

Начнем с обычных десятичных чисел. Важны следующие два взаимосвязанных свойства цифровой записи чисел:

1) любая последовательность десятичных цифр является каким-нибудь числом;

2) любую запись числа можно дополнить без изменения значения числа произвольным количеством нулей, которые приписываются к числу слева.

> Например, число 1 можно записать так:
> а) 1; б) 01; в) 001; г) 0001.
>
> Первое свойство используется в лотереях: в каком бы порядке цифры ни появлялись, все равно получится число. Второе свойство используется в телефонных номерах, номерах автомобилей и документов. Здесь нули приписывают к числам для того, чтобы все они были одной длины.

Получается, что в записи числа могут находиться цифры, которые могут быть отброшены без изменения самого числа. Остальные цифры числа отбрасывать нельзя, поскольку ими определяется число. Цифрами, которые могут быть отброшены, являются, конечно, нули, стоящие слева.

Значащие цифры — все цифры числа, кроме нулей, стоящих слева. Эти нули называются *незначащими*. Исключение: значащей цифрой нуля является 0. Значащие цифры записывают в виде целого числа.

Без примеров в объяснении этого понятия не обойтись.

а. Значащими цифрами числа 123 являются цифры 123.

б. Значащими цифрами числа 0022 являются цифры 22.

в. Значащие цифры числа 00001230,0456 — цифры 12300456.

г. Значащие цифры числа 000987,654000 — 987654000.

В двоичной системе счисления дело обстоит точно так же, как м в десятичной. Определение значащих цифр остается без каких-либо изменений.

а. Значащими цифрами числа 101 являются цифры 101.

б. Значащими цифрами числа 0011 являются цифры 11.

в. Значащие цифры числа 00001101,0101 — цифры 11010101.

г. Значащие цифры числа 000111,111000 — 111111000.

Определение значащих цифр без каких-либо изменений верно и для любой системы счисления с основанием n.

2°. **Операции над двоичными числами**

Рассмотрим четыре элементарные арифметические операции над целыми двоичными числами. Операции над двоичными числами удобно производить точно так же, как и над десятичными числами — столбиком.

При проведении операций над двоичными числами, как и в случае операций над десятичными числами, удобно производить операции столбиком. При этом следует учитывать, что $1_2 + 1_2 = 10_2$, $1_2 + 1_2 + 1_2 = 11_2$, а $10_2 - 1_2 = 1_2$.

1. На рисунке 1 показано сложение двух пар двоичных чисел: 110010_2 (=50) и 110111_2 (=55), 110111_2 (=55) и 111011_2 (=59).

Когда при сложении текущих разрядов двух чисел в двоичной системе получается 10_2 или 11_2, то 1 переходит в следующий разряд.

2. При вычитании нужно помнить, что уменьшаемое не должно быть меньше вычитаемого! Если уменьшаемое меньше вычитаемого, то тогда наоборот, из вычитаемого вычитается уменьшаемое, а к разности приписывается знак –.

На рисунке 1 показано вычитание двух пар двоичных чисел: 1100110_2 (=102) и 1001_2 (=9), 101111_2 (=47) и 11100000_2 (=224).

При вычитании в двоичной системе если из 0 вычитается 1, то этот 0 в вычитаемом становится 10_2, ближайшая слева 1 становится 0, а все нули между ними становятся 1.

```
 1 1  1 1          1 1 1 1 1 1          1 1 10  0 10          0 10
  110010            110111              1100110              11100000
+ 110111          + 111011            -    1001            -  101111
 1101001           1110010             1011101              10110001
```

а б в г

Рис. 1. Сложение и вычитание двоичных чисел:
а) $110010_2 + 110111_2 = 1101001_2$ (50 + 55 = 105);
б) $110111_2 + 111011_2 = 1110010_2$ (55 + 59 = 114);
в) $1100110_2 - 1001_2 = 1011101_2$ (102 – 9 = 93);
г) $101111_2 - 11100000_2 = -10110001_2$ (47 – 224 = –177)

3. На рисунке 2 показано умножение двух пар двоичных чисел: 10001_2 (=17) и 1011_2 (=11); 101001_2 (=41) и 1101000_2 (=104).

Итак, самое сложное в умножении двоичных чисел — сложение *нескольких* чисел, равных первому сомножителю, цифры которого просто сдвинуты по разрядам влево.

4. На рисунке 2 показано деление двух пар двоичных чисел: 100100_2 (=36) и 11_2 (=3); 10001111_2 (=143) и 1101_2 (=13).

Самое сложное в этом процессе — вычитание *двух* чисел, причем это вычитание сильно упрощается тем, что вычитается всегда делитель. Таким образом, двоичное деление проще двоичного умножения.

```
    10001            101001          100100|11          10001111|1101
×    1011          ×   1101000         11    1100          1101     1011
    10001            101001              11                10011
   10001           101001               11                 1101
+ 10001          + 101001               0                  1101
  10111011         1000010101000                            0
```

а б в г

Рис. 2. Умножение и деление двоичных чисел:
а) $10001_2 \times 1011_2 = 10111011_2$ (17 × 11 = 187);
б) $101001_2 \times 1101000_2 = 1000010101000_2$ (41 × 104 = 4264);
в) $100100_2 : 11_2 = 1100_2$ (36 : 3 = 12);
г) $10001111_2 : 1101_2 = 1011_2$ (143 : 13 = 11)

Так же производят операции над числами в любой системе.

3°. **Круглые числа и сумма степеней двоек**

Изучим некоторые свойства целых положительных чисел.

Круглым числом называется целое число, которое оканчивается на один или несколько нулей.

Например, числа 10, 110, 340 — круглые.

Понятно, что круглость числа зависит от системы счисления, в одной системе счисления число может быть круглым, а в другой — нет.

Например, в десятичной системе число 4 не круглое, а в двоичной оно записывается как 100 и является круглым.

Рассмотрим десятичную систему. Любое число в десятичной системе можно представить как сумму степеней 10:

$$a = a_{m-1}10^{m-1} + a_{m-2}10^{m-2} + \ldots + a_1 10^1 + a_0 10^0.$$

В этой формуле по степеням 10 разложено число a, которое имеет в своей записи m десятичных знаков. Если к тому же $a_{m-1} \neq 0$, то число a имеет m значащих цифр.

Например, $123 = 1\cdot 10^2 + 2\cdot 10^1 + 3\cdot 10^0$.

Теперь можно дать другое определение круглого числа.

Круглым числом называется число, у которого количество единиц $a_0 = 0$.

Поэтому круглое число можно записать в виде

$$a = a_{m-1}10^{m-1} + a_{m-2}10^{m-2} + \ldots + a_1 10^1.$$

Круглые числа имеют следующее интересное свойство. Из последнего равенства следует, что *круглое число в десятичной системе делится на 10*:

$$a = 10(a_{m-1}10^{m-2} + a_{m-2}10^{m-3} + \ldots + a_1).$$

Любое число a можно записать не только в десятичной системе счисления, но также и в двоичной, т. е. в виде

$$a = a_{l-1}2^{l-1} + a_{l-2}2^{l-2} + \ldots + a_1 2^1 + a_0 2^0,$$

где l — количество цифр в двоичной записи числа a.

Аналогично десятичной системе, *круглое число в двоичной системе делится на 2*:

$$a = a_{l-1}2^{l-1} + a_{l-2}2^{l-2} + \ldots + a_1 2^1 = 2(a_{l-1}2^{l-2} + a_{l-2}2^{l-3} + \ldots + a_1).$$

Наконец, рассмотрим систему счисления с произвольным основанием n.

Любое число a можно записать в произвольной n-ричной системе счисления, т. е. представить его в виде

$$a = a_{k-1}n^{k-1} + a_{k-2}n^{k-2} + \ldots + a_1n^1 + a_0n^0,$$

где k — количество цифр в n-ричной записи числа a, $n \geqslant 2$.

И в общем случае *круглое число в произвольной n-ричной системе всегда делится на n*!

$$a = a_{k-1}n^{k-1} + a_{k-2}n^{k-2} + \ldots + a_1n^1 = n(a_{k-2}^{k-2} + a_{k-3}^{k-3} + \ldots + a_1).$$

С помощью записи числа в двоичной системе легко доказать следующее полезное соотношение.

Рассмотрим сумму степеней числа 2 от нулевой до l-й:

$$2^0 + 2^1 + \ldots + 2^{l-1} + 2^l.$$

Покажем, что эта сумма равна $2^{l+1} - 1$:

$$2^0 + 2^1 + \ldots + 2^{l-1} + 2^l = \underbrace{11\ldots11}_{l+1\ раз} = \underbrace{100\ldots00}_{l+1\ раз} - 1 = 2^{l+1} - 1.$$

Например, $1 + 2 + 4 = 7 = 8 - 1$, $1 + 2 + 4 + 8 + 16 = 31 = 32 - 1$.

Обобщим доказанное соотношение. Найдем сумму двоек, возведенных в последовательные степени от $k + 1$ до l:

$$2^k + \ldots + 2^l = \underbrace{11\ldots1}_{l-k+1\,раз}\underbrace{00\ldots0}_{k\,раз} = \underbrace{11\ldots1}_{l-k+1\,раз} \cdot \underbrace{10\ldots0}_{k\,раз} = (2^{l-k+1} - 1) \cdot 2^k = 2^{l+1} - 2^k.$$

4°. **Перевод шестнадцатеричных, восьмеричных и четверичных чисел в двоичные и обратно**

Рассмотрим простой алгоритм перевода шестнадцатеричных чисел в *двоичные*. Следующая технология основана на том факте, что однозначное шестнадцатеричное число является четырехзначным двоичным числом (см. табл. 8 или 12 в § 1). Это связано с тем, что основание системы 16 является степенью основания двоичной системы: $16 = 2^4$.

Доказательство этой технологии основано на вычислении суммы степеней двоек и оставляется читателю.

Для перевода числа из шестнадцатеричной системы в двоичную нужно просто заменить каждую шестнадцатеричную цифру ровно на четыре двоичные цифры.

Например, $D = 1101_2$, $2A = 0010\ 1010_2 = 101010_2$.

Перевод чисел из двоичной системы в шестнадцатеричную производится наоборот.

Например, $1101_2 = D$, $101010_2 = 0010\ 1010_2 = 2A$.

Байты, записываемые двоичными числами от $0000\ 0000_2$ до $1111\ 1111_2$, гораздо проще записывать соответствующими шестнадцатеричными числами от 00_{16} до FF_{16}.

Например, символ уникода кодируется 2 байтами и может принимать значения от $00\ 00_{16}$ до $FF\ FF_{16}$, а глубина цвета задается 3 байтами и принимает значения от $00\ 00\ 00_{16}$ до $FF\ FF\ FF_{16}$.

Однозначное восьмеричное число является трехзначным двоичным числом (см. табл. 12 в § 1). Это связано с тем, что основание системы 8 является степенью основания двоичной системы: $8 = 2^3$.

Аналогично для перевода числа из восьмеричной системы в двоичную нужно просто заменить каждую восьмеричную цифру ровно на три двоичные цифры.

Например, $6_8 = 110_2$, $25_8 = 010\ 101_2 = 1\ 0101_2$.

Перевод чисел из двоичной системы в восьмеричную производится наоборот.

Например, $110_2 = 6_8$, $1\ 0101_2 = 010\ 101_2 = 25_8$.

Наконец, однозначное четверичное число — это двузначное двоичное (см. табл. 12 в § 1), поскольку основание системы 4 является степенью основания двоичной системы: $4 = 2^2$.

Точно так же для перевода числа из четверичной системы в двоичную нужно просто заменить каждую четверичную цифру ровно на две двоичные цифры.

Например, $3_4 = 11_2$, $12_4 = 01\ 10_2 = 110_2$.

Перевод чисел из двоичной системы в четверичную производится наоборот.

Например, $11_2 = 3_4$, $110_2 = 01\ 10_2 = 12_4$.

5°. **Перевод целых двоичных чисел в десятичные**

Сначала займемся переводом только натуральных, т. е. целых неотрицательных, чисел.

Как было сказано выше, любое натуральное число можно разложить единственным образом по степеням десятки с коэффициентами, принимающими значение от 0 до 9.

Точно так же любое натуральное число $m > 0$ можно разложить *единственным* образом по степеням двойки:

$$m = a_0 \cdot 2^0 + a_1 \cdot 2^1 + \ldots + a_{n-2} \cdot 2^{n-2} + a_{n-1} \cdot 2^{n-1},\ a_{n-1} = 1.$$

Равенство $a_{n-1} = 1$ означает, что все степени двойки со степенями, большими или равными n, равны 0. Остальные коэффициенты $a_0, a_1, \ldots, a_{n-2}$ равны либо 0, либо 1.

Нуль является исключением: *все* его коэффициенты равны нулю: $0 = 0$.

Например, $1 = 2^0$, $2 = 2^1$, $3 = 2^0 + 2^1$, $4 = 2^2$.

Кроме того, любое натуральное число m можно записать в двоичной системе счисления, причем двоичные цифры *совпадают* с коэффициентами разложения числа m по степеням двойки, но расположены в обратном порядке по сравнению с предыдущей формулой:

$$m = \langle a_{n-1} a_{n-2} \ldots a_1 a_0 \rangle_2,\ a_{n-1} = 1.$$

Равенство $a_{n-1} = 1$ означает, что старший разряд числа равен 1. Остальные $n - 1$ цифры $a_0, a_1, \ldots, a_{n-2}$ равны либо 0, либо 1.

Эти свойства и положены в основу перевода двоичных чисел в десятичные и обратно. Объединяя две последние формулы, получаем самый известный и простой способ перевода двоичных чисел в десятичные: суммированием степеней двоек.

Например,

$0010_2 = 0 + 1 \cdot 2 + 0 \cdot 4 + 0 \cdot 8 + 1 \cdot 16 = 18,$

$100010_2 = 1 \cdot 2 + 1 \cdot 32 = 34,$

$1\,1\,0\,0\,1\,1_2 = 2^0 + 2^1 + 2^4 + 2^5 = 1 + 2 + 16 + 32 = 51,$
5 4 3 2 1 0

$1\,1\,0\,1\,1\,0_2 = 32 + 16 + 4 + 2 = 54.$
32 16 8 4 2 1

6°. **Перевод целых десятичных чисел в двоичные справа налево**

Приведенное выше разложение

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

позволяет по числу m вычислить коэффициенты разложения $a_0, a_1, \ldots, a_{n-2}$ и максимальную ненулевую степень двойки $n - 1$.

Перепишем эту формулу в виде

$$m = a_0 + 2(a_1 \cdot 2^0 + \ldots + a_{n-2} \cdot 2^{n-3} + 2^{n-2}).$$

Тогда при делении числа m на 2 остаток равен a_0, а частное

$$a_1 + a_2 \cdot 2^1 + \ldots + a_{n-2} \cdot 2^{n-3} + 2^{n-2} = a_1 + 2(a_2 \cdot 2^0 + \ldots + a_{n-2} \cdot 2^{n-4} + 2^{n-3}).$$

При делении этого частного на 2 остаток равен a_1, и т. д.:

коэффициент a_0 равен остатку от делении m на 2^1;

коэффициент a_1 равен остатку от делении m на 2^2;

…

коэффициент a_{n-1} равен всегда 1: остатку от делении m на 2^n.

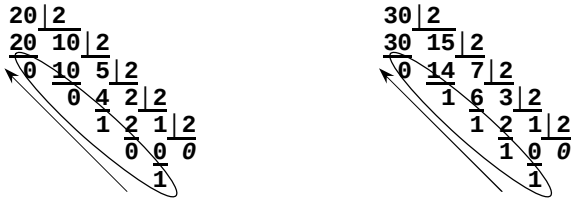
Т. е. коэффициент a_i в разложении произвольного числа m по степеням двойки равен остатку от деления на 2^{i+1}, где i — индекс, пробегающий значения от 0 до $n - 1$, $n - 1$ — максимальная степень двойки, не превышающая исходного числа m:

$$2^{n-1} \leqslant m < 2^n.$$

Но коэффициенты разложения числа m по степеням двойки являются также цифрами его двоичного представления:

$$m = \langle a_{n-1} a_{n-2} \ldots a_1 a_0 \rangle_2,$$

получаем алгоритм перевода десятичного числа в двоичное. Алгоритм прост, поскольку деление производится в хорошо знакомой десятичной системе.



а б

Рис. 3. Перевод десятичных чисел в двоичные: а) $20 = 10100_2$; б) $30 = 11110_2$

7°. **Перевод целых десятичных чисел в двоичные слева направо**

Предыдущий алгоритм перевода десятичных чисел в двоичные еще называют *алгоритмом справа налево*, поскольку цифры двоичного числа получаются, начиная с младшего разряда.

Рассмотрим алгоритм перевода десятичных чисел в двоичные *слева направо*, в котором цифры двоичного числа находят, начиная со старшего разряда.

Снова воспользуемся разложением произвольного натурального числа $m > 0$ по степеням двойки

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

и тем фактом, что имеется показатель степени n такой, что

$$2^{n-1} \leqslant m < 2^n.$$

Приведем пример.

Переведем 5 в двоичную систему.

Подбором находим, что $2^2 \leqslant 5 < 2^3$.

Следовательно, $5 = 2^2 + \ldots$

Находим разность $5 - 2^2 = 5 - 4 = 1$.

Подбором находим, что $2^0 \leqslant 1 < 2^1$.

Следовательно, $5 = 2^2 + 2^0 + \ldots$

Находим разность $1 - 2^0 = 1 - 1 = 0$.

Следовательно, $5 = 2^2 + 2^0$, т. е. $5 = 101_2$.

Приведем еще один пример.

Переведем 54 в двоичную систему.

$$2^5 \leqslant 54 < 2^6,\ 54 - 32 = 22,$$

$$2^4 \leqslant 22 < 2^5,\ 22 - 16 = 6,$$

$$2^2 \leqslant 6 < 2^3,\ 6 - 4 = 2,$$

$$2^1 \leqslant 2 < 2^2,\ 2 - 2 = 0,$$

Получаем, что $54 = 2^5 + 2^4 + 2^2 + 2^1$, т. е. $51 = 110110_2$.

Заметим, что перевод целых отрицательных чисел производится точно так же: переводится модуль числа, а затем к числу снова приписывается знак минус.

8°. **Перевод дробных двоичных чисел в десятичные и обратно**

Любое дробное число состоит из целой и дробной частей, поэтому его можно представить как сумму целого и дробного числа. Как переводить целые числа, мы уже знаем. Разберемся с дробными числами, у которых целая часть равна нулю.

1. Пусть у нас есть дробное число m в двоичной системе:

$$m = \langle 0,a_1a_2 \ldots a_{n-1}a_n\rangle_2,\ a_n = 1.$$

Равенство $a_n = 1$ означает, что младший разряд числа равен 1. Остальные $n-1$ цифры $a_1, a_2, \ldots, a_{n-1}$ равны либо 0, либо 1. Тогда

$$m = \frac{a_1}{2^1} + \frac{a_2}{2^2} + \ldots + \frac{a_{n-1}}{2^{-n+1}} + \frac{a_n}{2^{-n}} = a_1 2^{-1} + a_2 2^{-2} + \ldots + a_{n-1}2^{-n+1} + a_n 2^{-n}.$$

По этой формуле дробные числа переводят из двоичной системы в десятичную.

Например,

$0{,}001_2 = 0\cdot 2^{-1} + 0\cdot 4^{-1} + 1\cdot 8^{-1} = 1/8 = 0{,}125,$

$0{,}1101_2 = 1/2 + 1/4 + 1/16 = 8/16 + 4/16 + 1/16 = 13/16.$
2 4 8 16

2. Приведенное выше разложение

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

позволяет по числу m вычислить коэффициенты разложения $a_1, a_2, \ldots, a_{n-1}$ и максимальную ненулевую степень двойки $-n$.

Перепишем эту формулу в виде

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

Тогда при умножении m на 2 целая часть равна a_1, а дробная

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

При умножении этой дроби на 2 целая часть равна a_2, и т. д.

Получаем алгоритм перевода дробного десятичного числа в двоичное. Алгоритм прост, поскольку умножение производится в хорошо знакомой десятичной системе.

```
0,125 0,25 0,5          0,8125 0,625 0,25 0,5
0,250 0,50 1,0          1,6250 1,250 0,50 1,0
```

а б

Рис. 4. Перевод десятичных дробей в двоичные:
а) $0{,}125 = 001_2$; б) $0{,}8125 = 1101_2$. Черта означает умножение на 2

9°. **Упражнения**

1. Вычислите в двоичной системе счисления.

$11101_2 + 10001_2$; $11011_2 + 10011_2$; $10111_2 + 10111_2$; $11111_2 + 10111_2$.

2. Вычислите в двоичной системе счисления.

Напомним, что столбиком можно вычитать из числа только не большее число. При вычитании большего из меньшего числа при вычитании меняются местами.

$11100_2 - 10001_2$; $11010_2 - 10011_2$; $10110_2 - 10111_2$; $11110_2 - 10111_2$.

3. Для умножения двоичных чисел придется разработать алгоритм сложения более чем 2 чисел (или складывать только по 2 числа). В десятичной системе подобные проблемы переноса в следующий разряд двузначного числа начинаются только при сложении 11 десятичных чисел и поэтому в школе не рассматриваются.

Вычислите в двоичной системе счисления.

$11101_2 \cdot 10001_2$; $11011_2 \cdot 10011_2$; $10111_2 \cdot 10111_2$; $11111_2 \cdot 10111_2$.

4. Вычислите в двоичной системе счисления.

$11011_2 / 11_2$; $11001_2 / 101_2$; $1111110_2 / 110_2$; $111111_2 / 1001_2$.

5. Переведите шестнадцатеричные числа в двоичные.

8D; AB; 1BA; 2D8.

6. Переведите двоичные числа в шестнадцатеричные.

$11\ 1011\ 1101_2$; $111\ 1110\ 0111_2$; $1001\ 1001\ 1001_2$; $1111\ 1101\ 1011_2$.

7. Переведите двоичные числа в десятичные.

1010101010_2; 1101101101_2; 1100110011_2; 1001001001_2.

8. Переведите в десятичные четверичные числа.

$3\ 23\ 31_4$; $13\ 32\ 13_4$; $21\ 21\ 21_4$; $33\ 31\ 23_4$.

9. Переведите в десятичные восьмеричные числа.

1675_8; 3747_8; 4631_8; 7733_8.

10. Переведите в десятичные шестнадцатеричные числа.

8D; AB; 1BA; 2D8.

11. Сколько значащих цифр получится при переводе следующих десятичных чисел в двоичные и какая при этом получится последняя цифра (разряда единиц)?

15; 20; 25; 30; 40; 60; 80; 100.

12. Переведите десятичные числа в двоичные по таблицам.

15; 20; 25; 30; 40; 60; 80; 100.

13. Переведите десятичные числа в двоичные подбором.
200; 300; 400; 800.
14. Переведите десятичные числа в двоичные делением.
1000; 2000; 3000; 10000.
15. Переведите десятичные числа в троичные делением.
1000; 2000; 3000; 10000.

Решения

1.

```
  1    1          1   11          1  111          11111
  11101           11011           10111           11111
+ 10001         + 10011         + 10111         + 10111
  101110          101110          101110          110110
```

2.

```
    0 1 10          0 1 10                          0 10
                      0 10                            0 10
  11100           11010           10111             11110
- 10001         - 10011         - 10110           - 10111
   1011            111             1               111
```

3.

```
     11101           11011           10111           11111
×    10001      ×    10011      ×    10111      ×    10111
     11101           11011           10111           11111
+ 10001         +    11011          10111           11111
  100101101      + 11011            10111           11111
                   1000000001     + 10111         + 11111
                                    1000010001      1011000001
```

4.

```
 11011|11        11001|101      1111110|110      111111|1001
-11    1001     -101   101     -110     10101    -1001   111
  0011             101             111              1101
 -  11            -101            -110             -1001
    0               0               110              1001
                                   -110             -1001
                                     0                0
```

5.

8D = 1000 1101; AB = 1010 1011;
1BA = 0001 1011 1010 = 1 1011 1010;
2D8 = 0010 1101 1000 = 10 1101 1000.

6.

$$11\ 1011\ 1101 = 0011\ 1011\ 1101 = 3BD;$$
$$111\ 1110\ 0111 = 111\ 1110\ 0111 = 7E7;$$
$$1001\ 1001\ 1001 = 999;\ 1111\ 1101\ 1011 = FDB.$$

7.

$$\underset{\textit{512 256 128 64 32 16 8 4 2 1}}{1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0_2} = 512 + 128 + 32 + 8 + 2 = 520 + 160 + 2 = 682;$$

$$\underset{\textit{512 256 128 64 32 16 8 4 2 1}}{1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1_2} = 512+256+64+32+8+4+1 = 520+260+96+1 = 877;$$

$$\underset{\textit{512 256 128 64 32 16 8 4 2 1}}{1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1_2} = 512 + 256 + 32 + 16 + 2 + 1 = 770 + 49 = 819;$$

$$\underset{\textit{512 256 128 64 32 16 8 4 2 1}}{1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1_2} = 512 + 64 + 8 + 1 = 520 + 65 = 585.$$

8.

$$\underset{\textit{256 64 16 4 1}}{3\ 2\ 3\ 3\ 1_4} = 768 + 128 + 48 + 12 + 1 = 780 + 177 = 957;$$

$$\underset{\textit{1024 256 64 16 4 1}}{1\ 3\ 3\ 2\ 1\ 3_4} = 1024 + 768 + 192 + 32 + 4 + 3 = 1060 + 960 + 3 = 2023;$$

$$\underset{\textit{1024 256 64 16 4 1}}{2\ 1\ 2\ 1\ 2\ 1_4} = 2048 + 256 + 128 + 16 + 8 + 1 = 2048 + 400 + 9 = 2457;$$

$$\underset{\textit{1024 256 64 16 4 1}}{3\ 3\ 3\ 1\ 2\ 3_4} = 3072 + 768 + 192 + 16 + 8 + 3 = 3080 + 960 + 19 = 4059.$$

9.

$$\underset{\textit{512 64 8 1}}{1\ 6\ 7\ 5_8} = 512 + 384 + 56 + 5 = 517 + 440 = 957;$$

$$\underset{\textit{512 64 8 1}}{3\ 7\ 4\ 7_8} = 1536 + 448 + 32 + 7 = 1543 + 480 = 2023;$$

$$\underset{\textit{512 64 8 1}}{4\ 6\ 3\ 1_8} = 2048 + 384 + 24 + 1 = 2049 + 408 = 2457;$$

$$\underset{\textit{512 64 8 1}}{7\ 7\ 3\ 3_8} = 3584 + 448 + 24 + 3 = 3608 + 451 = 4059.$$

10.

$8\,D_{16} = 128 + 13 = 141;$
16 1

$A\,B_{16} = 160 + 11 = 171;$
16 1

$1\;B\;A_{16} = 256 + 171 + 10 = 437;$
256 16 1

$2\;D\;8_{16} = 512 + 208 + 8 = 728.$
256 16 1

11.

$2^3 \leqslant 15 < 2^4$, поэтому в двоичной записи будет 4 цифры. 15 — нечетное число, и его двоичная запись оканчивается на 1.

$2^4 \leqslant 20 < 2^5$, поэтому в двоичной записи будет 5 цифр. 20 — четное число, и его двоичная запись оканчивается на 0.

$2^4 \leqslant 25 < 2^5$, поэтому в двоичной записи будет 5 цифр. 25 — нечетное число, и его двоичная запись оканчивается на 1.

$2^4 \leqslant 30 < 2^5$: имеем 5 цифр, последняя — 0.

$2^5 \leqslant 40 < 2^6$: имеем цифр, последняя — 0.

$2^5 \leqslant 60 < 2^6$: имеем 6 цифр, последняя — 0.

$2^6 \leqslant 80 < 2^7$: имеем 7 цифр, последняя — 0.

$2^6 \leqslant 100 < 2^7$: имеем 7 цифр, последняя — 0.

12.

$15 = 1111$; $20 = 1\,0100$; $25 = 1\,1001$; $30 = 1\,1101$;
$40 = 32 + 8 = 10\,0000 + 1000 = 10\,1000$;
$60 = 32 + 28 = 10\,0000 + 1\,1100 = 11\,1100$;
$80 = 64 + 16 = 100\,0000 + 1\,0000 = 101\,0000$;
$100 = 64 + 36 = 64 + 32 + 4 = 100\,0000 + 10\,0000 + 100 = 110\,0100$.

13.

$200 = 128 + 72 = 128 + 64 + 8 = 1100\,1000$;
$300 = 256 + 44 = 256 + 32 + 12 = 256 + 32 + 8 + 4 = 1\,0010\,1100$;
$400 = 256 + 144 = 256 + 128 + 16 = 1\,1001\,0000$;
$800 = 512 + 288 = 512 + 256 + 32 = 11\,0010\,0000$.

14.

Над чертой поместим результаты деления чисел на 2, под чертой — остатки от деления чисел на 2.

```
1000 500 250 125 62 31 15 7 3 1 0
   0   0   0   1  0  1  1 1 1 1
```

1000 = 11 1110 1000

```
2000 1000 500 250 125 62 31 15 7 3 1 0
   0    0   0   0   1  0  1  1 1 1 1
```

2000 = 111 1101 0000

```
3000 1500 750 375 187 93 46 23 12 6 3 1 0
   0    0   0   1   1  1  0  1  0 0 1 1
```

3000 = 1100 1011 1000

```
10000 5000 2500 1250 625 312 156 78 39 19 9 4 2 1 0
    0    0    0    0   1   0   0  0  1  1 1 0 0 1
```

10000 = 10 0111 0001 0000

15.

Над чертой поместим результаты деления чисел на 3, под чертой — остатки от деления чисел на 3.

```
1000 333 111 37 12 4 1 0
   1   0   0  1  0 1 1
```

1000 = 1101001

```
2000 666 222 74 24 8 2 0
   2   0   0  2  0 2 2
```

2000 = 2202002

```
3000 1000 333 111 37 12 4 1 0
   0    1   0   0  1  0 1 1
```

3000 = 11010010

```
10000 3333 1111 370 123 41 13 4 1 0
    1    0    1   1   0  2  1 1 1
```

10000 = 111201101

2. Алгоритмы

Итак, подведем итоги. Выпишем алгоритмы, которыми мы пользовались.

При этом все алгоритмы запишем как в некотором частном, так и в общем виде.

Алгоритм 1. Перевод шестнадцатеричного числа в двоичное.

1. Каждая цифра шестнадцатеричного числа записывается четырехзначным двоичным числом.

2. Незначащие нули, стоящие слева, можно отбросить.

Алгоритм 2. Перевод 2^n-ричного числа в двоичное.

1. Каждая цифра 2^n-ричного числа записывается n-значным двоичным числом.

2. Незначащие нули, стоящие слева, можно отбросить.

Алгоритм 3. Перевод двоичного числа в шестнадцатеричное.

1. Каждые четыре цифры двоичного числа, считая справа налево, записывается одним однозначным шестнадцатеричным числом.

2. Если количество цифр двоичного числа не делится на четыре, то двоичное число дополняется слева нулями до соответствующего количества цифр.

Алгоритм 4. Перевод двоичного числа в 2^n-ричное.

1. Каждые n цифр двоичного числа, считая справа налево, записывается одним однозначным 2^n-ричным числом.

2. Если количество цифр двоичного числа не делится на n, то двоичное число дополняется слева нулями до соответствующего количества цифр.

Алгоритм 5. Перевод двоичного числа в десятичное.

m-значное двоичное число переводится в десятичное по формуле:

$$\langle a_{m-1}a_{m-2}\ldots a_1a_0\rangle_2 = a_0\cdot 2^0 + a_1\cdot 2^1 + \ldots + a_{m-2}\cdot 2^{m-2} + a_{m-1}\cdot 2^{m-1},\ a_{m-1} = 1,$$

где все числа в правой части равенства записаны в десятичной системе.

Алгоритм 6. Перевод *n*-ричного числа в десятичное.

m-значное *n*-ричное число переводится в десятичное по формуле:

$$\langle a_{m-1}a_{m-2}\ldots a_1a_0\rangle_2 = a_0\cdot n^0 + a_1\cdot n^1 + \ldots + a_{m-2}\cdot n^{m-2} + a_{m-1}\cdot n^{m-1},\ a_{m-1} \neq 0,$$

где все числа в правой части равенства записаны в десятичной системе.

Алгоритм 7. Упрощенный перевод двоичного числа в десятичное.

Для перевода двоичного числа в десятичное складываем двойки, возведенные в степени, равные разрядам ненулевых двоичных цифр числа.

Алгоритм 8. Упрощенный перевод *n*-ричного числа в десятичное.

Для перевода *n*-ричного числа в десятичное складываем числа *n*, возведенные в степени, равные разрядам ненулевых *n*-ричных цифр числа.

Алгоритм 9. Перевод десятичного числа в двоичное делением.

1. Число в *десятичной* системе делится на 2. Частное снова делится на 2. И т. д. *Остатки* от деления — цифры 0 и 1 — являются цифрами соответствующего двоичного числа, записанными *справа налево*.

2. Процесс деления прекращается, когда *частное* становится равным нулю.

Алгоритм 10. Перевод десятичного числа в n-ричное делением.

1. Число в *десятичной* системе делится на n. Частное снова делится на n. И т. д. *Остатки* от деления — цифры в диапазоне от 0 до $n-1$ — являются цифрами соответствующего n-ричного числа, записанными *справа налево*.

2. Процесс деления прекращается, когда *частное* становится равным нулю.

Алгоритм 11. Перевод десятичного числа в двоичное подбором.

1. Для данного числа m найдем подбором такую степень двойки n, что

$$2^{n-1} \leqslant m < 2^n.$$

Следовательно, $m = 2^{n-1} + \ldots$

2. Уменьшим число m, вычтя из него 2^{n-1}: $l = m - 2^{n-1}$.

Если l равно 0, то представление числа m в двоичной системе найдено и имеет ровно n цифр: $m = \langle 10\ldots00\rangle_2$.

Если l не равно 0, то переходим к шагу 3.

3. Для нового числа l найдем подбором такое число k, что

$$2^{k-1} \leqslant l < 2^k.$$

Следовательно, $m = 2^{n-1} + \ldots + 2^{k-1} + \ldots$

4. Уменьшим число l, вычтя из него 2^{k-1}: $l \leftarrow l - 2^{k-1}$.

Снова получаем, что если l равно 0, то представление числа m в двоичной системе найдено и имеет вид $m = \langle a_{n-1}a_{n-2}\ldots a_2a_1a_0\rangle_2$, где цифры a_i, индексы которых совпадают со степенями двоек полученного разложения числа, равны 1, а остальные цифры равны 0.

Если l не равно 0, то переходим к шагу 3.

3. Задачи

1°. **Количество нулей или единиц в двоичной записи числа**

Задача 2006.А4

Количество значащих нулей в двоичной записи десятичного числа 126 равно

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

Задача 2007.А4

Сколько единиц в двоичной записи числа 195?

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

Задача 2008.А4

Сколько единиц в двоичной записи десятичного числа 194,5?

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

2°. **Двоичная**, **восьмеричная и шестнадцатеричная системы**

Задача 2006.А5

Вычислите сумму чисел x и y при $x = 1D_{16}$, $y = 72_8$.

Результат представьте в двоичной системе счисления.

1) 10001111_2 2) 1100101_2 3) 101011_2 4) 1010111_2

Задача 2007.А5

Значение выражения $10_{16} + 10_8 \cdot 10_2$ в двоичной системе счисления равно

1) 1010 2) 11010 3) 100000 4) 110000

Задача 2008.А5

Вычислите сумму чисел x и y, при $x = A6_{16}$, $y = 75_8$. Результат представьте в двоичной системе счисления.

1) 11011011_2 2) 11110001_2 3) 11100011_2 4) 10010011_2

Задача 2009.А4

Чему равна сумма чисел 43_8 и 56_{16}?

1) 121_8 2) 171_8 3) 69_{16} 4) 1000001_2

Задача 2009.А3

Дано: $a = D7_{16}$, $b = 331_8$. Какое из чисел c, записанных в двоичной системе, отвечает условию $a < c < b$?

1) 11011001 2) 11011100 3) 11010111 4) 11011000

Задача 2007.А13

Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа (от 00 до 11 соответственно). Если таким способом закодировать последовательность символов ГБВА и записать результат шестнадцатеричным кодом, то получится:

1) 138 2) DBCA 3) D8 4) 3120

Задача 2008.А13

Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа (от 00 до 11 соответственно). Если таким способом закодировать последовательность символов ГБАВ и записать результат в шестнадцатеричной системе счисления, то получится:

1) 132 2) D2 3) 3102 4) 2D

Задача 2009.А11

Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа (от 00 до 11, соответственно). Если таким способом закодировать последовательность символов БАВГ и записать результат шестнадцатеричным кодом, то получится

1) 4B 2) 411 3) BACD 4) 1023

3°. Системы с другими основаниями

Задача 2006.В1

В системе счисления с некоторым основанием число 17 записывается в виде 101. Укажите это основание.

Задача 2007.В1

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

Задача 2008.В1

Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись числа 23 оканчивается на 2.

Задача 2009.В3

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

4. Ответы

1°. **Количество нулей или единиц в двоичной записи числа**

Задача 2006.A4. 1) 1.
Задача 2007.A4. 4) 4.
Задача 2008.A4. 4) 4.

2°. **Двоичная, восьмеричная и шестнадцатеричная системы**

Задача 2006.A5. 4) 101 0111_2.
Задача 2007.A5. 3) 10 0000.
Задача 2008.A5. 3) 111 00011_2.
Задача 2009.A4. 2) 171_8.
Задача 2009.A3. 4) 1101 1000.
Задача 2007.A13. 3) D8.
Задача 2008.A13. 2) D2.
Задача 2009.A11. 1) 4B.

3°. **Системы с другими основаниями**

Задача 2006.B1. 4.
Задача 2007.B1. 6, 9, 18.
Задача 2008.B1. 3, 7, 21.
Задача 2009.B3. 5, 21.

5. Решения

1°. **Количество нулей или единиц в двоичной записи числа**

Замечание. Простые задачи.

Рекомендации. 1. Условия не представляют трудностей.

2. Возможны более короткие решения по сравнению с полным переводом числа в двоичную систему.

3. Варианты ответов в одной системе счисления.

Задача 2006.А4

Ответ: 1) 1.

Решение 1. Переведем число 126 в двоичную систему:

```
126|2                    126₁₀ = 111 1110₂.
126 63|2
  0 62 31|2
    1 30 15|2
      1 14 7|2
        1 6 3|2
          1 2 1|2
            1 0 0
              1
```

Получаем один нуль в записи числа.

Решение 2. Переведем 126 в двоичную систему, воспользовавшись хорошим значением числа:

$126_{10} = 128_{10} - 2_{10} = 2^7{}_{10} - 2^1{}_{10} = 1000\ 0000_2 - 10_2 = 111\ 1110_2$.

Задача 2007.А4

Решение 1. Переведем число 195 в двоичную систему:

```
195|2                    195₁₀ = 1100 0011₂.
194 97|2
  1 96 48|2
    1 48 24|2
      0 24 12|2
        0 12 6|2
          0 6 3|2
            0 2 1|2
              1 1 0
                1
```

Получаем четыре единицы в записи числа.

Решение 2. Посчитаем, сколько раз встретятся нечетные числа при делении 195 на 2:

$\underline{195} \to \underline{97} \to 48 \to 24 \to 12 \to 6 \to \underline{3} \to \underline{1}$.

Получаем четыре единицы в записи числа.

Задача 2008.А4

Решение 1. Переведем число 194,5 в двоичную систему:

```
194|2                    0,5
194 97|2                 1,0
  0 96 48|2
     1 48 24|2
        0 24 12|2
           0 12 6|2
              0 6 3|2
                0 2 1|2
                  1 1 0
                    1
```

$194{,}5_{10} = 1100\ 0010{,}1_2$.

Получаем четыре единицы в записи числа.

Решение 2. Посчитаем, сколько раз встретятся нечетные числа при делении 194 на 2. В следующей цепочки нечетные числа подчеркнуты: $194 \to \underline{97} \to 48 \to 24 \to 12 \to 6 \to \underline{3} \to \underline{1}$.

И подсчитаем, сколько раз встретится целая 1 при умножении 0,5 на 2: $0{,}5 \to \underline{1}{,}0$.

Получаем четыре единицы в записи числа.

2°. **Двоичная**, **восьмеричная и шестнадцатеричная системы**

Замечание. Задачи средней сложности. Самая сложная — 2009.А3.

Рекомендации. 1. Условия не представляют трудностей.

2. Числа быстро и легко переводятся в двоичную систему, поскольку они записаны, кроме двоичной, только в восьмеричной и шестнадцатеричной системах.

3. Нужно быть внимательным с вариантами ответов, которые могут быть приведены в разных системах счисления.

Задача 2006.А5

Ответ: 4) $101\ 0111_2$.

Решение. Переведем оба данных числа в двоичную систему счисления. Воспользуемся тем, что $16 = 2^4$, а $8 = 2^3$.

$1D_{16} = 0001\ 1101_2 = 1\ 1101_2$, $72_8 = 111\ 010_2$.

Осталось сложить два двоичных числа.

```
  11101
+ 111010
 1010111
```

Задача 2007.А5

Ответ: 3) 10 0000.

Решение 1. Переведем оба данных числа в двоичную систему счисления. Воспользуемся тем, что $16 = 2^4$, а $8 = 2^3$.

$10_{16} = 0001\ 0000_2 = 1\ 0000_2$, $10_8 = 001\ 000_2 = 1\ 000_2$.

Перемножим два числа в двоичной системе.

$1\ 000_2 \times 10_2 = 10\ 000_2$.

Осталось сложить два двоичных числа.

```
  10000
+ 10000
 100000
```

Решение 2. Числа в условии круглые, поэтому переведем их в десятичную систему, посчитаем результат, а затем переведем результат в двоичную:

$10_{16} + 10_8 \times 10_2 = 16 + 8 \times 2 = 16 + 16 = 32 = 10\ 0000_2$.

Задача 2008.А5

Ответ: 3) $111\ 00011_2$.

Решение. Полностью аналогично решению задачи 2006.А5.

Задача 2009.А4

Ответ: 2) 171_8.

Решение. Аналогично решению предыдущей задачи, причем все числа, в том числе и варианты ответа, проще перевести в двоичную систему.

Задача 2009.А3

Ответ: 4) 1101 1000.

Решение. Переведем оба данных числа в двоичную систему счисления. Воспользуемся тем, что $16 = 2^4$, а $8 = 2^3$.

$a = D7_6 = 1101\ 0111_2$, $b = 331_8 = 011\ 011\ 001_2 = 1101\ 1001_2$.

Сравним числа из вариантов ответа с числами a и b.

Первый вариант ответа: $a < 1101\ 1001 \leqslant b$ — не подходит.

Второй вариант ответа: $a < 1101\ 1100 > b$ — не подходит.

Третий вариант ответа: $a \leqslant 1101\ 0111 < b$ — не подходит.

Четвертый вариант ответа: $a < 1101\ 1000 < b$ — подходит.

Задача 2007.А13

Ответ: 3) D8.

Решение 1. Запишем данную последовательность ГБВА в виде двоичного числа: 1101 1000. Переведем его в шестнадцатеричное: D8.

Решение 2. 4 символа закодируются 8 двоичными цифрами, поэтому при переводе его в шестнадцатеричный код получится не более 2 шестнадцатеричных цифр. Но все ответы, кроме 3), имеют более 2 шестнадцатеричных цифр и поэтому не подходят.

Задача 2008.А13

Ответ: 2) D2.

Решение. Полностью аналогично предыдущей задаче. Только не проходит решение 2.

Задача 2009.А11

Ответ: 1) 4B.

Решение. Полностью аналогично предыдущей задаче. И решение 2 проходит.

3°. Системы с другими основаниями

Замечание. Задачи повышенной сложности. Их решение носит в некоторой степени творческий характер.

Рекомендации. 1. Приходится внимательно изучать условие задания, чтобы правильно выбрать метод решения.

2. Нужно хорошо представлять общие свойства систем счисления и уметь ими пользоваться. По условию задачи и свойствам систем счисления нужно суметь подобрать метод решения задачи. Иногда по данным задачи удается составить уравнение, что может быть использовано учащимися, слабо разбирающимися в системах счисления. Часто помогает переход к круглому числу.

3. Варианты ответов не приводятся.

Задача 2006.B1

Ответ: 4.

Решение 1. Обозначим искомое основание через n. Тогда получим уравнение $n^2 + 1 = 17$, или $n^2 = 16$. Отсюда $n = 4$, поскольку n — положительное число.

Решение 2. 17 записывается в виде 101. Тогда 16 запишется в виде круглого числа 100, т. е. если n — искомое основание системы, то $n^2 = 16$. Отсюда $n = 4$.

Решение 3. Переберем все системы, пока не получим 17.

Двоичная: $101_2 = 4 + 1 = 5 < 17$.

Троичная: $101_3 = 9 + 1 = 10 < 17$.

Четверичная: $101_4 = 16 + 1 = 17$.

Задача 2007.B1

Ответ: 6, 9, 18.

Решение 1. Если запись числа 22 оканчивается на 4, то запись числа 18 круглая и оканчивается на 0. Следовательно, основания искомых систем счисления являются делителями 18.

С другой стороны, если запись числа 22 оканчивается на 4, то основания искомых систем больше или равны 5.

Делителями 18, которые больше или равны 5, являются числа 6, 9, 18.

Решение 2. Если запись числа 22 оканчивается на 4, то основания искомых систем больше или равны 5.

Поскольку $5^2 > 22$, то запись числа 22 содержит не более 2 цифр даже для пятеричной системы, а для систем с бóльшим основанием — тем более.

С другой стороны, число 22 не может быть записано одной цифрой 4.

Итак, запись числа 22 состоит из двух цифр. Пусть первая их них x, а основание системы n. Тогда получаем уравнение $xn + 4 = 22$, или $xn = 18$, где x и n — целые положительные числа, $n \geqslant 5$. Следовательно, $n = 6, 9, 18$.

Задача 2008.B1

Ответ: 3, 7, 21.

Решение. Полностью аналогично предыдущей задаче.

Задача 2009.B3

Ответ: 5, 21.

Решение. Просто переберем все эти числа.

Наименьшее число в четверичной системе, которое оканчивается на 11, это 11. В десятичной системе оно равно $4 + 1 = 5 \leqslant 25$.

Следующее число в четверичной системе, которое оканчивается на 11, это 111. В десятичной системе оно равно $16 + 4 + 1 = 21 \leqslant 25$.

Следующее число в четверичной системе, которое оканчивается на 11, это 211. В десятичной системе оно равно $32 + 4 + 1 = 37 > 25$.

Все остальные числа будут тем более больше 25.

§ 3. Электронная таблица

1. Теория

1°. Запись больших и дробных чисел

Рассмотрим пунктуационные правила записи десятичных чисел в русских и американских текстах. Русское письмо необходимо знать русскому грамотному человеку.

Американское письмо также полезно знать по следующим причинам:

1) чтобы не путать с русским письмом;

2) по причине использования в компьютерах.

Рассмотрим две особенности русского письма.

1. Записи *целых* чисел обычно разбивают (можно и не разбивать) для улучшения их восприятия на группы по три цифры, считая *справа налево*.

Причем разбивают только числа, больше или равные 10 000. При этом символом отделения групп цифр может быть либо пробел, либо точка.

Примеры. а. 10 000. б. 10.000. в. 10.000.000. г. 10 000 000.

В обычном (не учебном) тексте можно использовать только один из символов отделения: либо пробел, либо точку.

Числа с дробной частью на группы не делится.

2. Дробная часть числа отделяется от целой части запятой.

Примеры. а. $\pi = 3{,}141593$. б. $e = 2{,}718281828$.

Приведем особенности американского письма.

1. Записи целых чисел, больших или равных 10 000, обычно также разбивают на группы по три цифры, считая *справа налево*. Но символом отделения групп цифр может быть либо пробел, либо *запятая*.

Примеры. а. 10 000. б. 10,000. в. 10,000,000. г. 10 000 000.

В обычном (не учебном) тексте можно использовать только один из символов отделения: либо пробел, либо запятую.

2. Дробная часть числа отделяется от целой части *точкой*.

Примеры. а. $\pi = 3.141593$. б. $e = 2.718281828$.

2°. **Константа. Правила записи чисел на компьютере**

Некоторые данные в формулах известны сразу,— это *константы*.

Константа — фиксированное данное, не меняющееся при расчетах.

Важной особенностью записи чисел на компьютере является обязательное наличие двух стандартных типов числовых констант: *с фиксированной точкой* и *с плавающей точкой*.

Константа в фиксированной форме, или с фиксированной точкой — запись числа с десятичной запятой.

Экспонента в записи числа — это символ e или E (от английского exponential — «экспонента»). Играет роль основания системы 10.

Константа в экспоненциальной форме, или с плавающей точкой — запись числа с экспонентой.

Мантисса числа — часть константы в экспоненциальной форме, стоящая перед экспонентой.

Порядок числа — часть константы в экспоненциальной форме, стоящая после экспонентой.

Константой можно представить любое вещественное число (в том числе и целое) по следующей формуле:

$$\textbf{число} = \textbf{мантисса} \times 10^{\textbf{порядок}}$$

Каждая из следующих двух групп констант представляет одно и то же число, первая — +123, вторая — –0,123:

1) 123; 123,0; 123,;
+00123e–00; 0,0123e4; 0012300E–2; +00,0123E+04;

2) –0,123; –00,123; –123e–3; –00123e–03; –0,123E0; –00,00123E+002.

3°. **Формула. Правила записи формул на компьютере**

Формула — математическое выражение, обязательно включающее один или несколько из следующих элементов:

1) арифметическую операцию;
2) арифметическую функцию;
3) отношение сравнения;

4) логическую операцию;

5) логическую функцию.

Арифметическая формула состоит только из арифметических операций и функций.

Отметим две важные особенности записи формул на компьютере (кроме специальных компьютерных математических программ). Эти жесткие правила вызваны тем, что компьютеру необходимо распознавать, что написано в формуле, с помощью специального алгоритма.

Правило 1. Запись формул на компьютере.

1. Формулы записываются только аски-кодами. Это приводит к тому, что все арифметические операции обозначаются значками из аски-кодов, латинскими буквами и цифрами.

2. Формулы пишутся в одну строку. Например, все дроби пишутся в строку. Показатели степени и индексы тоже пишутся таким образом, чтобы оказаться в общей строке.

3. Аргументы функций всегда заключаются в скобки.

4. Знаки операций обязательно указываются. Особенно это относится к знаку умножения.

5. Для задания порядка выполнения операций используются только круглые скобки.

6. Порядок операций с учетом скобок обычный, как в математике: сначала выполняется унарные операции, затем возведение в степень, потом умножение и деление и наконец — сложение и вычитание. Операции одного приоритета выполняются слева направо. Для изменения порядка выполнения операций ставят скобки.

При нарушении первых пяти правил компьютер вообще не воспримет текст как формулу, а при нарушении последнего — воспримет неправильно.

Например, следующая математическая формула на компьютере записывается аски-кодами так:

$$\frac{\sin x \sin 2x}{\cos x \cos 2x} = \text{sin(x)*sin(2*x) / (cos(x)*cos(2*x))}$$

4°. **Арифметические операции**

Над данными можно выполнять разные *операции*, т. е. действия.

Арифметическая операция — вычисление из значений числовых данных новых числовых значений.

Операнды — данные, которые входят в операцию.

Унарная, или одноместная, операция состоит из одного операнда.

Бинарная, или двуместная, операция включает два операнда.

Тернарная, или трехместная, операция имеет три операнда.

Итак, результатом выполнения операции и формулы является значение.

В формулах могут использоваться восемь арифметических операций. Их значки берутся из аски-кодов.

I. *Унарные операции*.

1. *Унарный плюс* + — приписывание перед единственным операндом знака +. Такой плюс игнорируется при вычислениях. Например: +5 = 5.

2. *Унарный минус* - — приписывание перед единственным операндом знака -. Получается противоположное число. Например: -5 = –5.

3. *Процент* % — приписывание после единственного операнда знака %. Получается одна сотая часть числа. Например: 5% = 0,05.

II. *Бинарные операции*.

1. *Сложение* + равно сумме двух *слагаемых* — чисел. Знак операции + стоит между двумя операндами. Например, 2 + 2 = 4.

2. *Вычитание* - равно разности *уменьшаемого* и *вычитаемого* — чисел. Знак операции - стоит между двумя операндами. Например, 5 - 2 = 3.

3. *Умножение* * равно произведению двух *сомножителей* — чисел. Знак операции * стоит между двумя операндами. Например, 2 * 2 = 4.

4. *Деление* / равно частному *делимого* и *делителя* — чисел. Знак операции / стоит между двумя операндами. Например, 5 / 2 = 2,5.

5. *Возведение в степень* ^ равно *основанию* в степени значения *степени* — чисел. Знак операции ^ стоит между двумя операндами. Например, 2^5 = 32.

6. *Деление нацело* div равно частному положительных целых *делимого* и *делителя*, округленному вниз до целого числа. Символ операции div стоит между двумя операндами. Например, 5 div 2 = 2.

7. *Остаток от деления* mod вычисляется по формуле

$$a \bmod b = a - b(a \operatorname{div} b) —$$

для двух целых положительных чисел. Символ операции mod стоит между двумя операндами. Например, 5 mod 2 = 1.

5°. **Арифметические функции**

Перечислим наиболее простые и часто встречающиеся арифметические функции. Запомните, что не только константы и переменные, но и *функции могут быть аргументами функций*!

I. *Функции без аргументов*.

1. pi (другое написание использует скобки: pi() или **пи**()) — значение функции равно числу $\pi = 3{,}141592\ldots$ Конечно, с некоторой точностью, поскольку функция не может возвратить все цифры этого числа. (Вопросы точности вычислений выходят за рамки книги.)

2. *Случайное число* random (другое написание использует скобки: rand() или **слчис**()) равно числу, случайно выбранному из полуинтервала [0, 1).

II. *Функции одного аргумента*. Аргумент всегда заключен в скобки.

1. *Абсолютное значение, или модуль, числа* abs (x) — значение функции равно абсолютному значению аргумента $|x|$. Например, abs(-5) = 5.

2. *Корень квадратный* sqrt (*x*) (другое написание **корень**(*x*)) равен квадратному корню $\sqrt{x}$. Например, sqrt(4) = 2.

3. *Экспонента* exp (*x*) равна экспоненте e^x. Например, exp(0) = 1.

4. *Натуральный логарифм* ln (*x*) равен натуральному логарифму ln *x*. Например, ln(1) = 0, ln(exp(1)) = 1.

5. *Синус* sin (*x*) равен синусу sin *x*. Например, sin(0) = sin(pi) = 0.

6. *Косинус* cos (*x*) равен косинусу cos *x*. Например, cos(0) = 1, cos(pi) = –1.

7. *Арктангенс* arctan (*x*) (другое написание atan (*x*)) равен арктангенсу arctg *x*. Например, arctan(0) = 0.

8. *Целая часть числа* int (*x*) (другое написание **целое** (*x*)) округляет число *x* до ближайшего меньшего или равного целого. Например, int(5,5) = 5.

III. *Функции двух аргументов*. Аргументы всегда заключены в скобки и разделены точкой с запятой.

1. *Степень числа* **степень** (*x*; *y*) равна числу *x* в степени числа *y*. Например, степень(2; 5) = 32.

2. *Остаток от деления числа* **остат** (*x*; *y*) равен остатку от деления числа *x* на число *y*. Например, остат(5; 2) = 1.

IV. *Функции многих аргументов*. Аргументы всегда заключены в скобки и разделены точкой с запятой.

1. *Сумма чисел* sum (*x*; *y*; *z*;…) (другое написание **сумм** (*x*; *y*; *z*;…)) равна сумме аргументов. Например, sum(1; 2; 3; 4) = 10, sum(2^5; exp(0)) = 33.

2. *Минимум чисел* min (*x*; *y*; *z*;…) (другое написание **мин** (*x*; *y*; *z*;…)) равен минимуму перечисленных чисел. Например, min(1; 2; 3; 4) = 1.

3. *Максимум чисел* max (*x*; *y*; *z*;…) (другое написание **макс** (*x*; *y*; *z*;…)) равен максимуму перечисленных чисел. Например, max(1; 2; 3; 4) = 4.

4. *Среднее арифметическое* average (*x*; *y*; *z*;…) (другое написание **срзнач** (*x*; *y*; *z*;…)) равно среднему арифметическому перечисленных чисел. Например, average(1; 2; 3; 4) = 2,5.

6°. **Электронная таблица**, **ее структура и особенности**

Электронные таблицы являются вершиной числовой обработки для пользователей компьютеров, их можно рассматривать как двумерные обобщения формул.

Электронная таблица (ЭТ) — это структура таблицы для хранения чисел, формул и текста, а также для обработки чисел и формул.

Электронной таблицей также называют компьютерную программу для работы с электронной таблицей в предыдущем смысле.

ЭТ как компьютерная программа называется также *табличным процессором, или табличным редактором*.

Выясним детали структуры ЭТ.

Строки ЭТ нумеруются натуральными числами, столбцы — латинскими буквами.

Минимальный элемент хранения данных — *ячейка* — имеет *адрес*, составленный из номера столбца и строки (в этом порядке), на пересечении которых она находится (см. рис. 1).

Столбец / Строка	A	B	C	D
1	Ячейка A1	Ячейка B1	Ячейка C1	Ячейка D1
2	Ячейка A2	Ячейка B2	Ячейка C2	Ячейка D2
3	Ячейка A3	Ячейка B3	Ячейка C3	Ячейка D3
4	Ячейка A4	Ячейка B4	Ячейка C4	Ячейка D4

Рис. 1. Структура рабочего листа электронной таблицы

Текст вводится в ячейках электронной таблицы обычным образом.

ЭТ мощнее, конечно, чем калькулятор. Перечислим особенности ЭТ и ее отличия от калькулятора.

1. В формуле, которая записана в ячейке ЭТ, можно использовать значения нескольких других ячеек (см. рис. 2).

2. Расчет по формулам происходит автоматически. При изменении содержания любой ячейки данные во всех ячейках, с ней связанных, сразу автоматически пересчитываются (см. рис. 2).

На рисунке 2 показан пересчет значений ячеек ЭТ при изменении значения одной ячейки. Поскольку пользователь в данный момент времени может редактировать только одну ячейку, проблем при изменении значений в нескольких ячеек не возникает, поскольку ячейки изменяются последовательно.

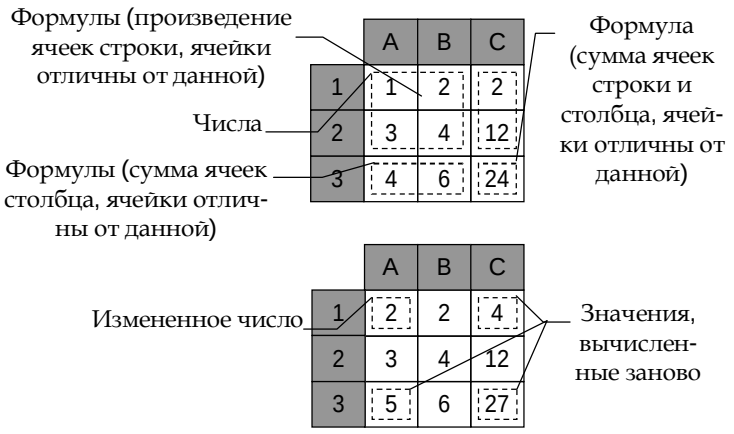


Рис. 2. Вверху: исходная таблица.
Внизу: при изменении содержания одной из ячеек пересчитываются все формулы и значения в ячейках, прямо (C1 и A3) или косвенно (C3) зависящих от измененных

7°. **Относительная ссылка**

Наконец, разберемся со ссылками.

Ссылка — адрес ячейки. В вычислениях используется *значение* этой ячейки.

Формулы, записанные в ячейках ЭТ, могут содержать ссылки на другие ячейки. Другими словами, значение *формул*, находящихся в некоторых ячейках ЭТ, зависит от значения других *ячеек*.

Зависимая ячейка — ячейка, содержащая формулу со ссылкой.

Очевидно, что значение зависимой ячейки зависит от значения ячейки, на которую она ссылается. Значение зависимой ячейки сразу пересчитывается при изменении значения ячейки, от которой она зависит.

Относительная ссылка, или относительный адрес,— просто указание в формуле адреса ячейки. При копировании формулы с относительной ссылкой эта ссылка автоматически изменяется таким образом, чтобы сохранялось расстояние от ячейки с формулой до ячейки, на которую она ссылается.

Примеры формул с относительными ссылками приведены на рисунке 3, примеры копирования формул с относительными ссылками — на рисунке 4.

	A	B	C
1	2	A2	МИН(B1; B3)
2	3	A3	СУММ(B2; B3)
3	5	A1	СРЗНАЧ(B1; B2)

	A	B	C
1	2	3	2
2	3	5	7
3	5	2	4

Рис. 3. Формулы с относительными ссылками.
Слева — содержание ячеек, справа — значения ячеек

	A	B	C
1	2	A2	МИН(B1; B3)
2	3	A3	СУММ(B2; B3)
3	5	A1	СРЗНАЧ(B1; B2)
4			

	A	B	C	D
1				
2		2	B3	МИН(C2; C4)
3		3	B4	СУММ(C3; C4)
4		5	B2	СРЗНАЧ(C2; C3)

Рис. 4. Копирование формул с относительными ссылками.
Слева — содержание ячеек до копирования формул,
справа — после копирования формул на одну ячейку вправо и вниз

8°. **Абсолютная ссылка**

Абсолютная ссылка (адрес) — указание в формуле адреса ячейки, перед одной или обеими координатами которой ставится знак денежной единицы $.

В *полностью абсолютной ссылке (адресе)* перед обеими координатами ставится знак денежной единицы $, в *частично абсолютной ссылке (адресе)* — только перед одной из координат.

При копировании формулы абсолютная часть ссылки не изменяется (см. рис. 5).

Приведем пример двух частично и одной полностью абсолютной ссылки:

$A1; A$1; A1.

	A	B	C
1	2	A$2	МИН(B1; B3)
2	3	A$3	СУММ($B2; B$3)
3	5	A1	СРЗНАЧ(B1; B2)
4			

	A	B	C	D
1				
2		2	B$2	МИН(B1; B3)
3		3	B$3	СУММ($B3; C$3)
4		5	B2	СРЗНАЧ(C2; C3)

Рис. 5. Копирование формул с относительными и абсолютными ссылками. Вверху — содержание ячеек до копирования формул, внизу — после копирования формул на одну ячейку вправо и вниз

Кроме того, в функциях, аргументами которых могут быть несколько чисел, вместо перечисления ячеек может использоваться *диапазон ячеек*.

Диапазон ячеек — это серия подряд идущих ячеек либо по горизонтали, либо по вертикали. Обозначается диапазон указанием крайних ячеек диапазона, разделенных двоеточием.

Например, A3:A7 — диапазон из пяти ячеек, B2:E2 — диапазон из четырех ячеек. Формулы с этими диапазонами могут иметь следующий вид:

МИН(A3:A7), МАКС(A3:A7),

СУММ(B2:E2), СРЗНАЧ(B2:E2).

9°. **Упражнения**

1. Запишите следующие числа как компьютерные константы с плавающей точкой тремя разными способами.

$$1;\ -10^{-10};\ -10^{10};\ 10^{-10}$$

2. Запишите следующие компьютерные константы в виде обычных чисел.

–00.00e–00, –01.01e–01, –01.01e01, 01.01e–01

3. Запишите следующие выражения в виде компьютерных формул.

$$n^2;\ n^3;\ 2n\cdot 3n;\ (-2)\cdot(-n)\cdot(-3)\cdot(-n);$$

$$\frac{x^2+y^3}{1-\dfrac{x^2-y^3}{2}};\ 1+x+\frac{x^3}{2{,}5};\ 1+|x|+|1+x|;\ \sqrt{1+\sqrt{1-x}}$$

4. Запишите следующие компьютерные формулы в математической форме.

```
sqrt(x+y)-sqrt(x-y)
    sqrt(cos(x))
x+y/(x+y)-(x+y)/x+y
1.0+sqrt(cos((x+y)/2.0))
```

5. Запишите, как изменятся следующие ссылки при копировании формулы на одну ячейку влево и вверх:

B2; $B3; B$4; C2; C3; $C4; D$2; D3.

6. Дана электронная таблица. Вычислите значения ее ячеек.

	A	B	C	D
1	5	A1+2	A2*2	A3/2
2	B1+3	СРЗНАЧ(A2; A3; C1)	СРЗНАЧ(A1:D1)	СРЗНАЧ(A1:A4)
3	C1*1,5	СУММ(A1:A4)	СУММ(A1:D1)	СУММ(A2:D2)
4	D1/3	МИН(A1:D1)	МИН(A2:D2)	МИН(A3:D3)

Решения

1.

1e0; 0,1e1; 10e–1.
–1e–10; –0,1e–9; –10e–11.
–1e10; –0,1e11; –10e9.
1e–10; 0,1e–9; 10e–11.

2.

0; –0,101; –10,1; 0,101.

3.

```
n^2
n^3
2*n*3*n
(-2)*(-n)*(-3)*(-n)
(x^2 + y^3)/(1 - (x^2 - y^3)/2)
1 + x + (x^3)/2.5
1 + abs(x) + abs(1 + x)
sqrt(1 + sqrt(1 - x))
```

4.

$\sqrt{x+y}-\sqrt{x-y}$; $\sqrt{\cos x}$; $x+\frac{y}{x+y}-\frac{x+y}{x}+y$; $1+\sqrt{\cos\frac{x+y}{2}}$

5.

A1; $B2; A$4; C2; B2; $C3; C$2; D3.

6.

	A	B	C	D
1	5	7	20	15
2	10	20	11,75	12,5
3	30	50	47	54,25
4	5	5	10	30

2. Алгоритмы

Алгоритм 1. Запись экспоненциальной формы.

1. Для перевода записи числа в экспоненциальную форму оно представляется в виде

$$\textbf{мантисса} \times 10^{\textbf{порядок}}$$

Затем в этой формуле основание 10 заменяется на букву *e*, а знак умножения убирается.

2. Для перевода экспоненциальной формы числа в обычную запись число, стоящее перед буквой *e*, умножается на 10 в степени числа, стоящего после буквы *e*.

Алгоритм 2. Запись формул на компьютере.

1. Формулы записываются только аски-кодами, т. е. значками из аски-кодов, латинскими буквами и цифрами.

2. Формулы пишутся в одну строку.

3. Аргументы функций всегда заключаются в скобки.

4. Знаки операций обязательно указываются. Особенно это относится к знаку умножения *.

5. Для задания порядка выполнения операций используются только круглые скобки.

6. Порядок операций с учетом скобок обычный, как в математике.

Алгоритм 3. Относительная и абсолютная адресация.

Относительный адрес — указание в формуле адреса ячейки. При копировании формулы с такой ссылкой эта ссылка автоматически изменяется так, чтобы сохранялось расстояние от ячейки с формулой до ячейки, на которую она ссылается.

Абсолютный адрес — указание в формуле адреса ячейки, перед одной или обеими координатами которой ставится знак денежной единицы $. При копировании формулы абсолютная часть ссылки не изменяется.

3. Задачи

1°. Адресация в электронной таблице

Задача 2006.А18

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

Примечание: символ $ в формуле обозначает абсолютную адресацию.

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

Задача 2007.А18

В ячейке B1 записана формула =2*$A1. Какой вид приобретет формула после того, как ячейку B1 скопируют в ячейку C2?

Примечание: знак $ используется для обозначения абсолютной адресации.

1) =2*$B1 2) =2*$A2 3) =3*$A2 4) =3*$B2

Задача 2008.А18

Дан фрагмент электронной таблицы:

	A	B	C
1	10	20	=A1+B$1
2	30	40	

Чему станет равным значение ячейки C2, если в нее скопировать формулу из ячейки C1?

Знак $ обозначает абсолютную адресацию.

1) 40 2) 50 3) 60 4) 70

Задача 2009.А16

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

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

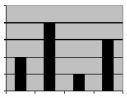
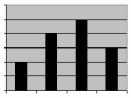
2°. **Диаграмма по электронной таблице**

Задача 2006.А19

Дан фрагмент электронной таблицы:

	A	**B**
1	=B1+1	1
2	=A1+2	2
3	=B2–1	
4	=A3	

После выполнения вычислений, была построена диаграмма по значениям диапазона ячеек A1:A4. Укажите получившуюся диаграмму.

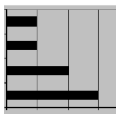
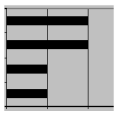
1)  2)  3)  4) 

Задача 2008.А19

Дан фрагмент электронной таблицы:

	A	B	C	D
1		3	4	
2	=C1-B1	=B1-A2*2	=C1/2	=B1+B2

После выполнения вычислений была построена диаграмма по значениям диапазона ячеек A2:D2. Укажите получившуюся диаграмму.

1)  2)  3)  4) 

4. Ответы

1°. Адресация в электронной таблице

Задача 2006.А18. 2) =E1-$D2
Задача 2007.А18. 3) =3*$A2
Задача 2008.А18. 2) 50.
Задача 2009.А16. 4) 4.

2°. Диаграмма по электронной таблице

Задача 2006.А19. 2) 

Задача 2008.А19. 4) 

5. Решения

1°. Адресация в электронной таблице

Замечание. Простые задачи.

Рекомендации. 1. При работе с текстом задания нужно внимательно смотреть, перед какими символами стоит знак $.

2. Решение простое, поэтому одно.

3. Варианты ответов проблем не вызывают.

Задача 2006.A18

Ответ: 2) =E1-$D2

Решение. Буква D в адресе D2 имеет абсолютный адрес, поэтому она не изменится.

Не изменятся в адресах и все числа, поскольку при копировании ячейки A1 в B1 число не изменилось.

Относительный адрес D1 изменит букву D на следующую E, поскольку при копировании ячейки A1 в B1 буква A перешла в следующую B.

Задача 2007.A18

Ответ: 2) =2*$A2

Решение. Полностью аналогично решению предыдущей задачи.

Задача 2008.A18

Ответ: 2) 50.

Решение. Сначала вычислим, какая формула получится в ячейке C2. При копировании формулы из ячейки C1 в C2 буквы формулы не поменяются, а числа из относительных адресов увеличатся на 1. Поэтому формула =A1+B$1 из ячейки C1 перейдет в формулу =A2+B$1 в ячейке C2.

Осталось подсчитать значение формулы:

=A2+B$1 = 30+20 = 50.

Задача 2009.A16

Ответ: 4) 4.

Решение. Заменяя функции, указанные в таблице, на арифметические операции, получаем систему уравнений:

B1 + B2 = 5, (B1 + B2 + B3) / 3 = 3.

Отсюда 5 + B3 = 9, или B3 = 4.

2°. **Диаграмма по электронной таблице**

Замечание. Задачи сложны подбором диаграмм.

Рекомендации. 1. При работе с текстом задания нужно внимательно смотреть, по каким ячейкам составляется диаграмма.

2. Расчеты в таблице трудностей не вызывают, если водить в правильной последовательности, т. е. использовать только уже посчитанные значения ячеек таблицы.

3. Возможны трудности при выборе варианта для определения адекватной диаграммы.

Задача 2006.A19

Ответ: 2)

Решение. Сначала вычислим значения ячеек с A1 по A4.

A1 = B1+1 = 2; A2 = A1+2 = 4; A3 = B2–1 = 1; A4 = A3 = 1.

Получаем два одинаковых самых маленьких значения ячеек, одно значение в два раза больше самого маленького и одно значение в четыре раза больше самого маленького.

На предложенных ответах 1) и 3) нет двух самых маленьких значений, а на ответе 4) большие значения одинаковые.

Под указанные соотношения значений ячеек с A1 по A4 подходит только ответ 2).

Задача 2008.A19

Ответ: 4)

Решение. Полностью аналогично предыдущей задаче.

Клиент либо жив, либо мертв. Если клиент мертв, то его либо можно оживить, либо нельзя.

А. Толстой. Золотой ключик

Глава 2
Логика

§ 1. Логические операции

1. Теория

1°. **Множество**. **Множество как элемент другого множества**

Множество — набор любых объектов. Объекты, входящие в множество, называются его *элементами* и образуют *состав* множества. Ни повторяемость, ни порядок элементов не влияет на состав множества.

Множество — это *один объект*!

Множества будем обозначать прописными латинскими буквами A, …, Z, а их абстрактные элементы — строчными латинскими буквами a, …, z.

Множества будем задавать с помощью перечисления их элементов в фигурных скобках: {…}.

Примеры 1.

Запишем три множества по три элемента:

$$A = \{\textit{улица}, \textit{фонарь}, \textit{аптека}\},$$

$$B = \{a, b, c\},$$

$$C = \{1, 2, 3\}.$$

Множество A состоит из трех слов-элементов *улица*, *фонарь* и *аптека*, множество B — из трех букв-элементов a, b и c, а множество C — из трех чисел-элементов 1, 2 и 3.

Для принадлежности элемента множеству используется значок $\in$.

Примеры 2.

Обозначим принадлежность элементов нашим трем множествам:

$$\textit{улица} \in A, \textit{фонарь} \in A, \textit{аптека} \in A;$$

$$a \in B, b \in B, c \in B;$$

$$1 \in C, 2 \in C, 3 \in C.$$

Множество, состоящее только из одного элемента, называется *одноэлементным*, из двух элементов — *двухэлементным*, из трех — *трехэлементным* и т. д.

Множество, состоящее из n элементов, называется n-элементным.

Множество, совсем не имеющее никаких элементов, называется *нульэлементным, или пустым*, и имеет специальное обозначение:

$$\varnothing \equiv \{\}.$$

Множество называется *конечным*, если оно содержит конечное количество элементов, и *бесконечным*, если содержит бесконечное количество элементов.

Очень важно, что элементами множества могут быть другие множества.

Примеры 3.

Запишем новые множества:

$$D = \{\varnothing\} \equiv \{\{\}\};$$

$$E = \{\{a, b\}, \{a, b, c\}\},$$

$$F = \{\varnothing, \{a\}\}.$$

Обратите внимание, что два множества $\varnothing$ и $\{\varnothing\}$ — совершено разные множества. Множество $\varnothing$ не содержит никаких элементов и является пустым, а множество $\{\varnothing\}$ содержит ровно один элемент — другое множество — и является одноэлементным.

Множество E содержит два элемента: двухэлементное множество $\{a, b\}$ и трехэлементное множество $\{a, b, c\}$.

Множество F содержит тоже два элемента: пустое множество $\varnothing$ и одноэлементное множество $\{a\}$.

2°. **Подмножество**

Пример 4.

Рассмотрим множество U, состоящее из двенадцати студентов, пронумерованных числами от 1 до 12: $U = \{1, 2, \ldots, 12\}$.

Предположим, что студенты с номерами 1, 2, 3 и 4 не достигли 18 лет, а студенты 1, 5 и 6 являются отличниками.

Обозначим множество студентов, не достигших 18 лет, буквой A, а множество отличников — буквой B. Тогда можно выписать множества A и B:

$$A = \{1, 2, 3, 4\}, B = \{1, 5, 6\}.$$

Говорят, что множество U включает два *подмножества*: A и B, а множества A и B, в свою очередь, принадлежат множеству U. Этот факт обозначается так:

$$A \subset U, B \subset U.$$

Множество U по отношению к множествам A и B называется *универсальным*, поскольку элементы для A и B берутся только из U.

Если хорошо присмотреться к полученной конструкции: множество U и два его подмножества A и B: то можно разглядеть еще четыре готовых подмножества универсального множества U:

1) $C_0 = \{7, 8, 9, 10, 11, 12\}$ — студенты, которые не обладают ни одним из названных свойств;

2) $C_1 = \{2, 3, 4\}$ — студенты, которые обладают только свойством A — быть моложе 18 лет;

3) $C_2 = \{5, 6\}$ — студенты, которые обладают только свойством B — быть отличниками;

4) $C_3 = \{1\}$ — студенты, которые обладают сразу обоими свойством A и B — быть отличниками моложе 18 лет.

Эти четыре множества — подмножества не только универсального множества U, но и подмножества его подмножеств A и B. Выпишем цепочки принадлежностей множеств друг другу:

$$C_0 \subset U;$$
$$C_1 \subset A \subset U;$$
$$C_2 \subset B \subset U;$$
$$C_3 \subset A \subset U, C_3 \subset B \subset U;$$

3°. **Диаграмма Эйлера — Венна**

Изобразим эти множества графически на полной диаграмме Эйлера — Венна (см. рис. 5). *Полная диаграмма Эйлера — Венна* — диаграмма, на которой представлены все возможные случаи взаимного расположения подмножеств A и B универсального множества U.

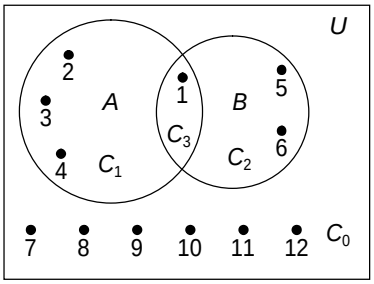


Рис. 5. Полная диаграмма Эйлера — Венна

Диаграммы Эйлера — Венна могут и не иметь всей полноты общности, как на рисунке 5.

Пример 6.

На диаграмме Эйлера — Венна на рисунке 7 множество A является подмножеством множества B, чего не было в примере 4, т. е.

$$A \subset B \subset U, \text{ или } \{1, 5\} \subset \{1, 2, 3, 4, 5\} \subset \{1, 2, \ldots, 12\}.$$

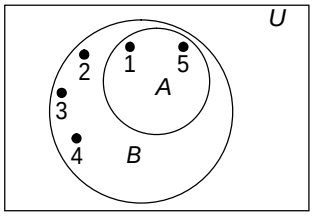


Рис. 7. Неполная диаграмма Эйлера — Венна

4°. **Операция объединения множеств**

Введем операцию объединения множеств на конкретном примере.

На примере универсального множества $U = \{1, 2, \ldots, 12\}$ *объединением множеств* $A = \{1, 2, 3, 4\} \subset U$ и $B = \{1, 5, 6\} \subset U$ называется множество

$$A \cup B = \{1, 2, 3, 4, 5, 6\},$$

где $\cup$ — символ объединения множеств.

Таким образом, объединением множеств A и B охватываются три множества C_1, C_2 и C_3, которые на диаграмме (рис. 8) закрашены.

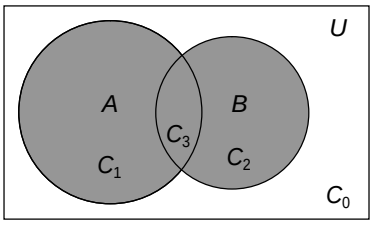


Рис. 8. Объединение множеств $A \cup B$

Другими словами, *объединение множеств* A и B обозначается $A \cup B$ и содержит элементы как множества A, так и множества B, причем общие элементы входят по одному разу.

Логически операция объединения двух множеств характеризуется так: элемент x принадлежит множеству A *или* множеству B. То, что x принадлежит A или B или им обоим, выражается формулой

$$x \in A \cup B \text{ то же самое, что } (x \in A) \vee (x \in B),$$

где $\vee$ — символ *логической связки* «*или*», которая называется *дизъюнкцией*.

Обратите внимание, что логическая связка *или* относится именно к *свойствам* какого-то объекта x: объект x обладает свойствами A *или* B.

С точки зрения логики вместо одной *предметной переменной* x, принимающей значения 1, 2, …, 12, удобно для элементов множеств A и B ввести две *логические переменные* a и b. Областью определения a и b будут уже не натуральные числа, а только два *логических значения*: 1 для *истины* и 0 для *лжи*.

Элементы множества C_0 не принадлежат ни множеству A, ни множеству B, поэтому логические значения для элементов множества C_0 будут $a = 0$ и $b = 0$.

Элементы C_1 принадлежат A и не принадлежат B, поэтому $a = 1$ и $b = 0$.

Элементы C_2 не принадлежат A и принадлежат B, для них $a = 0$ и $b = 1$.

Элементы C_3 принадлежат как A, так и B, для них $a = 1$ и $b = 1$.

Учитывая, что объединение множеств $A \cup B$ состоит из трех множеств C_1, C_2 и C_3, оформляем таблицу 9, которую называют *таблицей истинности*.

Другими словами, *дизъюнкция* равна 0 тогда и только тогда, когда обе логические переменные равны 0.

Таблица 9

Таблица истинности дизъюнкции

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

Между таблицей истинности и диаграммой Эйлера — Венна существует взаимно однозначное соответствие:

1) число единиц в последнем столбце совпадает с числом закрашенных областей на диаграмме;

2) четыре комбинации логических переменных a и b отвечают четырем областям C_0, C_1, C_2 и C_3.

5°. **Операция пересечения множеств**

Пересечением множеств A и B называется множество $A \cap B$, содержащее элементы, входящие сразу в оба множества (рис. 10):

$$A \cap B = \{1, 2, 3, 4\} \cap \{1, 5, 6\} = \{1\} = C_3.$$

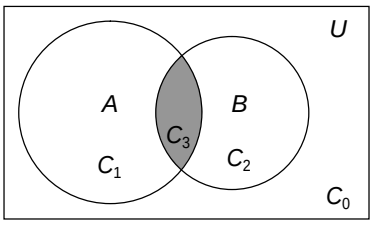


Рис. 10. Пересечение множеств $A \cap B$

Другими словами, *пересечение множеств* A и B обозначается $A \cap B$ и содержит только элементы, общие для множества A и B, по одному разу.

То, что x принадлежит сразу обоим множествам A *и* B, записывается так:

$$x \in A \cap B \text{ то же самое, что } (x \in A) \wedge (x \in B),$$

где $\wedge$ — символ *логической связки* «*и*», или *конъюнкции*.

Другими словами, *конъюнкция* равна 1 тогда и только тогда, когда обе логические переменные равны 1, как показано в таблице истинности 11 ниже.

Логическая связка *и* относится именно к *свойствам* какого-то объекта x: объект x обладает свойствами A *и* B.

Таблица 11

Таблица истинности конъюнкции

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

6°. **Операция дополнения множества**

Дополнением множества A называется множество $\overline{A}$, содержащее элементы, не входящие во множество A (см. рис. 12):

$$\overline{A} = \{5, 6, 7, 8, 9, 10, 11, 12\} = C_0 \cup C_2.$$

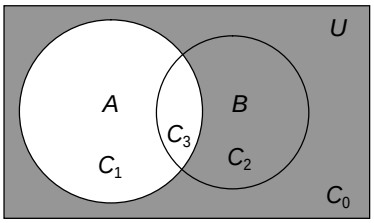


Рис. 12. Дополнение множества A

Другими словами, *дополнение множества* A обозначается $\overline{A}$ и содержит только те элементы универсального множества U, которых нет в A.

Запишем то, что x принадлежит дополнению множества A:

$$x \in \overline{A} \text{ то же самое, что } \neg(x \in A),$$

где $\neg$ — символ *логической связки* «*не*», которая называется *отрицанием*.

Другими словами, *отрицание* равно 1 тогда и только тогда, когда логическая переменная равна 0.

Логическая связка *не* относится именно к свойствам какого-то объекта x: объект x *не* обладает свойством A.

Очевидно, что точно так же можно определить дополнение множества B (или отрицание свойства B), и математики считают, что это то же самое определение.

Таблица 13

Таблица истинности отрицания

a	$\neg a$
0	1
1	0

7°. **Операция импликации множеств**

Импликацией множеств A и B называется множество $A \to B$, содержащее все элементы, входящие в B и не входящие в A (см. рис. 14):

$$A \to B = \{1, 2, 3, 4\} \to \{1, 5, 6\} = \{1, 5, 6, 7, 8, 9, 10, 11, 12\} = C_2 \cup C_3 \cup C_4.$$

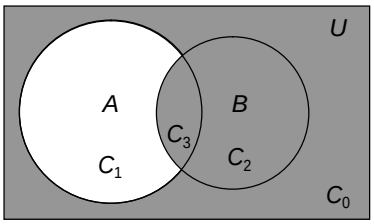


Рис. 14. Импликация множеств $A \to B$

Импликация множеств A и B обозначается $A \to B$ и содержит все элементы универсального множества U без элементов, принадлежащих только A.

Элемент x не принадлежит только множеству A без множества B:

$x \in A \to B$ то же самое, что *если* $(x \in A)$, *то* $(x \in B)$.

где $\to$ — символ *логической связки* «*если… то*», которая называется *импликацией*.

Другими словами, *импликация* равна 0 тогда и только тогда, когда *посылка* — логическая переменная a — равна 1, а *следствие* — логическая переменная b — равна 0 (см. табл. 15).

Таблица 15

Таблица истинности импликации

a	b	$a \sim b$
0	0	1
0	1	1
1	0	0
1	1	1

8°. **Основные законы и соотношения**

Основные *теоретико-множественные законы, или законы логики*, позволяют переписать формулу по-другому, при этом новая формула имеет те же значения истинности, что и исходная, т. е. она *равносильна* исходной формуле.

Законы будем записывать сразу на двух языках: сначала на языке теории множеств, затем на языке логики.

1. *Законы идемпотентности*.

$$A \cup A = A,\ A \cap A = A,$$
$$a \vee a = a,\ a \wedge a = a.$$

2. *Законы коммутативности*.

$$A \cup B = B \cup A,\ A \cap B = B \cap A,$$
$$a \vee b = b \vee a,\ a \wedge b = b \wedge a.$$

3. *Законы ассоциативности*.

$$(A \cup B) \cup C = A \cup (B \cup C),\ (A \cap B) \cap C = A \cap (B \cap C),$$
$$(a \vee b) \vee c = a \vee (b \vee c),\ (a \wedge b) \wedge c = a \wedge (b \wedge c).$$

4. *Законы дистрибутивности*.

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C),\ A \cap (B \cup C) = (A \cap B) \cup (A \cap C),$$
$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c),\ a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c).$$

5. *Законы де Моргана*.

$$\overline{A \cup B} = \overline{A} \cap \overline{B},\ \overline{A \cap B} = \overline{A} \cup \overline{B},$$
$$\neg(a \vee b) = \neg a \wedge \neg b,\ \neg(a \wedge b) = \neg a \vee \neg b.$$

6. *Законы нуля и единицы*.

$$A \cup \overline{A} = U,\ A \cap \overline{A} = \varnothing,\ A \cap U = A,\ A \cap \varnothing = \varnothing,\ A \cup U = U,\ A \cup \varnothing = A,$$
$$a \vee \neg a = 1,\ a \wedge \neg a = 0,\ a \wedge 1 = a,\ a \wedge 0 = 0,\ a \vee 1 = 1,\ a \vee 0 = a.$$

7. *Закон двойного отрицания*.

$$\overline{\overline{A}} = A,$$
$$\neg\neg a = a.$$

Так же запишем основные соотношения.

8. Импликация выражается через объединение и дополнение:

$$A \to B = \overline{A} \cup B,$$
$$a \to b = \neg a \vee b.$$

9. Обобщенные законы де Моргана.

$$\overline{A \cup B \cup C} = \overline{A} \cap \overline{B} \cap \overline{C},\ \overline{A \cap B \cap C} = \overline{A} \cup \overline{B} \cup \overline{C},$$
$$\neg(a \vee b \vee c) = \neg a \wedge \neg b \wedge \neg c,\ \neg(a \wedge b \wedge c) = \neg a \vee \neg b \vee \neg c.$$

С помощью *полных* диаграмм Эйлера — Венна доказывают законы теоретико-множественных операций. Докажем первый закон ассоциативности (рис. 16) $(A \cup B) \cup C = A \cup (B \cup C)$.

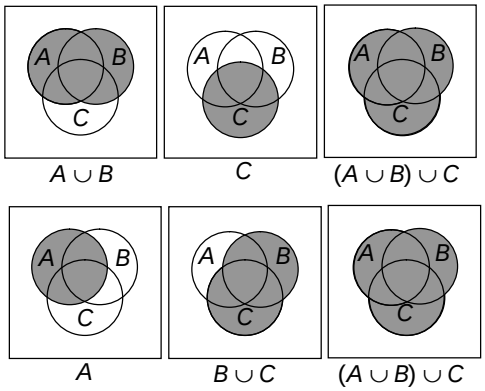


Рис. 16. В верху — множества $A \cup B$, C и $(A \cup B) \cup C$, в низу — A, $B \cup C$ и $(A \cup B) \cup C$. Множества $(A \cup B) \cup C$ и $(A \cup B) \cup C$ равны

Законы также доказывают с помощью таблиц истинности, доказывая, что значения формул совпадают. Докажем первый закон ассоциативности с помощью таблицы 17, в которой закрашенные столбцы совпадают.

Таблица 17

Таблица истинности закона ассоциативности

$(a \vee b) \vee c = a \vee (b \vee c)$

a	b	c	$a \vee b$	$(a \vee b) \vee c$	$b \vee c$	$a \vee (b \vee c)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

9°. **Упражнения**

1. Докажите с помощью диаграмм Эйлера — Венна первый закон де Моргана.

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

2. Докажите с помощью таблиц истинности второй закон де Моргана.

$$\neg(a \wedge b) = \neg a \vee \neg b$$

3. Докажите с помощью таблиц истинности формулу выражения импликации через объединение и дополнение.

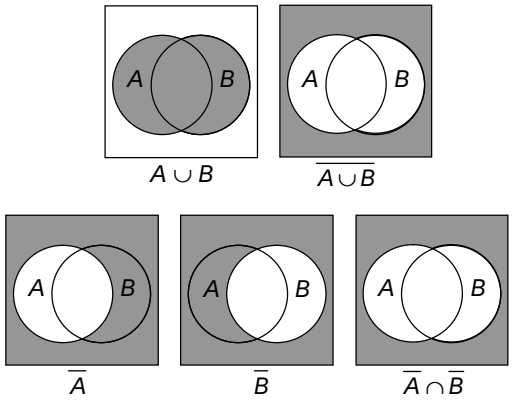
$$a \to b = \neg a \vee b$$

4. Докажите с помощью таблиц истинности первый обобщенный закон де Моргана.

$$\neg(a \vee b \vee c) = \neg a \wedge \neg b \wedge \neg c$$

Решения

1.



2.

a	b	$a \wedge b$	$\neg(a \wedge b)$	$\neg a$	$\neg b$	$\neg a \vee \neg b$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

3.

a	b	$a \to b$	$\neg a$	$\neg a \vee b$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	0	1

4.

a	b	c	$a \vee b$	$(a \vee b \vee c)$	$\neg(a \vee b \vee c)$	$\neg a$	$\neg b$	$\neg c$	$\neg a \wedge \neg b$	$\neg a \wedge \neg b \wedge \neg c$
0	0	0	0	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0	1	0	0
0	1	1	1	1	0	1	0	0	0	0
1	0	0	1	1	0	0	1	1	0	0
1	0	1	1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0	1	0	0
1	1	1	1	1	0	0	0	0	0	0

2. Алгоритмы

Алгоритм 1. Когда выражение истинно.

Чтобы узнать, при каких значениях переменной логическое выражение истинно, нужно просто подставить в выражение все значения переменной и вычислить значение логического выражения: истина (1) или ложь (0).

Алгоритм 2. Переход к равносильной формуле, или равносильному выражению.

1. Если имеется отрицание выражения, стоящего в скобках, для упрощения выражения используем соответствующий закон де Моргана.

2. Если имеется двойное отрицание, для упрощения выражения используем закон двойного отрицания.

Алгоритм 3. Поиск выражения, соответствующего таблице истинности.

Путем последовательного перебора проверяются все строки таблицы на соответствие выражению.

Алгоритм 4. Изучение выражения, включающего импликацию.

Если в выражение входит импликация, например, $А \to Б$, то можно использовать ее основное свойство, заключающееся в том, что импликация ложна только в одном случае: когда посылка А истинна, а следствие Б ложно (см. табл. 15 раздела 1).

Алгоритм 5. Изучение выражения, включающего дизъюнкцию.

Если дизъюнкция соединяет два высказывания, например, А ИЛИ Б, то дизъюнкция истинна тогда и только тогда, когда истинно либо А, либо Б, либо и А и Б (см. табл. 9 раздела 1).

Это определение не всегда удобно на практике, поскольку некоторые наборы переменных могут учитываться по два-три раза.

Чтобы исключить повторение наборов переменных, например, если нужно подсчитать количество наборов переменных, на которых выражение истинно, то нужно перебрать следующие два варианта:

1) истинно А;
2) истинно Б и ложно А.

Алгоритм 6. Поисковый запрос в Интернете.

По поисковым запросам в Интернете, которые делаются с использованием логических операций «ИЛИ» и «И» (т. е. дизъюнкции и конъюнкции), получается количество страниц, которое можно сравнить друг с другом при помощи операций объединения и пересечения множеств.

Сделать это можно тремя способами, с помощью трех средств:

1) логических рассуждений;
2) диаграмм Эйлера-Венна;
3) таблиц истинности.

3. Задачи

1°. Отношения и логические операции

Задача 2006.A9

Для какого из указанных значений числа X истинно высказывание:

$(X > 4) \vee ((X > 1) \to (X > 4))$?

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

Задача 2007.A9

Для какого числа X истинно высказывание

$((X > 3) \vee (X < 3)) \to (X < 1)$?

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

Задача 2008.A9

Для какого из указанных значений числа X истинно высказывание

$((X < 5) \to (X < 3)) \wedge ((X < 2) \to (X < 1))$?

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

Задача 2009.A7

Для какого из указанных значений X истинно высказывание

$\neg((X > 2) \to (X > 3))$?

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

Задача 2007.B2

Каково наибольшее целое число X, при котором истинно высказывание

$(90 < X \cdot X) \to (X < (X - 1))$?

Задача 2009.B4

Каково наибольшее целое число X, при котором истинно высказывание

$(50 < X \cdot X) \to (50 > (X + 1) \cdot (X + 1))$?

2°. Логические формулы

Задача 2006.А10

Какое логическое выражение равносильно выражению $\neg(\neg A \vee B) \vee \neg C$?

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

Задача 2007.А10

Какое логическое выражение равносильно выражению $\neg(A \wedge B) \wedge \neg C$?

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

Задача 2008.А10

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

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

Задача 2009.А8

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

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

3°. Таблицы истинности

Задача 2006.А11

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

Дан фрагмент таблицы истинности выражения F:

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

Какое выражение соответствует F?

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

Задача 2007.А11

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

Дан фрагмент таблицы истинности выражения F:

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

Какое выражение соответствует *F*?

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

Задача 2008.А11

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

Дан фрагмент таблицы истинности выражения *F*:

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

Какое выражение соответствует *F*?

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

Задача 2009.А9

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

Дан фрагмент таблицы истинности выражения *F*:

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

Какое выражение соответствует *F*?

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

Задача 2006.B2

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

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

ложно.

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

Задача 2008.B2

Сколько различных решений имеет уравнение

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

где *K*, *L*, *M*, *N* — логические переменные?

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

4°. Фильтрация запросов

Задача 2006.A16

Ниже в табличной форме представлен фрагмент базы данных о результатах тестирования учащихся (используется стобалльная шкала):

Фамилия	Пол	Математика	Русский язык	Химия	Информатика	Биология
Аганян	ж	82	56	46	32	70
Воронин	м	43	62	45	74	23
Григорчук	м	54	74	68	75	83
Роднина	ж	71	63	56	82	79
Сергеенко	ж	33	25	74	38	46
Черепанова	ж	18	92	83	28	61

Сколько записей в данном фрагменте удовлетворяют условию

«Пол = 'м' ИЛИ Химия > Биология»?

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

Задача 2009.А14

Результаты тестирования представлены в таблице:

Фамилия	Пол	Математика	Русский язык	Химия	Информатика	Биология
Аганян	ж	82	56	46	32	70
Воронин	м	43	62	45	74	23
Григорчук	м	54	74	68	75	83
Роднина	ж	71	63	56	82	79
Сергеенко	ж	33	25	74	38	46
Черепанова	ж	18	92	83	28	61

Сколько записей в ней удовлетворяют условию

«Пол='ж' ИЛИ Химия > Биология»?

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

Задача 2006.В8

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

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

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

Задача 2007.В8

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

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

А	волейбол | баскетбол | подача
Б	волейбол | баскетбол | подача | блок
В	волейбол | баскетбол
Г	волейбол & баскетбол & подача

Задача 2008.B8

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

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

1	Физкультура
2	физкультура & подтягивания & отжимания
3	физкультура & подтягивания
4	физкультура | фитнесс

Задача 2009.B10

В таблице приведены запросы к поисковому серверу. Расположите номера запросов в порядке возрастания количества страниц, которые найдет поисковый сервер по каждому запросу.

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

1	принтеры & сканеры & продажа
2	принтеры & продажа
3	принтеры | продажа
4	принтеры | сканеры | продажа

4. Ответы

1°. Отношения и логические операции

Задача 2006.A9. 1) 1.
Задача 2007.A9. 3) 3.
Задача 2008.A9. 2) 2.
Задача 2009.A7. 3) 3.
Задача 2007.B2. 9.
Задача 2009.B4. 7.

2°. Логические формулы

Задача 2006.A10. 1) $(A \wedge \neg B) \vee \neg C$.
Задача 2007.A10. 2) $(\neg A \vee \neg B) \wedge \neg C$.
Задача 2008.A10. 4) $\neg A \wedge B \wedge \neg C$.
Задача 2009.A8. 3) $A \wedge B \wedge \neg C$.

3°. Таблицы истинности

Задача 2006.A11. 3) $X \vee Y \vee Z$.
Задача 2007.A11. 2) $X \wedge Y \wedge \neg Z$.
Задача 2008.A11. 1) $X \vee \neg Y \vee Z$.
Задача 2009.A9. 4) $\neg X \vee \neg Y \vee \neg Z$.
Задача 2006.B2. 1100.
Задача 2008.B2. 10.

4°. Фильтрация запросов

Задача 2006.A16. 4) 4.
Задача 2009.A14. 1) 5.
Задача 2006.B8. АГБВ.
Задача 2007.B8. ГВАБ.
Задача 2008.B8. 2314.
Задача 2009.B10. 1234.

5. Решения

1°. Отношения и логические операции

Замечание. Задача сложна, когда приходится перебирать бесконечное множество значений переменной.

Рекомендации. 1. При работе с текстом задания полезно понять, какая логическая операция находится на самом верхнем уровне.

2. Вариант используемого решения задачи зависит от склонностей учащегося.

3. Возможны трудности при решении задания из второй части (B) ЕГЭ.

Задача 2006.A9

Ответ: 1) 1.

Решение. Подставим по очереди все предлагаемые в ответах значения числа X в формулу и подсчитаем ее логическое значение. Начнем с 1.

$(1 > 4) \vee ((1 > 1) \to (1 > 4)) \Rightarrow 0 \vee (0 \to 0) \Rightarrow 0 \vee 1 \Rightarrow 1$.

Итак, при $X = 1$ высказывание истинно.

Для проверки проверим остальные значения.

$X = 2$: $(2 > 4) \vee ((2 > 1) \to (2 > 4)) \Rightarrow 0 \vee (1 \to 0) \Rightarrow 0 \vee 0 \Rightarrow 0$.

$X = 3$: $(3 > 4) \vee ((3 > 1) \to (3 > 4)) \Rightarrow 0 \vee (1 \to 0) \Rightarrow 0 \vee 0 \Rightarrow 0$.

$X = 4$: $(4 > 4) \vee ((4 > 1) \to (4 > 4)) \Rightarrow 0 \vee (1 \to 0) \Rightarrow 0 \vee 0 \Rightarrow 0$.

Задача 2007.A9

Ответ: 3) 3.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2008.A9

Ответ: 2) 2.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2009.A7

Ответ: 3) 3.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2007.B2

Решение 1. Будем перебирать все значения X, начиная с 1, и вычислять для них значения нашей импликации.

$$(90 < 1{\cdot}1) \to (1 < (1-1)) \Rightarrow (90 < 1) \to (1 < 0) \Rightarrow 0 \to 0 = 1.$$
$$(90 < 2{\cdot}2) \to (2 < (2-1)) \Rightarrow (90 < 4) \to (2 < 1) \Rightarrow 0 \to 0 = 1.$$
$$(90 < 3{\cdot}3) \to (3 < (3-1)) \Rightarrow (90 < 9) \to (3 < 2) \Rightarrow 0 \to 0 = 1.$$
$$(90 < 4{\cdot}4) \to (4 < (4-1)) \Rightarrow (90 < 16) \to (4 < 3) \Rightarrow 0 \to 0 = 1.$$
$$(90 < 5{\cdot}5) \to (5 < (5-1)) \Rightarrow (90 < 25) \to (5 < 4) \Rightarrow 0 \to 0 = 1.$$
$$(90 < 6{\cdot}6) \to (6 < (6-1)) \Rightarrow (90 < 36) \to (6 < 5) \Rightarrow 0 \to 0 = 1.$$
$$(90 < 7{\cdot}7) \to (7 < (7-1)) \Rightarrow (90 < 49) \to (7 < 6) \Rightarrow 0 \to 0 = 1.$$
$$(90 < 8{\cdot}8) \to (8 < (8-1)) \Rightarrow (90 < 64) \to (8 < 7) \Rightarrow 0 \to 0 = 1.$$
$$(90 < 9{\cdot}9) \to (9 < (9-1)) \Rightarrow (90 < 81) \to (9 < 8) \Rightarrow 0 \to 0 = 1.$$
$$(90 < 10{\cdot}10) \to (10 < (10-1)) \Rightarrow (90 < 100) \to (10 < 9) \Rightarrow 1 \to 0 = 0.$$

При дальнейшем увеличении X значения аргументов импликации не изменятся. Поэтому $X = 9$ — наибольшее, при котором импликация истинна.

Решение 2. Очевидно, что значения второго аргумента импликации $(X < (X-1))$ всегда ложно. При втором ложном аргументе импликация будет истинна, если первый аргумент $(90 < X{\cdot}X)$ ложен, и ложна, когда он истинен.

При всех $X \leqslant 9$ первый аргумент импликации ложен, а вся импликация истинна. При $X > 9$ — наоборот.

Задача 2009.B4

Решение 1. Будем перебирать все значения X, начиная с 1.

$$(50 < 1{\cdot}1) \to (50 > (1+1){\cdot}(1+1)) \Rightarrow (50 < 1) \to (50 > 4) \Rightarrow 0 \to 1 = 1.$$
$$(50 < 2{\cdot}2) \to (50 > (2+2){\cdot}(2+2)) \Rightarrow (50 < 4) \to (50 > 16) \Rightarrow 0 \to 1 = 1.$$
$$(50 < 3{\cdot}3) \to (50 > (3+3){\cdot}(3+3)) \Rightarrow (50 < 9) \to (50 > 36) \Rightarrow 0 \to 1 = 1.$$
$$(50 < 4{\cdot}4) \to (50 > 8{\cdot}8) \Rightarrow (50 < 16) \to (50 > 64) \Rightarrow 0 \to 0 = 1.$$
$$(50 < 5{\cdot}5) \to (50 > 10{\cdot}10) \Rightarrow (50 < 25) \to (50 > 100) \Rightarrow 0 \to 0 = 1.$$
$$(50 < 6{\cdot}6) \to (50 > 12{\cdot}12) \Rightarrow (50 < 36) \to (50 > 144) \Rightarrow 0 \to 0 = 1.$$
$$(50 < 7{\cdot}7) \to (50 > 14{\cdot}14) \Rightarrow (50 < 49) \to (50 > 14{\cdot}14) \Rightarrow 0 \to 0 = 1.$$
$$(50 < 8{\cdot}8) \to (50 > 16{\cdot}16) \Rightarrow (50 < 64) \to (50 > 16{\cdot}16) \Rightarrow 1 \to 0 = 0.$$

При дальнейшем увеличении X значения аргументов импликации не изменится. Поэтому $X = 7$ — наибольшее, при котором импликация истинна.

Решение 2. Сначала вычислим значения данного высказывания при $X = 1$.

$$(50 < 1\cdot 1) \to (50 > (1+1)\cdot(1+1)) \Rightarrow (50 < 1) \to (50 > 4) \Rightarrow 0 \to 1 = 1.$$

Получилась истина.

Очевидно, что при любых положительных целых X значения аргументов импликации $(50 < X\cdot X)$ и $(50 > (X+1)\cdot(X+1))$ всегда противоположны по своему значению.

Далее, при $X \leqslant 7$ первый аргумент импликации ложен, поэтому второй аргумент истинен и вся импликация истинна.

А при всех $X > 7$ первый аргумент импликации $(50 < X\cdot X)$ истинен и вся импликация ложна.

2°. Логические формулы

Замечание. Задачи на законы де Моргана.

Рекомендации. 1. При работе с текстом задания нужно увидеть, к какой части формулы применять закон де Моргана.

2. Вариантов решения нет.

3. Ответы трудностей не вызывают.

Задача 2006.А10

Ответ: 1) $(A \wedge \neg B) \vee \neg C$.

Решение. Упростим данное в условии задания выражение $\neg(\neg A \vee B) \vee \neg C$. Для этого воспользуемся первым законом де Моргана $\neg(A \vee B) = \neg A \wedge \neg B$:

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

Теперь воспользуемся законом двойного отрицания $\neg\neg A = A$.

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

Задача 2007.А10

Ответ: 2) $(\neg A \vee \neg B) \wedge \neg C$.

Решение. Аналогично предыдущей задаче.

Задача 2008.А10

Ответ: 4) $\neg A \wedge B \wedge \neg C$.

Решение. Упростим данное в условии задания выражение, воспользуемся первым обобщенным законом де Моргана $\neg(A \vee B \vee C) = \neg A \wedge \neg B \wedge \neg C$.

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

Теперь воспользуемся законом двойного отрицания $\neg\neg A = A$.

$$\neg A \wedge \neg\neg B \wedge \neg C = \neg A \wedge B \wedge \neg C.$$

Задача 2009.А8

Ответ: 4) $A \wedge B \wedge \neg C$.

Решение. Совершенно аналогично задаче 2006.А10.

3°. **Таблицы истинности**

Замечание. Задачи считаются принадлежащими к среднему уровню сложности.

Рекомендации. 1. Задачи на таблицы истинности.

2. Нужно просто перебрать все варианты.

3. Ответы трудностей не вызывают.

Задача 2006.А11

Ответ: 3) $X \vee Y \vee Z$.

Решение 1. Подставим все значения аргументов X, Y, Z из данной таблицы по очереди в каждое предложенное выражение для функции F и подсчитаем ее логические значения.

1) Начнем с $\neg X \vee \neg Y \vee \neg Z$.

$\neg 0 \vee \neg 0 \vee \neg 0 = 1 \vee 1 \vee 1 = 1$. Не соответствует таблице.

2) Подставим в выражение $X \wedge \neg Y \wedge \neg Z$.

$0 \vee \neg 0 \vee \neg 0 = 0 \vee 1 \vee 1 = 1$. Не соответствует таблице.

3) Подставим в выражение $X \vee Y \vee Z$.

$0 \vee 0 \vee 0 = 0$. Соответствует таблице.

$1 \vee 1 \vee 0 = 1$. Соответствует таблице.

$1 \vee 0 \vee 0 = 1$. Соответствует таблице.

Итак, F соответствует выражение $X \vee Y \vee Z$.

4) Для проверки подставим в $X \wedge Y \wedge Z$.

$0 \wedge 0 \wedge 0 = 0$. Соответствует таблице.

$1 \wedge 1 \wedge 0 = 0$. Не соответствует таблице.

Решение 2. Подставим значения аргументов X, Y, Z из первой строки данной таблицы в каждое предложенное выражение для F и подсчитаем их логические значения.

1) $\neg X \vee \neg Y \vee \neg Z$: $\neg 0 \vee \neg 0 \vee \neg 0 = 1 \vee 1 \vee 1 = 1$. Не соответствует таблице.

2) $X \wedge \neg Y \wedge \neg Z$: $0 \vee \neg 0 \vee \neg 0 = 0 \vee 1 \vee 1 = 1$. Не соответствует таблице.

3) $X \vee Y \vee Z$: $0 \vee 0 \vee 0 = 0$. Соответствует таблице.

4) $X \wedge Y \wedge Z$: $0 \wedge 0 \wedge 0 = 0$. Соответствует таблице.

Теперь подставим значения аргументов X, Y, Z из второй строки таблицы только в третье и в четвертое выражение.

3) $X \vee Y \vee Z$: $1 \vee 1 \vee 0 = 1$. Соответствует таблице.

4) $X \wedge Y \wedge Z$: $1 \wedge 1 \wedge 0 = 0$. Не соответствует таблице.

Для проверки подставим значения аргументов X, Y, Z из третьей строки таблицы в третье выражение.

3) $X \vee Y \vee Z$: $1 \vee 0 \vee 0 = 1$. Соответствует таблице.

Итак, F соответствует выражение $X \vee Y \vee Z$.

Задача 2007.A11

Ответ: 2) $X \wedge Y \wedge \neg Z$.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2008.A11

Ответ: 1) $X \vee \neg Y \vee Z$.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2009.A9

Ответ: 4) $\neg X \vee \neg Y \vee \neg Z$.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2006.B2

Ответ: 1100.

Решение 1. Переберем все 16 значений для 4 переменных.

K	L	M	N	$K \vee M$	$\neg L$	$M \vee \neg L$	$M \vee \neg L \vee N$	$(K \vee M) \to (M \vee \neg L \vee N)$
0	0	0	0	0	1	1	1	1
0	0	0	1	0	1	1	1	1
0	1	0	0	0	0	0	0	1
0	1	0	1	0	0	0	1	1
0	0	1	0	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	1	1	0	1	0	1	1	1
0	1	1	1	1	0	1	1	1
1	0	0	0	1	1	1	1	1
1	0	0	1	1	1	1	1	1
1	1	0	0	1	0	0	0	0
1	1	0	1	1	0	0	1	1
1	0	1	0	1	1	1	1	1
1	0	1	1	1	1	1	1	1
1	1	1	0	1	0	1	1	1
1	1	1	1	1	0	1	1	1

Данное в условии задания выражение ложно только в одном случае: при $K = 1$, $L = 1$, $M = 0$, $N = 0$.

Решение 2. Перебирать все 16 значений для 4 переменных не будем.

Операция самого верхнего уровня в выражении, данном задании,— это импликация $\to$. А импликация ложна только при одном условии: когда левая часть равна 1, а правая — 0.

Правая часть состоит из одних дизъюнкций, поэтому она равна 0, когда все ее составляющие тоже равны 0. Получаем, что $M = 0$, $L = 1$, $N = 0$.

Левая часть теперь $K \vee 0$ и равна 1 только при $K = 1$.

Задача 2008.B2

Ответ: 10.

Решение 1. Переберем все 16 значений для 4 переменных.

K	L	M	N	$K \vee L$	$L \wedge M$	$L \wedge M \wedge N$	$(K \vee L) \to (L \wedge M \wedge N)$
0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1
0	1	0	0	1	0	0	0
0	1	0	1	1	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	1
0	1	1	0	1	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	1	0	1	1	0	0
1	1	1	1	1	1	1	1

Данное в условии задания выражение ложно только в 10 случаях, поскольку в столбце таблицы истинности, соответствующему этому выражению, стоит 10 нулей.

Решение 2. Импликация ложна только при одном условии: когда левая часть равна 1, а правая — 0.

Левая часть равна 1 в трех случаях.

1) $K = L = 1$. Тогда правая часть $1 \wedge M \wedge N$ равна 0 в трех случаях: $M = N = 0$; $M = 0$, $N = 1$; $M = 1$, $N = 0$.

2) $K = 1$, $L = 0$. Тогда правая часть $0 \wedge M \wedge N$ равна 0 при любых значения M и N, т. е. в четырех случаях.

3) $K = 0$, $L = 1$. Аналогично 1) имеем три таких же случая: $M = N = 0$; $M = 0$, $N = 1$; $M = 1$, $N = 0$. Итак, получаем 10 случаев.

4°. **Фильтрация запросов**

Замечание. Нужно хорошо представлять себе, что союз «и» в запросах означает не объединение, а пересечение множеств страниц.

Рекомендации. 1. Нужно внимательно смотреть, какие логические операции используются в запросах, приведенных в условии заданий.

2. Вариант решения можно подобрать по вкусу. Рекомендуется использовать вариант, который кажется Вам самым коротким.

3. Ответы трудностей не вызывают.

Задача 2006.А16

Ответ: 4) 4.

Решение 1. Условию «Пол = 'м' ИЛИ Химия > Биология» удовлетворяют записи, которые удовлетворяют или условию «Пол = 'м'», или условию «Химия > Биология», или им обоим.

Сначала рассмотрим условие Пол = 'м'. Условию Пол = 'м' отвечают 2 записи.

Поскольку у нас дизъюнкция ИЛИ, то осталось рассмотреть условие Пол = 'ж' при условии Химия > Биология. Этим двум условиям сразу удовлетворяют тоже 2 записи.

Всего 4 записи.

Решение 2. Сначала рассмотрим условие Химия > Биология. Условию Химия > Биология отвечают 3 записи.

Поскольку у нас дизъюнкция ИЛИ, то осталось рассмотреть условие Химия $\leqslant$ Биология при условии Пол = 'м'. Этим двум условиям сразу удовлетворяют только 1 запись.

Всего 4 записи.

Задача 2009.А14

Ответ: 1) 5.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2006.B8

Ответ: АГБВ.

Решение 1. Меньше всего множество страниц получится при запросе, в котором больше всего конъюнкций (логических операций «И»),— в запросе А три конъюнкции, и по нему будут найдены страницы со всеми четырьмя словами «разведение», «содержание», «меченосцы», «сомики» из запросов.

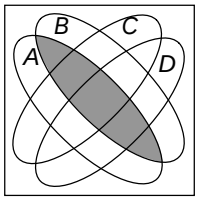
На втором месте по количеству будет множество страниц при запросе, в котором количество конъюнкций на одну меньше,— в запросе Г две конъюнкции, и количество страниц получится больше, т. к. на них будут обязательно присутствовать только три слова из четырех.

На третьем месте по количеству стоит множество страниц, получающееся при запросе, в котором только одна конъюнкция, но нет дизъюнкций (логических операций «ИЛИ»),— это запрос Б, по которому будут найдены страницы только с двумя обязательными словами «содержание» и «меченосцы».

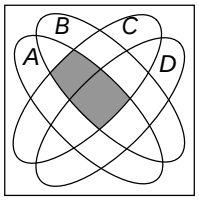
Наконец, больше всего множество страниц получится при запросе, в котором присутствует дизъюнкция,— при запросе В будут найдены как страницы со словами «содержание» и «меченосцы», так и страницы с одним словом «сомики».

Решение 2. Нарисуем на диаграммах Эйлера-Венна 4 множества — страницы, получаемые по 4 элементарным запросам: «разведение», «содержание», «меченосцы» и «сомики», причем множества рисуем так, чтобы имелись все возможные варианты их пересечений. Затем закрасим множества, которые соответствуют четырем предложенным в ответах запросам.

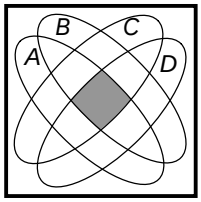
А) разведение & содержание & меченосцы & сомики = *A & B & C & D*



A & B

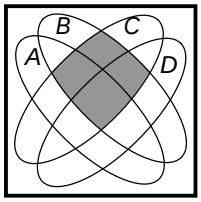


A & B & C



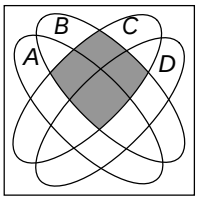
A & B & C & D

Б) содержание & меченосцы = $B \,\&\, C$

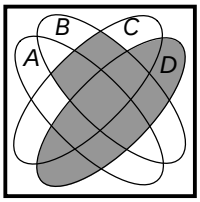


$B \,\&\, C$

В) (содержание & меченосцы) | сомики = $(B \,\&\, C) \mid D$

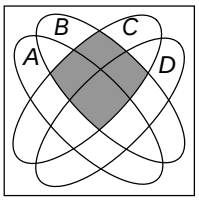


$B \,\&\, C$

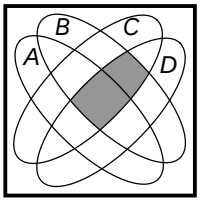


$(B \,\&\, C) \mid D$

Г) содержание & меченосцы & сомики = $B \,\&\, C \,\&\, D$



$B \,\&\, C$



$B \,\&\, C \,\&\, D$

Теперь очевидно, что

$$A \,\&\, B \,\&\, C \,\&\, D \subset B \,\&\, C \,\&\, D \subset B \,\&\, C \subset (B \,\&\, C) \mid D,$$

или А) ⊂ Г) ⊂ Б) ⊂ В).

Решение 3. В обозначениях предыдущего варианта решения составим таблицы истинности, которые соответствуют четырем предложенным в ответах запросам.

Чем больше единиц соответствует множеству, тем оно больше. Если единицы одного множества являются и единицами другого, то первое множество — подмножество второго.

A	*B*	*C*	*D*	*B* & *C*	*B* & *C* & *D*	*A* & *B* & *C* & *D*	(*B* & *C*) | *D*
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	1	0	0	1
0	1	1	1	1	1	0	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	1
1	1	1	0	1	0	0	1
1	1	1	1	1	1	1	1

Получаем $A \,\&\, B \,\&\, C \,\&\, D \subset B \,\&\, C \,\&\, D \subset B \,\&\, C \subset (B \,\&\, C) \mid D$, или А) ⊂ Г) ⊂ Б) ⊂ В).

Задача 2007.B8

Ответ: ГВАБ.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2008.B8

Ответ: 2314.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2009.B10

Ответ: 1234.

Решение. Совершенно аналогично предыдущей задаче.

§ 2. Поиск закономерностей

1. Теория

1°. Файл

Одним из основных понятий на компьютере является концепция *файла*.

Файл — форма организации информации на компьютере.

Авторские данные, созданные с помощью приложений, имеют форму файла. Содержанием файла являются данные, по которым файлы классифицируют по *типам*.

Текстовый файл — тип файла, основное содержание которого — текст. *Графический файл* — тип файла, основное содержание которого — графика. *Звуковой файл* — тип файла, основное содержание которого — звук. *Видеофайл* — тип файла, основное содержание которого — видео.

Объем файлов измеряется в байтах, килобайтах и мегабайтах. Файлы хранятся на устройствах постоянной памяти.

При сохранении файл получает *имя*. Отсюда возникло еще одно определение файла.

Файл — именованная область постоянной памяти с данными, записанными в виде байтов.

Имя файла — текстовая строка, которая характеризует файл и позволяет отличить данный файл от других файлов.

В ОС Windows имя файла может иметь практически произвольный вид на любом языке.

Имя файла обычно имеет *расширение*, играющее важную роль при работе на компьютере.

Расширение имени файла — последние символы имени файла, отделенные от остального имени точкой. Если расширение отсутствует, то в имени файла нет ни одной точки.

Основное имя файла — имя файла без расширения и без отделяющей расширение точки.

Расширение имени файла отражает тип файла, и поэтому широко используется при работе ОС и прикладных программ.

Некоторые ОС могут не выводить на экран стандартные расширения. Это опасно. Рекомендуется всегда выводить на экран полное имя файла. Одна из причин такой рекомендации — борьба с вирусами.

Например, программы хранятся в файлах с расширением .exe, неформатированные текстовые файлы — в файлах с расширением .txt, текстовые файлы в кодировке Word — .doc. Список стандартных расширений см. в приложениях.

Произвольная форма имени файла имеется только в Windows. Для правильной передачи файлов между разными ОС и в Интернете рекомендуется давать файлам *стандартные имена*.

Прописные и строчные буквы в именах файлов в ОС Windows не различаются, а в ОС Юникс различаются.

Стандартное имя файла, или стандарт 8.3 — имя файла, составленное по следующим двум правилам:

1) основное имя файла имеет длину от 1 до 8 символов, расширение, если есть, от 1 до 3 символов. Основное имя файла передает смысл его содержания, расширение — тип файла;

2) имя файла и расширение состоят только из латинских букв, цифр и некоторых специальных символов, например, подчеркивания, в число которых пробел не входит.

Пример:

lekcija.doc — текстовый файл с лекцией в кодировке Word.

Стандарт 8.3 действует на любом компьютере в любой стране, т. к. в этом стандарте имя файла составляется только из аски-кодов и это имя короткое. Максимальная длина имени файла, отвечающего стандарту 8.3, равна 12 символам: 8 символов может занять основное имя, 1 — точка, 3 — расширение. Оба формата стандартного имени файла см. на рисунке 1. Как правило, стандартное имя имеет в расширении 3 символа.

Рис. 1. Структура стандартного имени файла 8.3 без расширения (слева) и с расширением (справа)

2°. **Файловая система**. **Логический диск**, **форматирование**

На носителе постоянной памяти — на диске — обычно располагается много файлов.

Файловая система — логическая структура размещения файлов на носителе постоянной памяти.

Файловая система может занимать всю постоянную память или только ее часть. В последнем случае на носителе постоянной памяти может размещаться несколько независимых файловых систем.

Логический диск — именованная область носителя постоянной памяти, содержащая одну файловую систему.

Имя логического диска — название диска в виде одной буквы латинского алфавита с последующим двоеточием.

Логический диск называют просто *диском*.

Двоеточие в имени диска необходимо, чтобы отличать его от имени файла. Вставленная в дисковод гибких дисков любая дискета всегда является логическим диском A:. Все другие логические диски, отличные от размещаемого на дискете, имеют имена от C: до Z:. При наличии нескольких дисков последовательность их имен определяются операционной системой.

Форматирование высокого уровня производится отдельно для каждого логического диска. При этом определяется *метка диска*. Изменить метку диска можно после форматирования.

Форматирование высокого уровня — организация на логическом диске файловой системы.

Метка диска — краткое название логического диска в виде произвольной строки текста.

Форматирование высокого уровня достаточно часто используется (например, форматирование дискет) и называется просто *форматированием*.

Не спутайте имя диска с его меткой!

Если на компьютере имеется один жесткий диск с одним логическим диском, то он всегда имеет имя C:.

Если в компьютере два жестких диска и на каждом по одному логическому диску, то всегда один диск имеет имя C:, а другой — D:.

Форматирование среднего уровня — создание на носителе постоянной памяти одного или нескольких логических дисков.

> Программа форматирования среднего уровня — fdisk. Это опасная программа. Если удалить логический диск, то все данные на нем пропадают гораздо быстрее, чем при форматировании высокого уровня.

Жесткие диски и флэш-память можно разбивать на несколько логических дисков, тогда как CD и DVD обычно так не разбиваются.

Если в компьютере имеется только один винчестер, разбитый на несколько дисков, то эти диски всегда имеют имена C:, D: и т. д.

Имена логических дисков назначаются последовательно. Приведем один из стандартных списков логических дисков типичного домашнего компьютера:

1) A: — логический диск на дискете;
2) C:, D: и E: — логические диски на жестком диске;
3) F: — логический диск на флэш-памяти;
4) G: и H: — логические диски на CD (на двух CD-дисководах).

> Другие случаи сложнее. Если каждый из двух винчестеров имеет по несколько дисков, то их имена перемешиваются.
>
> Имена CD следуют за именами винчестеров. Каждый USB-порт имеет отдельное имя, следующее за именами винчестеров и перед именами CD.
>
> Файловая структура или часть файловой структуры, доступная на другом компьютере в *локальной сети*, также имеет отдельное имя. И т. д.

3°. **Директория**. **Дерево директорий**

На постоянных носителях файлов бывает очень много — десятки тысяч. Чтобы работать с таким большим количеством объектов, их необходимо классифицировать, т. е. распределить по группам.

Директория (папка, фолдер) (directory, folder) — именованная группа файлов или директорий.

Имя директории составляется по таким же правилам, как и имя файла, только обычно не имеет расширения.

Структура директорий всегда имеет вид *дерева*.

Поддиректория — директория, которая содержится в другой директории.

Наддиректория (родительская директория) — директория, содержащая другую директорию. Обозначение: две точки . . или стрелка углом ↑.

Корневая директория — самая верхняя директория на логическом диске, которая не является ничьей поддиректорией. Имя корневой директории получается из имени логического диска добавлением бэкслеша, например, C:\.

Содержание директории — список ее файлов и поддиректорий, и, если она не корневая, указатель на наддиректорию.

Дерево директорий — структура директорий логического диска.

Дерево — структура данных, при которой каждое объект, кроме одного, корневого, принадлежит одному надобъекту и не может принадлежать двум надобъектам.

Корневая директория — единственная, имеющаяся на логическом диске после его форматирования.

Форматирование высокого уровня — создание на логическом диске дерева директорий в виде одной корневой директории.

В одной директории полные имена всех ее файлов и директорий должны быть уникальными, без повторений.

На каждом диске имеется свое *дерево директорий* (см. рис. 2), растущее корнем вверх.

На дереве директорий иногда изображаются файлы — листья (см. рис. 2).

Поддерево — любая часть дерева, начинающаяся с какой-нибудь директории, которая является корневой для поддерева (см. рис. 2).

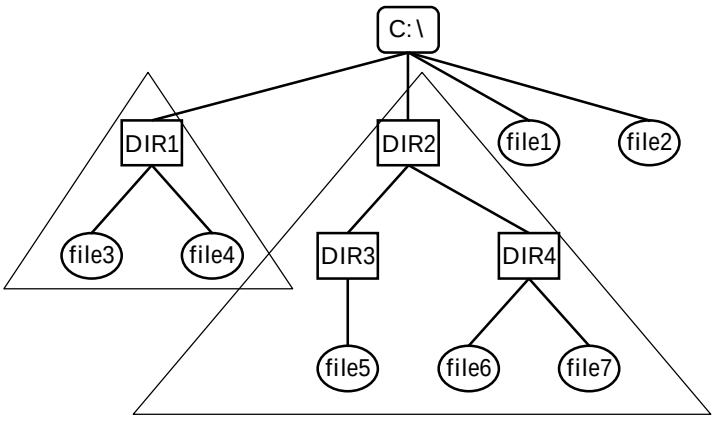


Рис. 2. Простая схема дерева директорий на диске C:.
Здесь C:\ — корневая директория, DIR — директории, file — файлы.
Треугольниками показаны два поддерева из четырех

Иногда бывает нужно записать *полный путь к файлу*.

Полный путь к файлу — это список всех директорий, по которым нужно пройти от корневой директории, чтобы попасть к файлу. Имена директорий отделяются друг от друга и от имени файла бэкслешем \.

Для примера выпишем полные имена всех семи файлов с рисунка 12.

C:\file1
C:\file2
C:\DIR1\file3
C:\DIR1\file4
C:\DIR2\DIR3\file5
C:\DIR2\DIR4\file6
C:\DIR2\DIR4\file7

При создании дерева директорий процесс создания директорий идет сверху вниз. Опишем его типичные шаги.

1. Сначала в корневой директории создаются директории — поддиректории корневой. Это директории второго уровня, если корневая директория находится на первом.

2. В директориях второго уровня создаются поддиректории — директории третьего уровня, и т. д. до тех пор, пока вся информация не будет классифицирована в виде дерева.

При появлении новой информации она добавляется в уже существующее дерево директорий:

1) либо в виде отдельного файла или файлов в какую-нибудь существующую директорию;

2) либо в директории, соответствующей по смыслу, создается новая поддиректория, в которую и записывается поступившая или созданная информация в виде поддерева, для которого эта новая поддиректория является корневой.

4°. **Два уровня глобальной сети**

Глобальные сети из-за своего огромного размера имеют двухуровневую структуру: уровень пользователя, получающего услуги, и уровень профессионала, организующего функционирование сети.

Первый, внешний уровень имеет следующие особенности:

1) в нем работает пользователь глобальной сети;

2) связь со вторым структурным уровнем обычно осуществляется по одному слабому каналу, как показано на рисунке 3.

Конечный пользователь — участник первого, пользовательского, уровня глобальной сети.

Конечный пользователь называется также просто *пользователем, или юзером (user)*.

Наиболее распространенный способ подключения компьютера к глобальным сетям — по телефонной линии. Подключиться к глобальной сети можно и по мобильному телефону через сотовую телефонную сеть.

При подключении по телефонной линии необходим *модем*.

Модем — компьютерное устройство для подключения системного блока к телефонной линии.

Коммутируемый доступ, или Dial-Up (читается «диал-ап») — доступ к глобальной сети по телефонной линии.

Рис. 3. Схемы подключения пользователя к глобальной сети:
а) посредством телефонной связи;
б) данные в сеть поступают по телефонной линии,
а обратно приходят через телевизионный спутник;
в) подключившись к локальной сети;
г) по радио- или спутниковой линии.

Рассмотрим второй, внутренний уровень глобальной сети.

Узел глобальной сети — элемент второго уровня глобальной сети, организация, обеспечивающая работу глобальной сети.

Узел глобальной сети называется также просто *узлом*.

Второй, профессиональный, уровень глобальной сети составляют ее узлы, к которым подключены пользователи. Эти узлы имеет следующие характеристики (см. рис. 4):

1) узлы обычно соединяться друг с другом мощными линиями связи, такими, как оптоволокно на близких расстояниях и двусторонняя спутниковая связь на очень дальних;

2) узел обычно соединен с несколькими другими узлами;

3) соединение узлов друг с другом организовано произвольным образом.

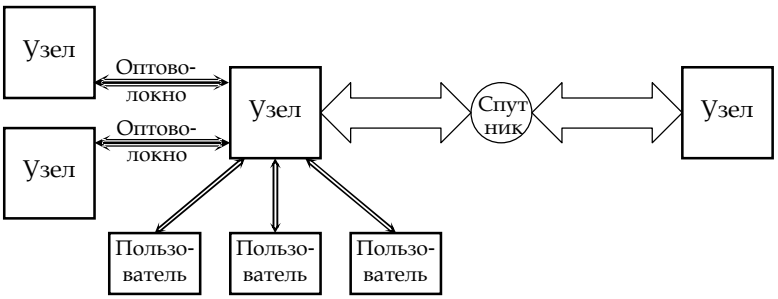


Рис. 4. Схема связи между уровнями глобальной сети

5°. **Адресное пространство**

Логическая структура соединений глобальной сети, которая необходима для передачи информации конечному пользователю, имеет вид иерархии уровней. Такая иерархическая структура связана с необходимостью адресации всех компьютеров, подключенных к сети.

В локальной сети для доступа к локальной станции достаточно знать ее имя. Компьютеров в локальной сети немного, поэтому имена локальных станций определяет администратор сети. В локальной сети подключение пользователя осуществляется при вводе им логина, и на одной локальной станции могут работать разные пользователи (в разное время, конечно). Логин пользователя определяет доступные ему ресурсы локальной сети.

Логин пользователя глобальной сети, наоборот, используется для входа в глобальную сеть и не влияет на доступность ее ресурсов. Кроме того, в глобальную сеть можно войти и без логина.

В глобальную компьютерную сеть входит огромное количество компьютеров, разбросанных по всему миру. Единого администратора у глобальной сети существовать не может. Поэтому для организации связи среди всех компьютеров, подключающихся к глобальной сети, распределяют уникальные *адреса*.

В глобальной сети логин пользователя необязателен. Это означает, что пользователь может войти в глобальную сеть двумя способами:

1) через уже подключенный к сети компьютер. Это компьютер может либо иметь постоянное подключение к глобальной сети в локальной сети со своим постоянным адресом, либо быть подключенным через логин другого пользователя, по сетевой карточке или иным способом;

2) с помощью своего постоянного логина. В этом случае компьютеру пользователя автоматически назначается в глобальной сети либо постоянный адрес в соответствии с его логином, либо временный адрес.

Адрес в сети — уникальная символьная строка, однозначно определяющая компьютер в сети.

Адресное пространство — система адресов глобальной сети. Каждому компьютеру, подключенному или подключившемуся к глобальной сети, назначается уникальный адрес.

Адресное пространство аналогично системе адресов обычной почты. Адресное пространство, как и любая система адресов, всегда является иерархической структурой со всеми вытекающими отсюда последствиями. Более подробно адресное пространство Интернета рассмотрено в следующем разделе.

6°. IP- и доменные адреса

В Интернете каждому подключившемуся компьютеру присваивается уникальный числовой *IP-адрес*.

IP-адрес (читается «ай-пи», internet protocol, межсетевой протокол) — четыре числа диапазона 0—255, образующие адрес компьютера в глобальной сети. При записи адреса эти числа разделяются точками, например, 255.255.255.255.

IP-адрес называется также *IP-номером*.

В IP-адресе четыре числа диапазона 0—255, поэтому к Интернету можно подключить 2^{32}, т. е. более 4 миллиардов, компьютеров.

Кроме системы IP-адресов в Интернете имеется другое, параллельное адресное пространство, состоящее из *доменов*.

Домéн — подразделение адресного пространства Интернета. Домены имеют иерархическую структуру.

Домéнный адрес — адрес из имен доменов, разделенных точками, начиная с нижнего уровня и заканчивая первым уровнем.

Домены первого уровня не относятся к какому-то отдельному компьютеру или узлу, и их имена можно разделить на две группы по следующим признакам.

1. *Географический*. Примеры:

su — страны СНГ;
ru — Россия;
ua — Украина;
us — США;
ca — Калифорния;
de — ФРГ;
jp — Япония.

2. *Профессиональный*. Примеры:

com — коммерческие организации;
edu — образовательные организации;
net — сетевые организации;
org — некоммерческие организации;
biz — бизнес-проекты;
info — информационный ресурс.

Рунет (Runet) — русская часть Интернета.

Домены второго и следующего уровней обычно принадлежат конкретным организациям, но могут обозначать снова географический объект или профессиональное подразделение.

Компьютер не может иметь своим адресом только имя домена первого уровня. Обычно доменное имя состоит из имен двух, трех или четырех доменов. Имена доменов в имени следуют в порядке возрастания уровня. Например:

ed.gov.ru — адрес компьютера Минобразования России;
radiorus.ru — адрес компьютера радио России;
microsoft.com — адрес компьютера фирмы Микрософт.

IP-адреса должны соответствовать доменным адресам.

DNS-сервер (читается «дэ-эн-эс», domain name server) — компьютер Интернета, на котором находятся таблицы соответствия IP- и доменных имен.

DNS-сервер называется также *сервером имен доменов*.

7°. **Веб-страница**. **Гиперссылка**. **Веб-пространство**

Сначала определим, что мы будем понимать под электронной страницей.

Электронная страница — электронная публикация, размещающаяся в одном текстовом окне.

Веб-страница — электронная страница, размещенная в Интернете и доступная в онлайн режиме.

Внутренние ссылки веб-страниц указывают на эту же страницу. Некоторые структуры внутренних ссылок показаны на рисунке 5. Внутренние ссылки могут составлять меню (оглавление) веб-страницы и разбивать страницу на части.

Рис. 5. Схемы внутренних ссылок:
а — без возврата на начало страницы;
б — возврат только с конца страницы;
в — возврат из любой части страницы

Внешние ссылки на веб-странице указывают на другие страницы. При этом возможны два случая (см. также рис. 6):

1) внешняя ссылка указывает сразу на всю другую страницу. Тогда при переходе на новую страницу на экране появляется ее начало;

2) внешняя ссылка указывает на конкретное место на другой странице. Тогда при переходе на новую страницу на экране она появляется, начиная с указанного места.

Гиперссылка, или гипертекстовая ссылка — ссылка в электронном документе.

Электронные страницы обычно связаны гиперссылками.

Рис. 6. Схемы внешних ссылок:
а — на всю новую страницу;
б — на конкретное место новой страницы

Если электронные страницы размещены на компьютере пользователя, то они могут быть не связаны гиперссылками со страницами других документов. Примерами таких страниц являются описания компьютерных программ, а также копии веб-страниц.

Если электронные страницы размещены в Интернете, но недоступны в онлайне (например, запакованы), то ни о каких гиперссылках, конечно, говорить не приходится.

Все веб-страницы, размещенные в Интернете, связаны гиперссылками друг с другом. Точнее, на каждую веб-страницу можно попасть с какой-нибудь другой веб-страницы.

> Даже если пользователь разместит в Интернете веб-страницу безо всяких ссылок, то по прошествии некоторого времени на эту страницу автоматически появятся ссылки. Автоматические поисковые системы просматривают все веб-страницы и размещают ссылки на эти веб-страницы на поисковых машинах.

Все веб-страницы, расположенные в Интернете, носят вместе одно специальное название.

WWW (World Wide Web), или Web, или веб-пространство, или Всемирная паутина, или ВВВ — множество всех веб-страниц.

Всемирная паутина — дословный перевод World Wide Web. Ее также называют просто *Паутиной*, или *Тремя «В»*.

8°. Сайт. Портал

Основной структурной и организационной единицей веб-пространства является *сайт*. Обычно страницы сайта имеют одинаковый дизайн оформления, а сам сайт принадлежит одному человеку или организации.

Сайт — несколько взаимосвязанных гиперссылками веб-страниц одной тематики, размещенных на одном узле Интернета и имеющих общий адрес.

Сайт — основная структурная единица Паутины, состоящей из сайтов: любая веб-страница принадлежит какому-нибудь сайту.

Сайт — основная организационная структура, так как новые веб-страницы появляются либо при добавлении к уже существующим сайтам, либо путем создания новых сайтов.

Веб-страницы одного сайта обычно располагаются на одном узле Интернета. Таким образом, в адрес сайта как составная его часть входит доменный адрес узла Интернета.

Например, сайт, на котором выложены предыдущие издания этого учебника, находится на узле с адресом newmail.ru, являющимся доменом второго уровня. Веб-страницы этого сайта расположены на поддомене newmail.ru, и адресом сайта может быть — домен третьего уровня matsievsky.newmail.ru.

Веб-сервер — узел Интернета, на котором расположены сайты.

Адрес сайта — общий доменный адрес веб-страниц сайта.

Адрес веб-страницы — адрес сайта, к которому добавлен относительный адрес (т. е. путь) копии веб-страницы на этом сайте.

Основная страница сайта — веб-страница сайта, которая появляется при обращении к сайту по его адресу.

С основной страницы сайта можно попасть непосредственно или через другие его страницы на любую страницу сайта.

При открытии сайта пользователь сразу попадает на основную страницу. Посмотреть другую страницу сайта можно двумя основными способами:

1) перейти к ней по гиперссылкам, начиная с основной страницы;

2) сразу указать вместе с адресом сайта адрес страницы.

Например, на страницу на сайте matsievsky.newmail.ru, на которой выложено предыдущее издание этой книги, можно сразу попасть по адресу matsievsky.newmail.ru/inf/inf.dhtml, т. е. страница находится в файле inf.dhtml, который находится в директории inf, которая находится в корневой директории сайта matsievsky.newmail.ru.

Одиночные сайты могут объединяться в более крупные структуры. Средством такого объединения снова является сайт.

Портал — сайт, который объединяет несколько простых сайтов.

Мультипортал — сайт, который объединяет несколько порталов.

Пример портала — Чертовы кулички (kulichki.ru).

Назовем единственный русскоязычный мультипортал: Кирилл и Мефодий (km.ru).

Небольшие порции, страницы данных просматриваются в онлайновом режиме в ресурсе WWW, где можно получить на свой компьютер большой объем данных. Чтобы отличить ресурс WWW от других ресурсов, ссылки на сайт в официальных источниках записывают, добавляя перед адресом строку «http://www.».

Например, ссылка на сайт автора имеет официальный вид http://www.matsievsky.newmail.ru.

Ресурс WWW обычно создается с помощью *FTP*.

FTP (file transfer protocol) — онлайновый доступ к данным в Интернете, аналогичный доступу к данным (файлам и директориям) на локальном компьютере.

FTP-сервер — сервер, хранящий в Интернете данные для свободного или зарегистрированного доступа по FTP.

FTP-сервер называется *файловым сервером, или файл-сервером*.

Чтобы отличить ресурс FTP от других ресурсов, ссылки на него в официальных источниках записывают с указанием этого ресурса, добавляя перед адресом строку «ftp://».

Для работы в FTP можно использовать те же самые программы, которые служат для работы с файлами и директориями на персональном компьютере.

9°. Упражнения

1. Имена файлов без расширения состоят только из цифр 0, 1.

Запишите все 6 имен, имеющих не более 2 символов.

2. Имена файлов состоят только из цифр 0, 1.

Запишите все 18 имен файлов, основное имя которых имеет не более 2 символов, а расширение — не более 1.

3. Пусть на диске A: находится две директории: корневая A:\ и ее поддиректория D1, а также один файл 1.doc.

Возможны только 2 варианта их взаимного расположения. Нарисуйте эти варианты.

4. Пусть на диске A: находится директории A:\, D1 и файлы 1.doc, 2.doc.

Возможны 4 варианта их расположения. Нарисуйте эти варианты.

5. Будем изображать директории одним значком, а файлы — другим.

Определите, какими значками в каждой из следующих четырех схем обозначены объекты и подсчитайте количество директорий и файлов на следующих деревьях, изображенных на рисунке 7.

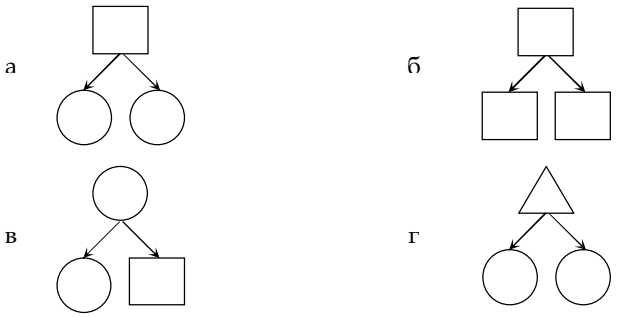


Рис. 7. Примеры деревьев директорий с файлами

6. Будем изображать директории одним значком, а файлы — другим.

Какие из следующих четырех схем могут быть деревьями директорий с файлами и какими значками в этом случае обозначены объекты, а какие — не могут и почему?

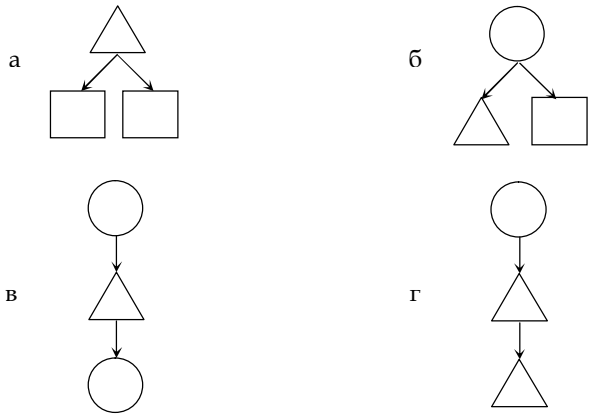


Рис. 8. Схемы, которые могут являться деревьями директорий с файлами, а могут никаким образом не быть ими

7. Глобальная сеть состоит из 3 узлов A, B и C, каждый из которых соединен с каждым другим 1 линией связи.

Имеются два принципиально разных пути передачи данных с узла A на узел B. Нарисуйте их.

8. Глобальная сеть состоит из 2 узлов A и B, которые соединены 2 линиями связи.

Имеются два принципиально разных пути передачи данных с узла A на узел B. Нарисуйте их.

9. Перечислите домены в порядке возрастания номера уровня в адресе сайта автора matsievsky.newmail.ru.

Решения

1. Не более двух — значит, 1 или 2.

Выпишем 2 односимвольных файла без расширения: 0; 1.

Выпишем 4 двусимвольных файла такэе без расширения: 00; 01; 10; 11.

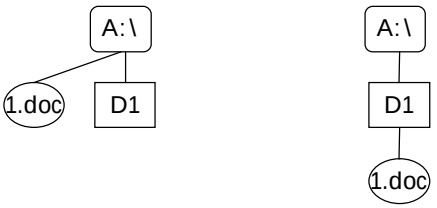
2. Выпишем 2 файла с основным именем из 1 символа без расширения: 0; 1.

Выпишем 4 файла с основным именем из 1 символа с расширением из 1 символа: 0.0; 0.1; 1.0; 1.1.

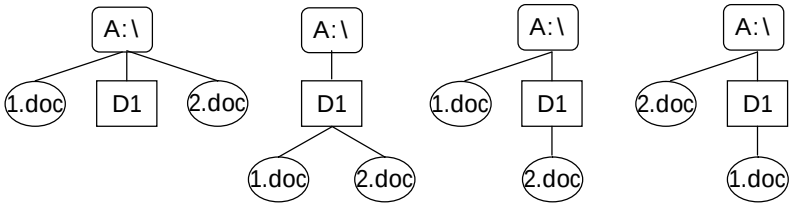
Выпишем 4 файла с основным именем из 2 символов без расширения: 00; 01; 10; 11.

Выпишем 8 файлов с основным именем из 2 символов с расширением из 1 символа: 00.0; 00.1; 01.0; 01.1; 10.0; 10.1; 1.10; 1.11.

3.



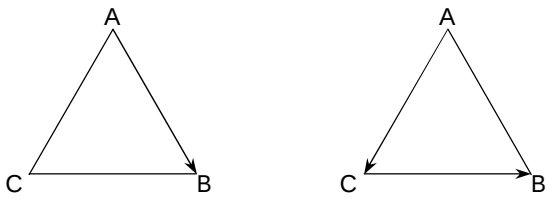
4.



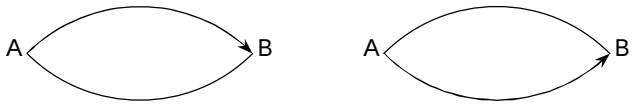
5. а) 1 прямоугольник — директория, 2 круга — файлы.
б) 3 прямоугольника — директории.
в) 2 круга — директории, 1 прямоугольник — файл.
г) 1 треугольник — директория, 2 круга — файлы.

6. а) 1 треугольник — директория, 2 прямоугольника — файлы.
б) Такой структуры быть не может.
в) Такой структуры быть не может.
г) Такой структуры быть не может.

7.



8.



9.
ru
newmail
matsievsky

2. Алгоритмы

Алгоритм 1. Выбор правильного ответа из нескольких предложенных.

1. Первый способ. Перебираем ответы по очереди. Неправильные ответы будут противоречить условию задания, правильный — не будет.

2. Второй способ. Выбираем какой-то один признак из условия задания и по нему исключаем из дальнейшего рассмотрения неправильные ответы. Затем берем другой признак и т. д., пока не останется один правильный ответ.

Алгоритм 2. Сравнение количества элементов множеств.

Если количество элементов в одном множестве больше количества элементов в другом, то первое множество не может быть подмножеством второго.

Алгоритм 3. Определение полного пути к файлу.

1. Полный путь к файлу всегда начинается с корневой директории.

2. Директория может принадлежать только одной наддиректории. (Но директория может иметь несколько поддиректорий.)

Алгоритм 4. Составление глобального адреса.

Глобальный адрес, и IP-, и доменный, имеет строго определенную структуру. Поэтому при составлении адреса из фрагментов его структура должна сохраниться.

Алгоритм 5. Расшифровка сообщения.

1. Сообщение расшифровывается последовательно, накапливая расшифрованные части, начиная с начала.

2. Иногда зашифрованное сообщение дается в явном виде, иногда — скрыто в тексте задания.

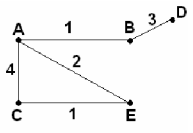
3. Задачи

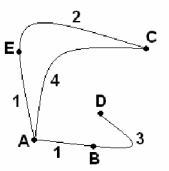
1°. **Выбор по признакам**

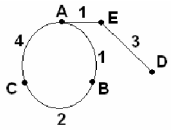
Задача 2006.A12

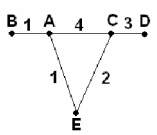
В таблице приведена стоимость перевозок между соседними железнодорожными станциями. Укажите схему, соответствующую таблице.

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

1) 

2) 

3) 

4) 

Задача 2006.A14

Для составления цепочек используются бусины, помеченные буквами: M, N, O, P, S. В середине цепочки стоит одна из бусин M, O, S. На третьем — любая гласная, если первая буква согласная, и любая согласная, если первая гласная. На первом месте — одна из бусин O, P, S, не стоящая в цепочке в середине.

Какая из перечисленных цепочек создана по этому правилу?

1) SMP 2) MSO 3) SNO 4) OSN

Задача 2007.А14

Для составления цепочек разрешается использовать бусины 5 типов, обозначаемых буквами А, Б, В, Е, И. Каждая цепочка должна состоять из трех бусин, при этом должны соблюдаться следующие правила:

1) на первом месте стоит одна из букв: А, Е, И;
2) после гласной буквы в цепочке не может снова идти гласная, а после согласной — согласная;
3) последней буквой не может быть А.

Какая из цепочек построена по этим правилам?

1) АИБ 2) ЕВА 3) БИВ 4) ИБИ

Задача 2008.А14

В формировании цепочки из четырех бусин используются некоторые правила.

В конце цепочки стоит одна из бусин P, N, T, O. На первом — одна из бусин P, R, T, O, которой нет на третьем месте. На третьем месте — одна из бусин O, P, T, не стоящая в цепочке последней. Какая из перечисленных цепочек могла быть создана с учетом этих правил?

1) PORT 2) TTTO 3) TTOO 4) OOPO

Задача 2009.А12

Цепочка из трех бусин, помеченных латинскими буквами, формируется по следующему правилу. В конце цепочки стоит одна из бусин A, B, C. На первом месте — одна из бусин B, D, C, которой нет на третьем месте. В середине — одна из бусин A, C, E, B, не стоящая на первом месте.

Какая из перечисленных цепочек создана по этому правилу?

1) CBB 2) EAC 3) BCD 4) BCB

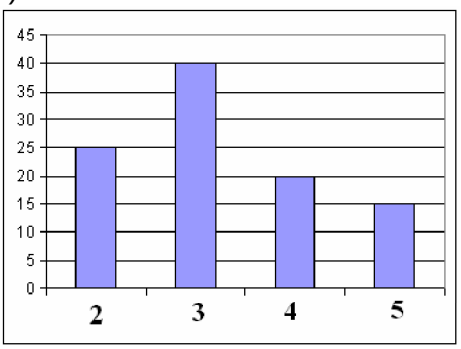
2°. **Сравнение количества элементов множеств**

Задача 2007.А19

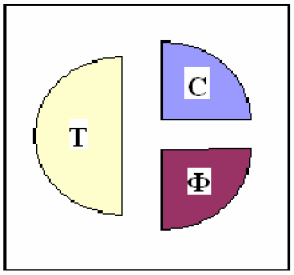
В цехе трудятся рабочие трех специальностей — токари (Т), слесари (С) и фрезеровщики (Ф). Каждый рабочий имеет разряд не меньший второго и не больший пятого. На диаграмме I отражено количество рабочих с различными разрядами, а на диаграмме II — распределение рабочих по специальностям.

Каждый рабочий имеет только одну специальность и один разряд.

I)



II)



Имеются четыре утверждения.

А) Все рабочие третьего разряда могут быть токарями.

Б) Все рабочие третьего разряда могут быть фрезеровщиками.

В) Все слесари могут быть пятого разряда.

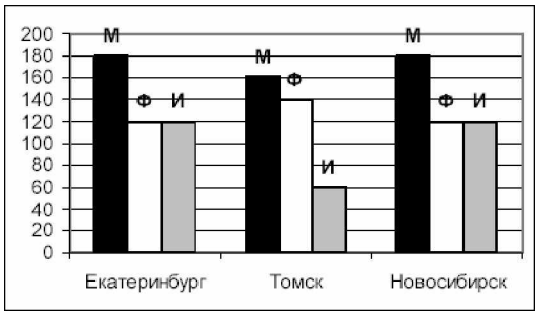
Г) Все токари могут быть четвертого разряда.

Какое из этих утверждений следует из анализа обеих диаграмм?

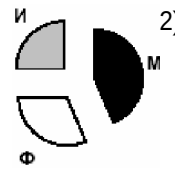
1) А 2) Б 3) В 4) Г

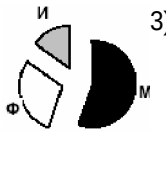
Задача 2009.А17

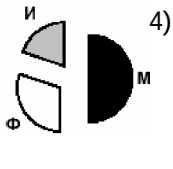
На диаграмме показано количество призеров олимпиады по информатике (И), математике (М), физике (Ф) в трех городах России.

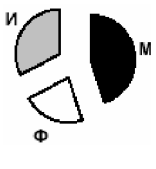


Какая из диаграмм правильно отражает соотношение общего числа призеров по каждому предмету для всех городов вместе?

1) 

2) 

3) 

4) 

3°. **Маски имен файлов**

Задача 2008.А15

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

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

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

Определите, какое из указанных имен файлов удовлетворяет маске:

?a???*

1) dad1 2) dad22 3) 3daddy 4) add444

Задача 2009.А13

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

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

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

Определите, какое из указанных имен файлов удовлетворяет маске:

?hel*lo.c?*

1) hello.c 2) hello.cpp 3) hhelolo.cpp 4) hhelolo.c

4°. Структура дерева директорий

Задача 2006.А15

В некотором каталоге хранился файл **Дневник**.txt. После того, как в этом каталоге создали подкаталог и переместили в созданный подкаталог файл **Дневник**.txt, полное имя файла стало

A:\SCHOOL\USER\TXT\MAY**Дневник**.txt

Каково полное имя каталога, в котором хранился файл до перемещения?

1) MAY
2) A:\SCHOOL\USER\TXT
3) TXT
4) A:\SCHOOL\USER\TXT\MAY

Задача 2007.А15

Перемещаясь из одного каталога в другой, пользователь последовательно посетил каталоги DOC, USER, SCHOOL, A:\, LETTER, INBOX. При каждом перемещении пользователь либо спускался в каталог на уровень ниже, либо поднимался на уровень выше. Каково полное имя каталога, из которого начал перемещение пользователь?

1) A:\DOC
2) A:\LETTER\INBOX
3) A:\SCHOOL\USER\DOC
4) A:\DOC\USER\SCHOOL

5°. **Структура глобальных адресов**

Задача 2006.В4

Доступ к файлу www.txt, находящемуся на сервере ftp.net, осуществляется по протоколу http. В таблице фрагменты адреса файла закодированы буквами от А до Ж. Запишите последовательность этих букв, кодирующую адрес указанного файла.

А	.txt	Б	http	В	/	Г	://	Д	.net	Е	www	Ж	ftp

Задача 2008.В7

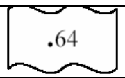
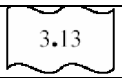
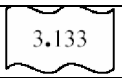
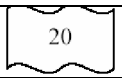
Доступ к файлу htm.net, находящемуся на сервере com.edu, осуществляется по протоколу ftp. В таблице фрагменты адреса файла закодированы буквами от А до Ж. Запишите последовательность этих букв, кодирующую адрес указанного файла.

А	/	Б	com	В	.edu	Г	://	Д	.net	Е	htm	Ж	ftp

Задача 2009.B9

Петя записал IP-адрес школьного сервера на листке бумаги и положил его в карман куртки. Петина мама случайно постирала куртку вместе с запиской. После стирки Петя обнаружил в кармане четыре обрывка с фрагментами IP-адреса. Эти фрагменты обозначены буквами А, Б, В и Г. Восстановите IP-адрес.

В ответе укажите последовательность букв, обозначающих фрагменты, в порядке, соответствующем IP-адресу.

			
А	**Б**	**В**	**Г**

6°. **Расшифровка сообщения**

Задача 2006.A13

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

В	К	А	Р	Д
000	11	01	001	10

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

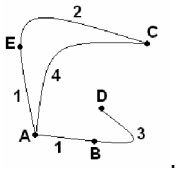
1) 110100000100110011
2) 111010000010010011
3) 110100001001100111
4) 110110000100110010

Задача 2009.B6

Классный руководитель пожаловался директору, что у него в классе появилась компания из 3 учеников, один из которых всегда говорит правду, другой всегда лжет, а третий говорит через раз то ложь, то правду. Директор знает, что их зовут Коля, Саша и Миша, но не знает, кто из них правдив, а кто — нет. Однажды все трое прогуляли урок астрономии. Директор знает, что никогда раньше никто из них не прогуливал астрономию. Он вызвал всех троих в кабинет и поговорил с мальчиками. Коля сказал: «Я всегда прогуливаю астрономию. Не верьте тому, что скажет Саша». Саша сказал: «Это был мой первый прогул этого предмета». Миша сказал: «Все, что говорит Коля,— правда». Директор понял, кто из них кто. Расположите первые буквы имен мальчиков в порядке: «говорит всегда правду», «всегда лжет», «говорит правду через раз». (Пример: если бы имена мальчиков были Рома, Толя и Вася, ответ мог бы быть: РТВ.)

4. Ответы

1°. Выбор по признакам

Задача 2006.A12. 2) 

.

Задача 2006.A14. 4) OSN.

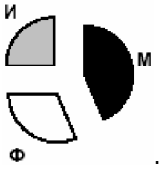
Задача 2007.A14. 4) ИБИ.

Задача 2008.A14. 4) ООРО.

Задача 2009.A12. 1) СВВ.

2°. Сравнение количества элементов множеств

Задача 2007.A19. 1) A.

Задача 2009.A17. 1) 

.

3°. Маски имен файлов

Задача 2008.A15. 2) dad22.

Задача 2009.A13. 3) hhelolo.cpp.

4°. Структура дерева директорий

Задача 2006.A15. 2) A:\SCHOOL\USER\TXT.

Задача 2007.A15. 3) A:\SCHOOL\USER\DOC.

5°. **Структура глобальных адресов**

Задача 2006.B4. БГЖДВЕА.
Задача 2008.B7. ЖГБВАЕД.
Задача 2009.B9. ГБВА.

6°. **Расшифровка сообщения**

Задача 2006.A13. 3) 110100001001100111.
Задача 2009.B6. СКМ.

5. Решения

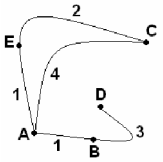
1°. Выбор по признакам

Рекомендации. 1. Условие заданий трудностей не вызывает.

2. Решение проводится по одному из вариантов алгоритма 1. Это сделать достаточно легко, если методично и последовательно перебирать предложенные варианты.

3. Перебор ответов трудностей не вызывают.

Задача 2006.A12



Ответ: 2) .

Решение 1. Проверим для каждой схемы из вариантов ответа, соответствуют ли ей соседство станций и стоимость перевозок, указанные в таблице.

Будем сравнивать всю таблицу с каждой схемой по очереди. Заметим, что сравнению подлежат только соседние станции, причем только те, стоимость перевозок между которыми находится выше главной диагонали таблицы.

1) Первая схема. На таблице станция A соседствует с 3 станциями B, C и E, что соответствует схеме. Но стоимость перевозок между ними не соответствует схеме: между A и E у таблицы и у схемы разные стоимости перевозки.

2) Вторая схема. На таблице станция A соседствует с 3 станциями B, C и E, что соответствует схеме. Стоимость перевозок между ними также соответствует схеме.

На таблице станция B соседствует с 1 еще на рассмотренной станцией D, что соответствует схеме. Стоимость перевозок между ними также соответствует схеме.

Наконец, на таблице станция C соседствует с 1 еще на рассмотренной станцией E, что соответствует схеме. Стоимость перевозок между ними также соответствует схеме.

Итак, подходит схема 2).

Для проверки рассмотрим оставшиеся схемы.

3) Третья схема. На таблице станция A соседствует с 3 станциями B, C и E, что соответствует схеме. Стоимость перевозок между ними также соответствует схеме.

На таблице станция B соседствует с 1 еще на рассмотренной станцией D, что не соответствует схеме.

4) Четвертая схема. На таблице станция A соседствует с 3 станциями B, C и E, что соответствует схеме. Стоимость перевозок между ними также соответствует схеме.

На таблице станция B соседствует с 1 еще на рассмотренной станцией D, что не соответствует схеме.

Решение 2. Проверим для каждой схемы из вариантов ответа, соответствуют ли ей соседство станций и стоимость перевозок, указанные в таблице.

Сначала сравним первую строку таблицы с каждой схемой, затем — вторую и, наконец, третью. Заметим, что сравнению подлежат только соседние станции, причем только те, стоимость перевозок между которыми находится выше главной диагонали таблицы.

1) Первая строка таблицы. Станция A соседствует с 3 станциями B, C и E, что соответствует схемам 1), 2), 3) и 4).

Стоимость перевозок между ними соответствует схемам 2), 3) и 4) и не соответствует схеме 1).

Схема 1) не подходит.

2) Вторая строка таблицы. Станция B соседствует с 1 еще на рассмотренной станцией D, что соответствует только схеме 2), а схемам 3) и 4) не соответствует.

Схемы 3) и 4) не подходят.

3) Третья строка. Проверим, что она соответствует оставшейся схеме 2). Станция C соседствует с 1 еще на рассмотренной станцией E, что соответствует схеме. Стоимость перевозок между ними также соответствует схеме.

Итак, подходит схема 2).

Задача 2006.A14

Ответ: 4) OSN.

Решение 1. Переберем варианты ответов и посмотрим, какой из них удовлетворяет поставленным условиям.

1. SMP. Не удовлетворяет условию 1).
2. ЕВА. Удовлетворяет условию 1).
Удовлетворяет условию 2).
Не удовлетворяет условию 3).
3. БИВ. Не удовлетворяет условию 1).
4. ИБИ. Удовлетворяет условию 1).
Удовлетворяет условию 2).
Удовлетворяет условию 3).

Решение 2. Поскольку в середине цепочки стоит одна из бусин M, O, S, то подходят только три цепочки 1), 2) и 4).

Из них подходят цепочки 2) и 4), т. к. на третьем месте стоит любая гласная, если первая буква согласная, и любая согласная, если первая гласная.

Наконец, на первом месте стоит одна из бусин O, P, S, и остается одна цепочка 4).

Задача 2007.A14

Ответ: 4) ИБИ.

Решение. Совершенно аналогично решению предыдущей задачи.

Задача 2008.A14

Ответ: 4) OOPO.

Решение. Совершенно аналогично решению предыдущей задачи.

Задача 2009.A12

Ответ: 1) CBB.

Решение. Совершенно аналогично решению предыдущей задачи.

2°. **Сравнение количества элементов множеств**

Замечание. Логические задачи по теории из § 1.

Рекомендации. 1. При работе с текстом задания нужно правильно вычислить количество элементов во всех множествах.

2. Решение проводится по одному из вариантов алгоритма 1 на основе алгоритма 2.

3. Перебор ответов трудностей не вызывают.

Задача 2007.А19

Ответ: 1) А.

Решение. Вычислим количество элементов во всех указанных множеств рабочих.

Из диаграммы I получаем, что общее количество всех рабочих $25 + 40 + 20 + 15 = 100$.

Из диаграммы II видно, что среди всех рабочих токари составляют половину, т. е. их 50, а слесари и фрезеровщики — по четверти, т. е. их по 25.

Теперь можно рассмотреть предложенные утверждения и просто сравнить количества тех или иных множеств рабочих, поскольку каждый рабочий имеет только одну специальность и один разряд.

А) Все рабочие 3-го разряда могут быть токарями, так как токарей (50) не меньше, чем рабочих 3-го разряда (40). Это правильный ответ.

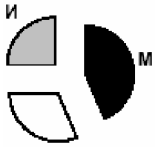
Для контроля рассмотрим оставшиеся утверждения.

Б) Все рабочие третьего разряда могут быть фрезеровщиками — неверное утверждение, так как рабочих третьего разряда (40) больше, чем фрезеровщиков (25).

В) Все слесари могут быть пятого разряда — неверное утверждение, так как слесарей (25) больше, чем рабочих пятого разряда (15).

Г) Все токари могут быть четвертого разряда — неверное утверждение, так как токарей (50) больше, чем рабочих четвертого разряда (20).

Задача 2009.А17



Ответ: 1) Ф .

Решение 1. Сравним количество элементов для всех указанных множеств с половиной и четвертью общего числа призеров.

Из диаграммы в условии задания видно, что число призеров по математике равно 180 + 160 + 180 = 520, по физике — 120 + 140 + 120 = 340 и по информатике — 120 + 60 + 120 = 300.

Общее число призеров по всем предметам получается 520 + 340 + 300 = 1160.

Сравним число призеров по предметам с половиной и четвертью общего числа призеров. Половина общего числа призеров равна 1160 / 2 = 580, а четверть — 580 / 2 = 290.

Таким образом, общее число призеров по математике (520) меньше половины и больше четверти общего числа призеров, по физике (340) — немногим больше четверти, и по информатике (300) — чуть-чуть больше четверти, меньше, чем по физике.

Теперь рассмотрим четыре предложенные в ответах диаграммы по порядку.

На диаграмме 1) число призеров по математике меньше половины, по физике — больше четверти и больше, чем по информатике, а по информатике — примерно четверть. Поэтому диаграмма 1) правильно отражает соотношение общего числа призеров по каждому предмету для всех городов вместе.

Для контроля рассмотрим оставшиеся диаграммы.

На диаграмме 2) число призеров по математике больше половины, поэтому эта диаграмма не подходит.

На диаграмме 3) число призеров по математике примерно равно половине, поэтому она не подходит.

На диаграмме 4) число призеров по математике меньше половины, но число призеров по физике меньше, чем по информатике. Поэтому она не подходит.

Решение 2. Вычислим соотношение количества элементов для всех указанных множеств призеров и будем искать правильную диаграмму в ответе методом исключения.

Из диаграммы в условии задания видно, что число призеров по математике равно 180 + 160 + 180 = 520, по физике — 120 + 140 + 120 = 340 и по информатике — 120 + 60 + 120 = 300.

Общее число призеров по всем предметам получается 520 + 340 + 300 = 1160.

Теперь рассмотрим четыре предложенные в ответах диаграммы. Точное соотношение призеров по ним установить невозможно. Однако соотношения можно сравнить друг с другом.

Проще всего оценить количество математиков, количество которых на четырех диаграммах сравнимо с половиной всех призеров. Количество призеров по математике на этих диаграммах различное: математиков на диаграммах 1) и 4) меньше половины всех призеров, на диаграмме 3) — ровно половина и на диаграмме 2) — больше половины.

Поскольку на самом деле математиков меньше половины: 520 < 1160 / 2 = 580, то правильно отражает соотношение общего числа призеров по каждому предмету для всех городов вместе либо диаграмма 1), либо диаграмма 4).

Рассмотрим диаграммы 1) и 4). На диаграмме 1) число призеров по физике больше числа призеров по информатике, а на диаграмме 4) — наоборот.

В действительности число призеров по физике больше, чем по информатике: 340 > 300. Поэтому верна диаграмма 1).

3°. **Маски имен файлов**

Рекомендации. 1. Не забыть, что * может быть пустым местом.

2. Решение проводится по одному из вариантов алгоритма 1.

3. Перебор ответов трудностей не вызывают.

Задача 2008.A15

Ответ: 2) dad22.

Решение 1. Переберем все варианты ответов.

1. dad1. Первый символ имени файла d соответствует первому знаку ? маски ?a???*. Второй символ имени a соответствует второму знаку маски — букве a. Третий и четвертый символы имени файла d1 соответствует третьему и четвертому знакам маски ??. А вот пятому знаку маски ? никакой символ в имени не соответствует. Поэтому это имя не соответствует маске.

2. dad22. Первый символ имени файла d соответствует первому знаку ? маски ?a???*. Второй символ имени a соответствует второму знаку маски — букве a. Третий, четвертый и пятый символы имени файла d22 соответствует третьему, четвертому и пятому знакам маски ???. На этом символы имени заканчиваются. Последний знак маски * может не соответствовать никакому символу имени. Поэтому это имя соответствует маске.

3. Последние два варианта ответов можно не рассматривать. Однако заметим, что оба варианта не подходят по одной простой причине: на втором месте имени нет буквы a, которая есть в маске.

Решение 2. Присмотримся к символам «?» в маске. Согласно этим символам, перед буквой «a» в имени файла должна стоять ровно одна буква, а после буквы «a» — еще три.

Четвертое имя «add444» перед буквой «a» вообще не имеет символов, а третье «3daddy» имеет перед буквой «a» два символа, поэтому они не подходят.

Первое имя «dad1» имеет после буквы «a» только два символа, поэтому оно не подходит.

Всем этим условиям удовлетворяет только одно второе имя «dad22».

Задача 2009.A13

Ответ: 3) hhelolo.cpp.

Решение. Совершенно аналогично решению предыдущей задачи.

4°. **Структура дерева директорий**

Замечание. Логические задачи на дерево директорий.

Рекомендации. 1. Внимательно изучайте взаимное расположение директорий: кто является чьей поддиректорией.

2. При решении помните, что директория не может быть поддиректорией двух директорий.

3. Ответ задания легко вычисляется без использования вариантов ответов.

Задача 2006.A15

Ответ: 2) A:\SCHOOL\USER\TXT.

Решение. Полное имя стало на 1 каталог больше, поэтому, полное имя старого каталога на 1 каталог меньше нового.

Задача 2007.A15

Ответ: 3) A:\SCHOOL\USER\DOC.

Решение. Очевидно, пользователь начал перемещение с каталога DOC. Найдем полный путь к каталогу DOC.

В списке каталогов есть корневой. Начнем с конца, т. е. с этого корневого каталога.

Каталог A:\ из списка каталогов, которые посетил пользователь,— корневой, следовательно, в каталог A:\ пользователь мог попасть только из его подкаталога. Итак, каталог SCHOOL является подкаталогом A:\.

В каталог SCHOOL пользователь попал из каталога USER. Но надкаталог SCHOOL уже известен — это корневой каталог A:\. Следовательно,— USER подкаталог SCHOOL.

Аналогично DOC — подкаталог USER.

Получаем полное имя A:\SCHOOL\USER\DOC.

5°. **Структура глобальных адресов**

Замечание. Логические задачи на глобальные адреса.

Рекомендации. 1. Внимательно изучите данные условия.

2. Используйте структуру глобального адреса.

3. Ответ задания легко вычисляется без использования вариантов ответов.

Задача 2006.B4

Ответ: БГЖДВЕА.

Решение. Первым идет имя протокола из условия: http://.
Затем — имя сервера, также указанного в условии: ftp.net.
Наконец, после слеша — имя файла из условия: www.txt.

Задача 2008.B7

Ответ: ЖГБВАЕД.

Решение. Совершенно аналогично предыдущей задаче.

Задача 2009.B9

Ответ: ГБВА.

Решение. Посмотрим, какие фрагменты могут быть перед и после данных фрагментов.

1) Перед фрагментом .64 может быть любой из остальных фрагментов, причем какой-то фрагмент должен быть обязательно, поскольку фрагмент .64 начинается с точки.

После фрагмента .64 не может быть никакого другого фрагмента, поскольку максимальное значение чисел — 256, и, кроме того, остальные фрагменты не начинаются с точки.

Итак, фрагмент .64 — последний в IP-адресе.

2) Перед фрагментом 3.13 находится или 20, или ничего.

После фрагмента 3.13 находится фрагмент .64 или 3.133 (этот фрагмент не может быть последним, так как последний уже найден).

3) Перед фрагментом 3.133 находится 3.12 или 20 или ничего.

После фрагмента 3.133 может находиться только фрагмент .64. Итак, имеем окончание адреса: 3.133.64.

4) После фрагмента 3.13 может находиться только один из двух возможных: 3.133. Имеем следующее окончание адреса: 3.133.133.64.

5) Оставшийся фрагмент 20 может стоять только в начале адреса: 203.133.133.64.

6°. **Расшифровка сообщения**

Рекомендации. 1. Внимательное изучение условия задания позволяет выбрать легкий способ его решения.

2. Сообщение, скрытое в задании, расшифровывается последовательно, начиная с начала.

3. Иногда расшифровываются варианты ответов, иногда — сам текст задания.

Задача 2006.А13

Ответ: 3) 110100001001100111.

Решение. Попробуем расшифровать каждое из сообщений.

1) В начале первого сообщения 110100000100110011 может стоять только К. Вторая буква сообщения К0100000100110011 — А. Третья буква сообщения КА00000100110011 — буква В. Четвертая буква сообщения КАВ00100110011 — буква Р. Пятая у КАВР00110011 — буква Р. Шестая у КАВРР10011 — Д. Седьмая у КАВРРД011 — А. Получаем сообщение КАВРРДА1, у которого последняя цифра 1 не расшифровывается.

2) Второе сообщение тоже не расшифровывается: КДДВАРР1.

3) Третье сообщение расшифровывается: КАВАРДАК.

Задача 2009.В6

Ответ: СКМ.

Решение. Условие задачи позволяет не перебирать варианты, а сразу вычислить решение. Поскольку никогда раньше никто из мальчиков не прогуливал астрономию, то сразу ясно, что Коля в первой фразе солгал, а Саша сказал правду.

Так как Коля солгал, то Миша тоже солгал.

Так как Саша говорит правду, то Коля солгал и во второй фразе.

Получается, что два мальчика всегда лгут. Следовательно, нужно исправить условие.

Поскольку Саша и Миша сказали по одной фразе, то они не могут говорить правду через раз. Поэтому говорит правду через раз Коля. Можно исправить его вторую фразу, чтобы она стала правдивой, так: «Не верьте тому, что скажет Миша».

§ 3. Поиск всех вариантов

1. Теория

1°. Задача о волке, козе и капусте

Решим следующую общеизвестную задачу (математический фольклор), вошедшую когда-то в упрощенно виде в первый сборник задач по математике для университетов.

Кроме того, правильно оформим решение этой задачи, чтобы оно было понятно для всех и, самое главное, чтобы это решение было легко проверить.

Задача 1. Волк, коза и капуста.

Крестьянин повез на рынок продавать волка, козу и большой кочан капусту. Чтобы сократить путь, он решил переправиться через реку: у него была с собой надувная лодка.

Но вот беда: лодка была одноместная и вмещала только одного крестьянина, или крестьянина только с волком, или только с козой, или только с большим кочаном капусты.

А на одном берегу нельзя оставить одних, без крестьянина, волка с козой или козу с капустой по естественным причинам.

Как все-таки хитрый крестьянин переправился через реку?

Назовем *состоянием задачи* расположение крестьянина и его имущества на двух берегах реки. Тогда задача заключается в нахождении *допустимых действий* крестьянина, не приводящих к гибели его имущества, по переходу из *начального состояния* «крестьянин, лодка, волк, коза и капуста на первом берегу» к конечному состоянию «крестьянин, лодка, волк, коза и капуста на втором берегу».

Состояние	Первый берег	Второй берег
Начальное	кр, в, ко, ка	—
Конечное	—	кр, в, ко, ка

Решение задачи можно получить «автоматически», не особенно задумываясь, если при этом следовать трем принципам:

1) крестьянин совершает только допустимые перевозки;

2) перевозки всегда изменяют состояния задачи;

3) перевозки не приводят в повторению состояний.

Проблема здесь заключается не в нахождении решения, а в его оформлении. Решение нужно записать достаточно коротко, но так, чтобы его можно было легко проверить. Приведем получение решения задачи и его правильное оформление.

1. Вначале есть только первое начальное состояние.

Состояние	1-й берег	2-й берег
1	кр, в, ко, ка	—

2. Что может сделать крестьянин в начальном состоянии? В принципе, каждый раз крестьянин может сделать только четыре действия:

1) перевезти волка;

2) перевезти козу;

3) перевезти капусту;

4) ничего не перевезти.

Посмотрим, какие действия крестьянин может выполнить без ущерба для своего имущества:

1) при перевозе волка погибнет капуста;

2) при перевозе козы ничего не погибнет;

3) при перевозе капусты погибнет коза;

4) при перевозе ничего погибнет капуста и коза.

Итак, в первом состоянии нужно перевезти козу.

Состояние	Перевоз	1-й берег	2-й берег
1	Козы	кр, в, ко, ка	—
2		в, ка	кр, ко

После первого действия, перевоза козы, крестьянин вместе с козой и лодкой переместился на второй берег. Впрочем, в этой задаче перемещение лодки можно не отслеживать, поскольку она всегда при крестьянине.

3. Посмотрим, какие действия крестьянин может выполнить во втором состоянии без ущерба для своего имущества и без возврата к прежнему состоянию:

1) при перевозе козы никто не погибнет, но получится прежнее состояние 1;

2) при перевозе ничего никто не погибнет и прежнего состояния не получится.

Итак, во втором состоянии крестьянину нужно просто переехать самому.

Состояние	Перевоз	1-й берег	2-й берег
1		кр, в, ко, ка	—
	Козы		
2		в, ка	кр, ко
	—		
3		кр, в, ка	ко

4. Посмотрим, какие действия крестьянин может выполнить в третьем состоянии:

1) при перевозе волка никто не погибнет и прежнего состояния не получится;

2) при перевозе капусты никто не погибнет и прежнего состояния не получится;

3) при перевозе ничего никто не погибнет, но получится прежнее состояние.

Итак, у нас получилось, что крестьянин может сделать два разных действия.

Поэтому с этого мест будем отслеживать две линии решения задачи.

Состояние	Перевоз	1-й берег	2-й берег	Перевоз	1-й берег	2-й берег
1		кр, в, ко, ка	—		кр, в, ко, ка	—
	Козы			Козы		
2		в, ка	кр, ко		в, ка	кр, ко
	—			—		
3		кр, в, ка	ко		кр, в, ка	ко
	Волка			Капусты		
4		ка	кр, в, ко		в	кр, ко, ка

5. Найдем четвертое действие крестьянина. 1-я линия:
1) перевоз волка возможен, но будет прежнее состояние;
2) перевоз козы возможен и прежнего состояния не будет;
3) перевоз ничего невозможен.
2-я линия:
1) перевоз козы возможен и прежнего состояния не будет;
ц) перевоз капусты возможен, но будет прежнее состояние;
3) перевоз ничего невозможен.
Итак, в обоих случаях крестьянин перевозит козу.

Состояние	Перевоз	1-й берег	2-й берег	Перевоз	1-й берег	2-й берег
1		кр, в, ко, ка	—		кр, в, ко, ка	—
	Козы			Козы		
2		в, ка	кр, ко		в, ка	кр, ко
	—			—		
3		кр, в, ка	ко		кр, в, ка	ко
	Волка			Капусты		
4		ка	кр, в, ко		в	кр, ко, ка
	Козы			Козы		
5		кр, ко, ка	в		кр, в, ко	ка

6. Найдем пятое действие крестьянина. 1-я линия:
1) перевоз козы возможен, но будет прежнее состояние;
2) перевоз капусты возможен и прежнего состояния не будет;
3) перевоз ничего невозможен.
2-я линия:
1) перевоз волка возможен и прежнего состояния не будет;
ц) перевоз козы возможен, но будет прежнее состояние;
3) перевоз ничего невозможен.

Со-стоя-ние	Перевоз	1-й берег	2-й берег	Перевоз	1-й берег	2-й берег
1	Козы	кр, в, ко, ка	—			
2	—	в, ка	кр, ко			
3		кр, в, ка	ко			
4	Волка	ка	кр, в, ко	Капусты	в	кр, ко, ка
5	Козы	кр, ко, ка	в	Козы	кр, в, ко	ка
	Капусты			Волка		
6		ко	кр, в, ка			

В результате мы пришли к одному и тому же состоянию в обоих линиях решения задачи. Поэтому дальше поддерживать две линии не имеет смысла, и мы возвращаемся к одной линии решения задачи.

Теперь окончание решения трудностей никаких не представляет. Крестьянину осталось съездить на другой берег за козой. Сделать это он может за два действия, что мы и отразим на схеме.

Итак, нами получено и правильно оформлено наиболее полное решение задачи о волке, козе и капусте. Самое главное, что представленное ниже решение легко проверяется на правильность: ясно видно, кого перевозили и что получилось.

Состояние	Перевоз	1-й берег	2-й берег	Перевоз	1-й берег	2-й берег
1		кр, в, ко, ка	—			
	Козы					
2		в, ка	кр, ко			
	—					
3		кр, в, ка	ко			
	Волка			Капусты		
4		ка	кр, в, ко		в	кр, ко, ка
	Козы			Козы		
5		кр, ко, ка	в		кр, в, ко	ка
	Капусты			Волка		
6		ко	кр, в, ка			
	—					
7		кр, ко	в, ка			
	Козы					
8			кр, в, ко, ка			

2°. **Задача о двух отцах и двух сыновьях**

Два отца, каждый со своим сыном, пошли вместе в поход и подошли к реке. У них с собой была легкая надувная лодка, выдерживающая одного взрослого или двух детей. Перевезти рюкзаки на тот берег оказалось просто. А вот как они переехали сами?

Решим эту задачу по схеме, которую только что изучили.

1. Вначале есть только первое начальное состояние.

Состояние	1-й берег	2-й берег
1	о, о, с, с, л	—

При решении этой задачи придется отслеживать движение лодки, потому что в лодку могут сесть разные люди. Необходимо показывать, с какого берега на какой она переезжает, она должна быть попеременно то на одном, то на другом берегу.

2. Как вы уже поняли, взрослых друг от друга можно не различать, как и детей. В начальном состоянии переехать может один взрослый, один ребенок и два ребенка. Но при переезде одного человека ему придется на следующем действии вернуться обратно, и снова получится начальное состояние. Поэтому первое действие одно: переезд двух детей.

Состояние	Перевоз	1-й берег	2-й берег
1		о, о, с, с, л	—
	Два сына		
2		о, о	с, с, л

3. Если теперь вернутся обоим сыновьям, то получится прежнее состояние. Поэтому возвращается один из сыновей.

Состояние	Перевоз	1-й берег	2-й берег
1		о, о, с, с, л	—
	Два сына		
2		о, о	с, с, л
	Один сын		
3		о, о, с, л	с

4. Если теперь переедет сын, то получится прежнее состояние. Поэтому переезжает один из отцов.

Состояние	Перевоз	1-й берег	2-й берег
1		о, о, с, с, л	—
	Два сына		
2		о, о	с, с, л
	Один сын		
3		о, о, с, л	с
	Один отец		
4		о, с	о, с, л

5. Переехать может только сын, иначе будет прежнее состояние.

Состояние	Перевоз	1-й берег	2-й берег
1		о, о, с, с, л	—
	Два сына		
2		о, о	с, с, л
	Один сын		
3		о, о, с, л	с
	Один отец		
4		о, с	о, с, л
	Один сын		
5		о, с, с, л	о

Одного отца перевезли. Рассуждая дальше и следя, чтобы не было прежнего состояния, причем не обязательно предпоследнего, а любого предыдущего, получаем решение задачи.

Состояние	Перевоз	1-й берег	2-й берег
1		о, о, с, с, л	—
	Два сына		
2		о, о	с, с, л
	Один сын		
3		о, о, с, л	с
	Один отец		
4		о, с	о, с, л
	Один сын		
5		о, с, с, л	о
	Два сына		
6		о	о, с, с, л
	Один сын		
7		о, с, л	о, с
	Один отец		
8		с	о, о, с, л
	Один сын		
9		с, с, л	о, о
	Два сына		
10			о, о, с, с, л

2. Алгоритмы

Алгоритм 1. Поиск всех вариантов.

1. Если возможен выбор количества первоначальных вариантов, рассматриваем тот случай, когда первоначальных вариантов меньше всего.

2. Перебираем все возможные линии развития ситуации, начиная со всех первоначальных вариантов.

3. Если становится ясно, что какая-то линия развития не приведет к нужному решению, то ее развитие продолжать не нужно.

Алгоритм 2. Решение с конца.

Когда при поиске всех вариантов возникают ветвящиеся деревья вариантов, то задачу можно также решить, начиная с конца.

Сначала выясняем, какое действие может быть последним.

Затем разбираемся с предпоследним действием.

И т. д. до первого действия.

3. Задачи

1°. **Максимумы и минимумы**

Задача 2007.A12

Таблица стоимости перевозок устроена следующим образом: числа, стоящие на пересечениях строк и столбцов таблиц, означают стоимость проезда между соответствующими соседними станциями. Если пересечение строки и столбца пусто, то станции не являются соседними.

Укажите таблицу, для которой выполняется условие: «Минимальная стоимость проезда из A в B не больше 6».

Стоимость проезда по маршруту складывается из стоимостей проезда между соответствующими соседними станциями.

1)

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

2)

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

3)

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

4)

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

Задача 2008.A12

Грунтовая дорога проходит последовательно через населенные пункты A, B, C и D. При этом длина дороги между A и B равна 80 км, между B и C — 50 км, и между C и D — 10 км.

Между A и C построили новое асфальтовое шоссе длиной 40 км. Оцените минимально возможное время движения велосипедиста из пункта A в пункт B, если его скорость по грунтовой дороге — 20 км/час, по шоссе — 40 км/час.

1) 1 час 2) 1,5 часа 3) 3,5 часа 4) 4 часа

Задача 2008.А16

Из правил соревнования по тяжелой атлетике:

Тяжелая атлетика — это прямое соревнование, когда каждый атлет имеет три попытки в рывке и три попытки в толчке. Самый тяжелый вес поднятой штанги в каждом упражнении суммируется в общем зачете. Если спортсмен потерпел неудачу во всех трех попытках в рывке, он может продолжить соревнование в толчке, но уже не сможет занять какое-либо место по сумме 2-х упражнений.

Если два спортсмена заканчивают состязание с одинаковым итоговым результатом, высшее место присуждается спортсмену с меньшим весом.

Если же вес спортсменов одинаков, преимущество отдается тому, кто первым поднял победный вес.

Таблица результатов соревнований по тяжелой атлетике:

Фамилия И. О.	Вес спортсмена	Взято в рывке	Рывок с попытки	Взято в толчке	Толчок с попытки
Айвазян Г. С.	77,1	150,0	3	200,0	2
Викторов М. П.	79,1	147,5	1	202,5	1
Гордезиани Б. Ш.	78,2	147,5	2	200,0	1
Михальчук М. С.	78,2	147,5	2	202,5	3
Пай С. В.	79,5	150,0	1	200,0	1
Шапсугов М. Х.	77,1	147,5	1	200,0	1

Кто победил в общем зачете (сумме двух упражнений)?

1) Айвазян Г. С. 2) Викторов М. П. 3) Михальчук М. С. 4) Пай С. В.

Задача 2009.А10

Между четырьмя местными аэропортами: ОКТЯБРЬ, БЕРЕГ, КРАСНЫЙ и СОСНОВО, ежедневно выполняются авиарейсы. Приведен фрагмент расписания перелетов между ними:

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

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

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

2°. **Опрос свидетелей**

Задача 2006.B4

Три школьника, Миша (М), Коля (К) и Сергей (С), оставшиеся в классе на перемене, были вызваны к директору по поводу разбитого в это время окна в кабинете. На вопрос директора о том, кто это сделал, мальчики ответили следующее:

Миша: «Я не бил окно, и Коля тоже…»

Коля: «Миша не разбивал окно, это Сергей разбил футбольным мячом!»

Сергей: «Я не делал этого, стекло разбил Миша».

Стало известно, что один из ребят сказал чистую правду, второй в одной части заявления соврал, а другое его высказывание истинно, а третий оба факта исказил. Зная это, директор смог докопаться до истины.

Кто разбил стекло в классе? В ответе запишите только первую букву имени.

Задача 2007.B4

В школьном первенстве по настольному теннису в четверку лучших вошли девушки: Наташа, Маша, Люда и Рита. Самые горячие болельщики высказали свои предположения о распределении мест в дальнейших состязаниях.

Один считает, что первой будет Наташа, а Маша будет второй.

Другой болельщик на второе место прочит Люду, а Рита, по его мнению, займет четвертое место.

Третий любитель тенниса с ними не согласился. Он считает, что Рита займет третье место, а Наташа будет второй.

Когда соревнования закончились, оказалось, что каждый из болельщиков был прав только в одном из своих прогнозов.

Какое место на чемпионате заняли Наташа, Маша, Люда, Рита?

(В ответе перечислите подряд без пробелов числа, соответствующие местам девочек в указанном порядке имен.)

Задача 2008.B4

Перед началом Турнира Четырех болельщики высказали следующие предположения по поводу своих кумиров:

A) Макс победит, Билл — второй;

B) Билл — третий, Ник — первый;

C) Макс — последний, а первый — Джон.

Когда соревнования закончились, оказалось, что каждый из болельщиков был прав только в одном из своих прогнозов.

Какое место на турнире заняли Джон, Ник, Билл, Макс?

(В ответе перечислите подряд без пробелов места участников в указанном порядке имен.)

3°. **Получение заданного числа**

Задача 2007.B3

У исполнителя Калькулятор две команды, которым присвоены номера:

1. **прибавь** 2
2. **умножь на** 3

Выполняя первую из них, Калькулятор прибавляет к числу на экране 2, а выполняя вторую, утраивает его. Запишите порядок команд в программе получения из 0 числа 28, содержащей не более 6 команд, указывая лишь номера команд. (Например, программа 21211 — это программа:

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

которая преобразует число 1 в 19.)

Задача 2008.B3

У исполнителя Утроитель две команды, которым присвоены номера:

1. **вычти** 2
2. **умножь на три**

Первая из них уменьшает число на экране на 2, вторая – утраивает его. Запишите порядок команд в программе получения из 11 числа 13, содержащей не более 5 команд, указывая лишь номера команд.

(Например, 21211 – это программа:

умножь на три
вычти 2
умножь на три
вычти 2
вычти 2,

которая преобразует число 2 в 8).

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

Задача 2009.B5

У исполнителя Калькулятор две команды, которым присвоены номера:

1. **прибавь** 3
2. **умножь на** 4

Выполняя первую из них, Калькулятор прибавляет к числу на экране 3, а выполняя вторую, умножает его на 4. Запишите порядок команд в программе получения из числа 3 числа 57, содержащей не более 6 команд, указывая лишь номера команд.

(Например, программа 21211 — это программа:
умножь на 4
прибавь 3
умножь на 4
прибавь 3
прибавь 3,
которая преобразует число 2 в 50.)

4°. Логическая игра

Задача 2006.С3

Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 5, а во второй — 3 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или удваивает число камней в какой-то куче, или добавляет 4 камня в какую-то кучу. Выигрывает игрок, после хода которого в одной из куч становится не менее 22 камней. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Как должен ходить выигрывающий игрок? Ответ обоснуйте.

Задача 2007.С3

Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 3, а во второй — 2 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 3 раза число камней в какой-то куче, или добавляет 1 камень в какую-то кучу. Выигрывает игрок, после хода которого общее число камней в двух кучах становится не менее 16 камней. Кто выигрывает при безошибочной игре — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

Задача 2008.С3

Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 1, а во второй — 2 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 3 раза число камней в какой-то куче, или добавляет 2 камня в какую-то кучу. Выигрывает игрок, после хода которого общее число камней в двух кучах становится не менее 17 камней. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

Задача 2009.С3

Два игрока играют в следующую игру. На координатной плоскости стоит фишка. Игроки ходят по очереди. В начале игры фишка находится в точке с координатами (5, 2). Ход состоит в том, что игрок перемещает фишку из точки с координатами (x, y) в одну из трех точек: или в точку с координатами $(x + 3, y)$, или в точку с координатами $(x, y + 3)$, или в точку с координатами $(x, y + 4)$. Выигрывает игрок, после хода которого расстояние по прямой от фишки до точки с координатами (0, 0) не меньше 13 единиц. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

4. Ответы

1°. Максимумы и минимумы

Задача 2007.A12. 3)

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

Задача 2008.A12. 3) 3,5 часа.

Задача 2008.A16. 1) Айвазян Г. С.

Задача 2009.A10. 4) 17:25.

2°. Опрос свидетелей

Задача 2006.B4. М.

Задача 2007.B4. 1423.

Задача 2008.B4. 3124.

3°. Получение заданного числа

Задача 2007.B3. 121211.

Задача 2008.B3. 11121.

Задача 2009.B5. 22111.

4°. Логическая игра

Задача 2006.C3. Выигрывает первый игрок. Первый ход: удвоение количества камней во второй кучке. Обоснование: полное дерево игры или логические рассуждения с конца.

Задача 2007.C3. Выигрывает второй игрок. Первый ход: второй игрок утраивает кучку, в которой больше 4 камней, а если этого сделать нельзя, то прибавляет один камень к меньшей кучке. Обоснование: полное дерево игры или логические рассуждения с конца.

Задача 2008.C3. Выигрывает второй игрок. Первый ход: второй игрок прибавляет 2 камня в кучку, которую оставил без изменения первый игрок. Обоснование: полное дерево игры или логические рассуждения с конца.

Задача 2009.C3. Выигрывает второй игрок. Первый ход: второй игрок увеличивает на 3 меньшую координату, а если координаты после первого хода 1-го игрока одинаковые, то только левую координату. Обоснование: полное дерево игры или логические рассуждения с конца.

5. Решения

1°. Максимумы и минимумы

Рекомендации. 1. Выбор вариантов достаточно легок.

2. Чтобы получить минимум или максимум, нужно найти все варианты. Помогает перевод данных в удобную форму.

3. Ответ вычисляется и его выбор трудностей не вызывает.

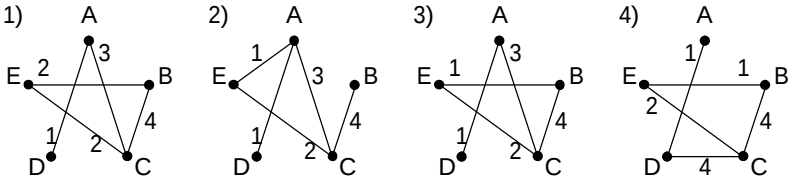
Задача 2007.A12

Ответ: 3)

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

Решение. Задание состоит в том, чтобы указать таблицу, в которой имеется стоимость проезда из A в B не больше 6. Здесь трудно допустить ошибку: если такая таблица не найдена, нужно просто начать искать сначала.

Чтобы было лучше видно, какими способами можно проехать из A в B и поэтому быстрее решить задачу, перерисуем таблицы в виде графов.



Осталось подсчитать стоимость проезда из A в B по всем возможным маршрутам, в которых станции не повторяются.

1) $A \to C \to B = 3 + 4 = 7$.

$A \to C \to E \to B = 3 + 2 + 2 = 7$.

2) $A \to C \to B = 3 + 4 = 7$.

$A \to E \to C \to B = 1 + 2 + 4 = 7$.

3) $A \to C \to B = 3 + 4 = 7$.

$A \to E \to C \to B = 3 + 4 + 1 = 6$.

4) $A \to D \to C \to B = 1 + 4 + 4 = 9$.

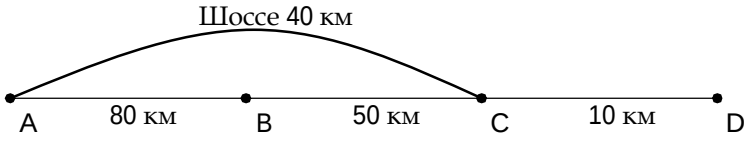
$A \to D \to C \to E \to B = 1 + 4 + 2 + 1 = 8$.

Теперь видно, что только в таблице 3) есть маршрут со стоимостью проезда A в B не больше 6.

Задача 2008.A12

Ответ: 3) 3,5 часа.

Решение. Нарисуем схему дорог.



Из пункта A в пункт B можно добраться двумя способами:

1) по грунтовой дороге сразу из A в B;

2) по шоссе из A в C, а затем по грунтовой дороге из C в B.

Поездку в D не рассматриваем, она только удлинит любую дорогу.

Вычислим, сколько времени займет каждый из двух способов.

1-й: $\dfrac{80\text{ км}}{20\text{ км/час}} = 4$ часа.

2-й: $\dfrac{40\text{ км}}{40\text{ км/час}} + \dfrac{50\text{ км}}{20\text{ км/час}} = 1\text{ час} + 2{,}5\text{ часа} = 3{,}5\text{ часа}$.

2-й способ быстрее 1-го и занимает 3,5 часа.

Задача 2008.A16

Ответ: 1) Айвазян Г. С.

Решение. Посчитаем сумму весов штанги для двух попыток. Удобно перерисовать таблицу, дописав еще один столбец с суммой взятых весов.

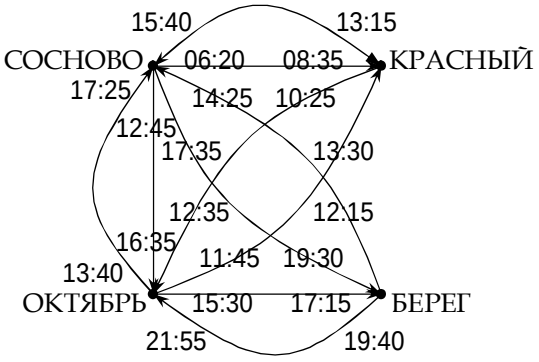
Фамилия И. О.	Вес спортсмена	Взято в рывке	Рывок с попытки	Взято в толчке	Толчок с попытки	Сумма
Айвазян Г. С.	77,1	150,0	3	200,0	2	350,0
Викторов М. П.	79,1	147,5	1	202,5	1	350,0
Гордезиани Б. Ш.	78,2	147,5	2	200,0	1	247,5
Михальчук М. С.	78,2	147,5	2	202,5	3	350,0
Пай С. В.	79,5	150,0	1	200,0	1	350,0
Шапсугов М. Х.	77,1	147,5	1	200,0	1	247,5

Имеем четырех спортсменов, набравших максимальную сумму. Из них меньший вес имеет только один — Айвазян Г. С.

Задача 2009.A10

Ответ: 4) 17:25.

Решение. Чтобы было лучше видно, какими способами можно долететь из аэропорта ОКТЯБРЬ в аэропорт СОСНОВО и поэтому быстрее решить задачу, перерисуем таблицу в виде графа.



Осталось пролететь из аэропорта ОКТЯБРЬ в аэропорт СОСНОВО по всем маршрутам, в которых аэропорты не повторяются.

ОКТЯБРЬ → СОСНОВО:

13:40 → 17:25.

ОКТЯБРЬ → БЕРЕГ → СОСНОВО:

15:30 → 17:15 → нет рейса после 17:15 в этот же день.

ОКТЯБРЬ → КРАСНЫЙ → СОСНОВО:

11:45 → 13:30 → нет рейса после 13:30 в этот же день.

Теперь видно, что из аэропорта ОКТЯБРЬ в аэропорт СОСНОВО можно попасть только по одному маршруту, чтобы долететь в тот же день. Этот маршрут ОКТЯБРЬ → СОСНОВО, время прилета в СОСНОВО 17:25.

2°. **Опрос свидетелей**

Рекомендации. 1. Выбор вариантов нужно проводить достаточно тщательно, чтобы минимизировать количество исходных вариантов.

2. Достаточно выбрать первоначальный вариант, рассмотрев все варианты какого-то одного логического условия. После этого все остальные логические условия раскрываются однозначно и вариантов не имеют.

3. Ответ просто вычисляется.

Задача 2006.B4

Ответ: М.

Решение. Рассмотрим три варианта:

1) окно разбил Миша;

2) окно разбил Коля;

3) окно разбил Сергей.

1. Если окно разбил Миша, то у него одно высказывание ложно, а другое истинно. У Коли оба утверждения ложны. А у Сергея оба утверждения истинны.

То, что окно разбил Миша, удовлетворяет условиям задачи.

Для проверки рассмотрим оставшиеся варианты.

2. Если окно разбил Коля, то Миша один раз сказал правду, а один раз нет. И Коля тоже один раз сказал правду, а один раз нет. Этот вариант не подходит.

3. Если окно разбил Сергей, то и Миша оба раза сказал правду, и Коля. Этот вариант не подходит.

Задача 2007.B4

Ответ: 1423.

Указание. Рассмотрите два варианта: 1) Наташа первая, а Маша не вторая; 2) Наташа не первая, а Маша вторая.

Задача 2008.B4

Ответ: 3124.

Указание. Рассмотрите два варианта: 1) Макс победит, Билл — не второй; 2) Макс не победит, Билл — второй.

3°. **Получение заданного числа**

Рекомендации. 1. Внимательно изучаем данные команды.

2. Решаем либо «в лоб» перебором всех вариантов, либо логическим путем, начиная с конца.

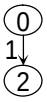
3. Ответ просто вычисляется.

Задача 2007.B3

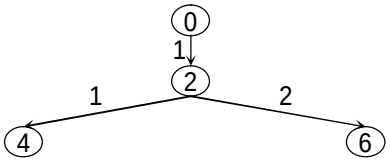
Ответ: 121211.

Решение 1. Начнем выписывать все числа, которые можно получить указанными командами из числа 0. Причем не будем применять команду, которая не меняет число, т. е. не будем умножать 0 на 3.

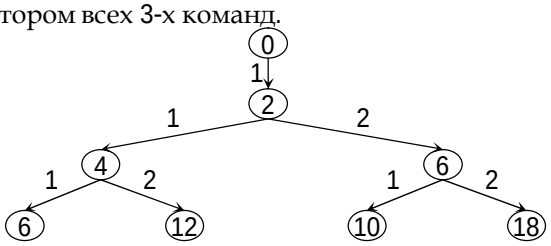
Запишем одно новое число, получающееся после выполнения калькулятором 1-й команды — **прибавь** 2.



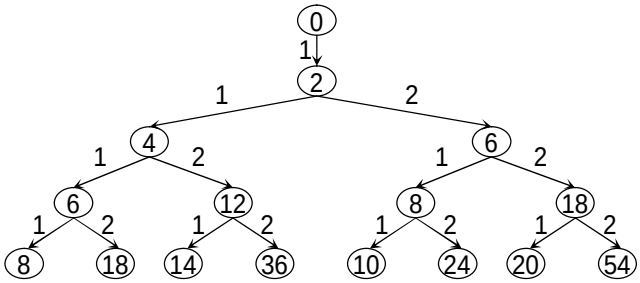
Запишем два новых числа, получающихся после выполнения калькулятором обеих 2-х команд.



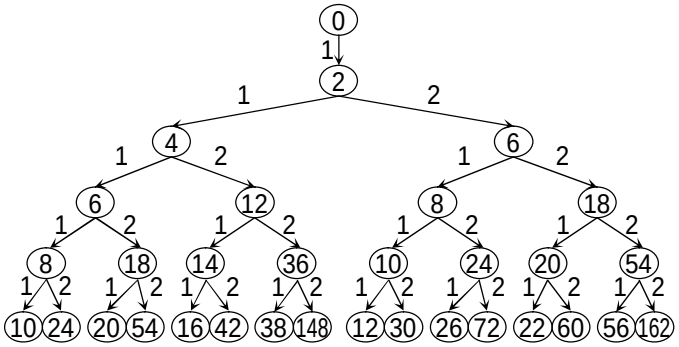
Запишем четыре числа, получающихся после выполнения калькулятором всех 3-х команд.



Запишем числа, получающихся после выполнения калькулятором всех 4-х команд.



Запишем числа, получающихся после выполнения калькулятором всех 5-х команд.



Из чисел, полученных после выполнения 5-й команды, найдем число, которое одной командой превращается в 28. Это 26, а команда — **прибавь** 2, поскольку 28 на 3 не делится.

Получаем, что для того, чтобы из 0 получить 28, нужно выполнить команды 121211.

Решение 2. Начнем решать задачу с конца. Какими должны быть последние команды?

28 на 3 не делится, поэтому последние команды — **прибавь** 2, которые действуют на число, делящееся на 3 и ближайшее к 28 снизу. Это число 24. Имеем три последние команды 211, которые действуют на число 8.

8 на 3 не делится, поэтому последние команды — **прибавь** 2, которые действуют на число, делящееся на 3 и ближайшее к 8 снизу. Это число 6. Имеем еще две команды 21, которые действуют на число 2 и получают число 8.

Наконец, число 2 получается из числа 0 командой 1.

Итак, имеем программу 121211.

Задача 2008.B3

Ответ: 11121.

Решение. Решение полностью аналогично решению предыдущей задачи.

Задача 2009.B5

Ответ: 22111.

Решение. Решение полностью аналогично решению предыдущей задачи.

4°. Логическая игра

Рекомендации. 1. Внимательно изучаем правила игры.

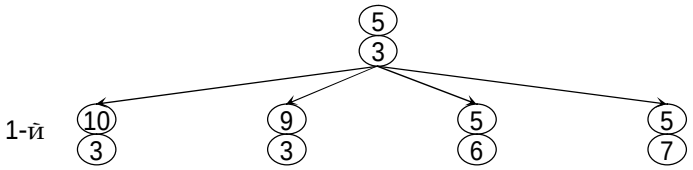
2. Решаем либо «в лоб» перебором всех вариантов, либо логическим путем, начиная с конца.

3. Это задача с развернутым ответом. Другими словами, нужно записать решение полностью и как можно более подробно. В решении обязательно должно содержаться доказательство ответа. Запись ответа в этой задаче составляет некоторую трудность, поэтому также пишите ответ как можно более подробно.

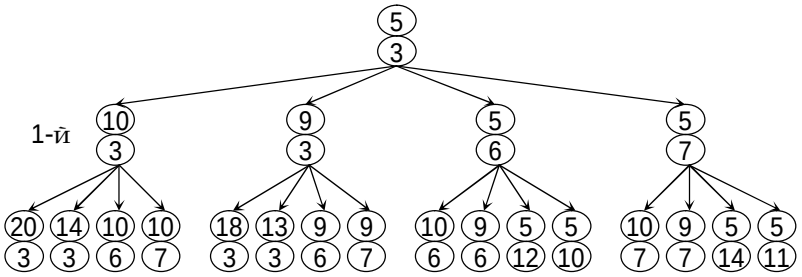
Задача 2006.C3

Ответ. Выигрывает первый игрок. Первый ход: удвоение количества камней во второй кучке. Обоснование: полное дерево игры или логические рассуждения с конца.

Решение 1. Нарисуем все варианты хода 1-го игрока.

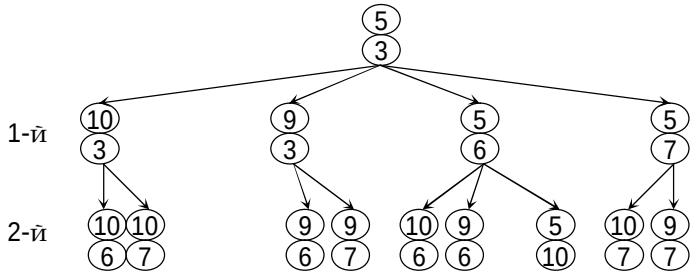


Теперь нарисуем все варианты ответа 2-го игрока.

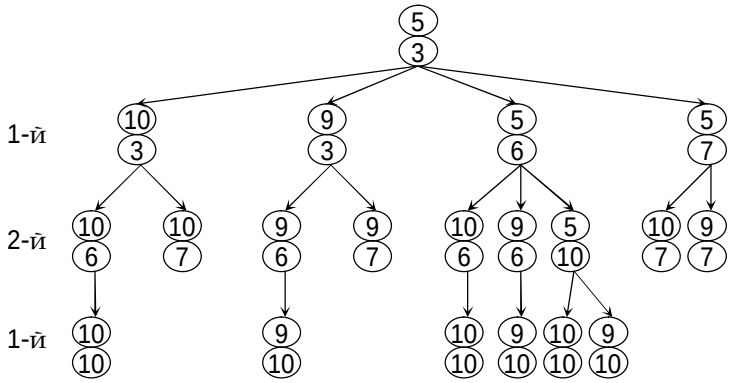


Очевидно, что если в одной из куч не менее 11 камней, то тот игрок, который ходит, сразу выигрывает, удваивая это количество камней.

Игроки должны играть, не поддаваясь! Поэтому исключим ответы 2-го игрока, которые сразу ведут к его проигрышу.



Нарисуем все варианты второго хода 1-го игрока, но только те, которые не ведут к его немедленному проигрышу.



Ясно видно, что при любом ответе 2-го игрока на нарисованный второй ход 1-го 2-й игрок проигрывает.

Итак, при первых ходах 1-го игрока (10, 3), (9, 3) и (5, 7) 2-й игрок может так походить, что 1-й игрок проиграет.

Но 1-й игрок волен выбирать свой первый ход. Если первый ход 1-го игрока будет (5, 6), то при любом ответе 2-го игрока 1-й может походить так, что выиграет.

Ответ: выигрывает 1-й игрок ходом (5, 6). Обоснование: приведенный рисунок и рассуждения выше.

Решение 2. Начнем решать задачу с конца.

1. Каким образом можно получить не менее 22 камней? В принципе, двумя способами:

1) удвоить кучку с не менее чем 11 камнями;

2) добавить 4 камня в кучку с не менее чем 18 камнями.

Очевидно, что если у игрока будет кучка с 11 камнями, то он не будет дожидаться кучки с 18 камнями, а сразу выиграет.

Итак, чтобы выиграть, нужно удвоить кучку с не менее чем 11 камнями.

2. А чтобы не проиграть, нельзя делать кучку с более чем 10 камнями. Поэтому если в каждой кучке будет более 6 камней, но менее 11, то тот, кто ходит, проиграет. Любой его ход приведет к кучке с более чем 10 камнями.

3. Поэтому если в одной кучке от 7 до 10 камней, а в другой кучке от 3 до 6, то тот, кто ходит, выиграет, просто добавив в меньшую кучку 4 камня.

4. Поэтому если в каждой кучке от 4 до 6 камней, то тот, кто ходит, проиграет: любой его ход делает в одной из кучек более 6 камней.

5. Поэтому если в одной кучке от 4 до 6 камней, а в другой кучке от 2 до 3, то тот, кто ходит, выиграет, удвоив количество камней в меньшей кучке.

В исходной позиции в одной кучке 5 камней, а во второй — 3. Поэтому выиграет тот, кто ходит первым. Ему нужно удвоить количество камней в меньшей кучке. Обоснование: предыдущие рассуждения с конца.

Задача 2007.C3

Ответ. Выигрывает второй игрок. Первый ход: второй игрок утраивает кучку, в которой больше 4 камней, а если этого сделать нельзя, то прибавляет один камень к меньшей кучке. Обоснование: полное дерево игры.

Решение. Решение полностью аналогично решению предыдущей задачи.

Задача 2008.C3

Ответ. Выигрывает второй игрок. Первый ход: второй игрок прибавляет 2 камня в кучку, которую оставил без изменения первый игрок. Обоснование: полное дерево игры.

Решение. Решение полностью аналогично решению предыдущей задачи.

Задача 2009.C3

Ответ. Выигрывает второй игрок. Первый ход: второй игрок увеличивает на 3 меньшую координату, а если координаты после первого хода 1-го игрока одинаковые, то только левую координату. Обоснование: полное дерево игры.

Решение. Решение полностью аналогично решению предыдущей задачи.

Когда программист ложится спать, то на тумбочку рядом с кроватью ставит два стакана: один с водой, другой пустой. Стакан с водой на случай, если пить захочется, пустой — если не захочется.

Анекдот

Глава 3
Алгоритмы

§ 1. Управление исполнителем

1. Теория

1°. Алгоритм и его формальное выполнение

Алгоритм — базовое понятие для программирования.

Алгоритм — описание последовательности действий для достижения конкретного результата.

Для возникновения какого-нибудь конкретного алгоритма совершенно необходимо наличие двух субъектов.

Составитель алгоритма — человек или группа людей, которые являются авторами алгоритма.

Исполнитель алгоритма — реальное или воображаемое устройство, которое выполняет алгоритм.

Исполнителем алгоритма может быть и человек. В математике исполнителем алгоритма обычно является специальная воображаемая машина. Универсальным исполнителем является компьютер.

Имеется два вида понимания алгоритма.

Понимание результата алгоритма — понимание того, зачем нужен алгоритм и какой результат достигается при его выполнении.

Понимание работы алгоритма — понимание того, каким образом при выполнении алгоритма достигается его результат.

Формальное выполнение алгоритма — выполнение алгоритма без какого-либо его понимания, когда исполнитель способен механически выполнять указанные действия в указанной последовательности.

Именно принципиальная возможность *формального выполнения алгоритма* является его важной характеристикой!

За алгоритм отвечает его составитель, который обязан не только понимать результат алгоритма, но и проверить правильность его выполнения, являясь первым исполнителем алгоритма.

Понимание алгоритма желательно, но не обязательно.

2°. **Дискретность, пошаговость и конечность алгоритма**

Рассмотрим первые два свойства алгоритма: *дискретность* и *модульность*.

Шаги, или такты, алгоритма — действия, из последовательности которых состоит алгоритм. Другими словами, *алгоритм* — это последовательность шагов.

Дискретность алгоритма — свойство алгоритма состоять из последовательности отдельных шагов по времени.

Искусство программирования состоит в том, чтобы решить поставленную задачу, используя наиболее удачный алгоритм.

Решить задачу — представить решение задачи в виде последовательности шагов, желательно наиболее удачных.

Шаги алгоритма выполняются не хаотично, а последовательно друг за другом. Все шаги алгоритма выстроены в цепочку от шага с номером 1 до шага с номером n, где n — количество шагов алгоритма.

Пошаговость, или последовательность, алгоритма — свойство алгоритма выполнять свои шаги один за другим по одному разу.

Следующее свойство алгоритма — следствие его дискретности и последовательности.

Конечность, или результативность, алгоритма — конечность последовательности шагов алгоритма, при выполнении которых алгоритм заканчивается.

Результативность алгоритма означает, что алгоритм должен гарантированно привести к решению задачи за конечное число шагов. При этом исполнитель алгоритма вполне может быть не реальным, а воображаемым. Если количество шагов алгоритма конечно, но таково, что его нельзя выполнить ни на каком компьютере, то такой алгоритм может быть выполнен только в воображении человека.

3°. **Массовость, однозначность и устойчивость алгоритма**

Рассмотрим два свойства алгоритма: *массовость* и *однозначность*.

Массовость, или абстрактность, алгоритма — свойство алгоритма быть использованным многократно.

Это тривиальное свойство очень важно. Алгоритм является абстрактной записью идей и может быть реализован в любой нужный момент.

Однозначность, или определенность, алгоритма — свойство алгоритма, состоящее в том, что каждый его шаг точно определен.

Однозначность вовсе не означает одинаковость результата.

Иногда однозначность трактуют как получение на каждом шаге одинакового результата при одинаковых условиях. Но это справедливо только для *детерминированных* алгоритмов.

Детерминированный шаг алгоритма выдает каждый раз один и тот же результат при одинаковых условиях, *недетерминированный, или случайный, шаг алгоритма* выдает все время разные результаты при равных условиях.

Детерминированный алгоритм — алгоритм, все шаги которого детерминированы, *недетерминированный, или случайный, алгоритм* — алгоритм, имеющий хотя бы один недетерминированный шаг.

Недетерминированным шагом может быть, например, вычисление случайного числа.

Следствием массовости и однозначности алгоритма является то, что он используется для решения разных однотипных задач, которые отличаются друг от друга исходными условиями.

Устойчивость, или универсальность, алгоритма — алгоритм предназначен для решения определенного класса задач, отличающихся только начальными условиями.

4°. **Результат выполнения алгоритма**

Рассмотрим, что должно получаться в результате выполнения основных алгоритмов. Как они работают, как достигается конечный результат, нас сейчас не интересует. Тем более что эти алгоритмы могут работать различными способами.

Для понимания *результата* работы алгоритма необходимо подробно описать две серии данных:

1) данные, которые подаются на вход алгоритма. Эти данные называют *входными данными* алгоритма;

2) данные, которые получаются на выходе алгоритма. Эти данные называют *выходными данными* алгоритма.

Что происходит *внутри* алгоритма, как он достигает своей *цели*, т. е. как перерабатывает входные данные в выходные, рассматривать здесь не будем. Другими словами, будем рассматривать работу основных алгоритмов как работу *черных ящиков*.

Реализацией алгоритма называется способ достижения алгоритмом своей цели.

Различные реализации некоторых алгоритмов будут рассмотрены в последнем параграфе.

Введем некоторые определения, которые нам потребуются уже на этапе изучения входных и выходных параметров алгоритмов.

В теории алгоритмов и в программировании последовательность чисел называется *массивом*, числа массива — *элементами массива*, а количество чисел в массиве — *длиной масива*.

Сортировка массива чисел — перестановка чисел в порядке неубывания.

Сортировку также называют *сортировкой по неубыванию*, а если все числа разные, то *сортировкой по возрастанию*.

Сортировка по невозрастанию массива чисел — перестановка чисел в порядке невозрастания. Если все числа массива разные, то такая сортировка называется *сортировкой по убыванию*.

А теперь перечислим некоторые основные алгоритмы с точки зрения их результата и опишем их входные и выходные данные.

5°. **Алгоритмы вычисления делителей целых чисел**

1. Вычисление наибольшего общего делителя l двух целых положительных чисел n и m. Математически это записывается так: $l = \text{НОД}(n, m)$.

Вход алгоритма: 2 целых положительных числа n и m.

Выход алгоритма: 1 целое положительное число l, наибольшее из тех, на которые делятся нацело как n, так и m.

Например:

если $n = m$, то $l = n = m$;

если n делится на m, то $l = m$;

если m делится на n, то $l = n$;

если $n = 4$, а $m = 6$, то $l = 2$;

если $n = 10$, а $m = 4$, то $l = 2$.

2. Вычисление всех делителей целого положительного числа.

Вход алгоритма: 1 целое положительное число n.

Выход алгоритма:

1) 1 целое положительное число m — количество всех делителей числа n;

2) массив l из m целых положительных чисел — делителей числа n.

Например:

если $n = 2$, то $m = 2$, $l = 1, 2$;

если $n = 3$, то $m = 2$, $l = 1, 3$;

если $n = 4$, то $m = 3$, $l = 1, 2, 4$;

если $n = 6$, то $m = 4$, $l = 1, 2, 3, 6$.

6°. **Алгоритмы решения уравнений**

1. Решение линейного уравнения $ax + b = 0$.

Вход алгоритма: 2 числа: a — коэффициент при x и b — свободный коэффициент.

Выход алгоритма:

либо сообщение «любой x» (если $a = 0$, $b = 0$);

либо сообщение «решений нет» (если $a = 0$, $b \neq 0$);

либо число $x = -b / a$ (если $a \neq 0$).

Например:
если $a = 0{,}0$, $b = 0{,}0$, то вывод сообщения «любой x»;
если $a = 0{,}0$, $b = 1{,}0$, то вывод сообщения «решений нет»;
если $a = 1{,}0$, $b = 0{,}0$, то $x = 0{,}0$;
если $a = 1{,}0$, $b = 1{,}0$, то $x = -1{,}0$.

2. Решение квадратного уравнения $ax^2 + bx + c = 0$.
Вход алгоритма: 3 числа: a — коэффициент при x^2, b — коэффициент при x и c — свободный коэффициент.
Выход алгоритма:
либо сообщение «любой x» (если $a = 0$, $b = 0$, $c = 0$);
либо сообщение «решений нет» (если $a = 0$, $b = 0$, $c \neq 0$);
либо число $x = -c / b$ (если $a = 0$, $b \neq 0$);
либо сообщение «решений нет» (если $a \neq 0$, $b^2 - 4ac < 0$);
либо число $x = -b / 2a$ (если $a \neq 0$, $b^2 - 4ac = 0$);
либо 2 числа $x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$, $x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ (если $a \neq 0$, $b^2 - 4ac > 0$).
Например:
если $a = 0{,}0$, $b = 0{,}0$, $c = 0{,}0$, то вывод сообщения «любой x»;
если $a = 0{,}0$, $b = 0{,}0$, $c = 1{,}0$, то вывод сообщения «решений нет»;
если $a = 0{,}0$, $b = 1{,}0$, $c = 0{,}0$, то $x = 0{,}0$;
если $a = 0{,}0$, $b = 1{,}0$, $c = 1{,}0$, то $x = -1{,}0$;
если $a = 1{,}0$, $b = 0{,}0$, $c = 0{,}0$, то $x = 0{,}0$;
если $a = 1{,}0$, $b = 0{,}0$, $c = 1{,}0$, то вывод сообщения «решений нет»;
если $a = 1{,}0$, $b = 1{,}0$, $c = 0{,}0$, то $x_1 = -1{,}0$, $x_2 = 0{,}0$;
если $a = 1{,}0$, $b = 1{,}0$, $c = 1{,}0$, то вывод сообщения «решений нет».

7°. **Алгоритмы вычисления суммы чисел**

1. Вычисление суммы s значений функции f, зависящей от целого числа i, когда i пробегает все значения от 1 до n.

Математически это записывается так:

$$s = f(1) + f(2) + \text{h} + f(n-1) + f(n) = \sum_{i=1}^{n} f(i).$$

Вход алгоритма: 1 целое положительное число n — количество слагаемых в сумме.

Выход алгоритма: 1 число s — сумма значений функции f при параметре i, пробегающем все значения от 1 до n.

Например:

если $f = 1/i$, $n = 2$, то $s = 1{,}5$;

если $f = 1/i$, $n = 3$, то $s = 11/6$.

2. Вычисление суммы s массива чисел a длиной n. (Определение массива см. в п. 4°.) Математически:

$$s = a_1 + a_2 + \text{h} + a_{n-1} + a_n = \sum_{i=1}^{n} a_i.$$

Вход алгоритма:

1) 1 целое положительное число n — длина массива a;

2) n чисел, организованных в виде массива a.

Выход алгоритма: 1 число s — сумма массива a.

Например:

если $n = 2$, $a = 1, 2$, то $s = 3$;

если $n = 2$, $a = 2{,}5, 2{,}0$, то $s = 4{,}5$.

8°. **Алгоритмы поиска**

1. Поиск заданного числа b в массиве чисел a длиной n.

Вход алгоритма:

1) 1 целое положительное число n — длина массива a;

2) n чисел, организованных в виде массива a;

3) 1 число b.

Выход алгоритма:

либо значение переменной $m = 1$, если число b является одним из элементов массива, и $m = 0$ в противном случае;

либо вывод сообщения «число найдено», если число b входит в массив, и сообщения «число не найдено», если не входит.

Например:
если $n = 3$, $a = 1, 3, 2$, $b = 2$, то $m = 1$;
если $n = 3$, $a = 2{,}5, 2{,}0, 1{,}5$, $b = 0{,}5$, то $m = 0$;
если $n = 3$, $a = 1, 2, 4$, $b = 2$, то $m = 1$.

2. Поиск k индексов вхождений l заданного числа b в массив чисел a длиной n.

Вход алгоритма:
1) 1 целое положительное число n — длина массива a;
2) n чисел, организованных в виде массива a;
3) 1 число b.

Выход алгоритма:
1) 1 целое положительное число k — длина массива l;
2) k чисел, организованных в виде массива l.

Например:
если $n = 3$, $a = 1, 2, 2$, $b = 2$, то $k = 2$, $l = 2, 3$;
если $n = 3$, $a = 2{,}5, 2{,}0, 2{,}5$, $b = 0{,}5$, то $k = 0$;
если $n = 3$, $a = 1, 2, 4$, $b = 2$, то $k = 1$, $l = 2$.

9°. **Алгоритмы вычисления экстремальных значений**

1. Вычисление максимального элемента m массива чисел a длиной n. Математически: $m = \max(a_1, a_2, \text{h}, a_{n-1}, a_n)$.

Вход алгоритма:
1) 1 целое положительное число n — длина массива a;
2) n чисел, организованных в виде массива a.

Выход алгоритма: 1 число m — максимум массива a.

Например:
если $n = 3$, $a = 1, 3, 2$, то $m = 3$;
если $n = 3$, $a = 2{,}5, 2{,}0, 1{,}5$, то $m = 2{,}5$;
если $n = 3$, $a = 1, 2, 4$, то $m = 4$.

2. Вычисление всех k индексов l максимальных элементов m массива чисел a длиной n.

Вход алгоритма:
1) 1 целое положительное число n — длина массива a;
2) n чисел, организованных в виде массива a.
Выход алгоритма:
1) 1 целое положительное число k — длина массива l;
2) k чисел, организованных в виде массива l.
Например:
если $n = 3$, $a = 1, 2, 2$, то $k = 2$, $l = 2, 3$;
если $n = 3$, $a = 2{,}5, 2{,}0, 2{,}5$, то $k = 2$, $l = 1, 3$;
если $n = 3$, $a = 1, 2, 4$, то $k = 1$, $l = 3$.

10°. **Алгоритмы сортировки**

1. Сортировка массива чисел a длиной n. (Определение сортировки см. в п. 4°.)
Вход алгоритма:
1) 1 целое положительное число n — длина массива a;
2) n чисел, организованных в виде массива a.
Выход алгоритма: n отсортированных чисел массива a.
Например:
если $n = 3$, $a = 3, 1, 2$, то $a = 1, 2, 3$;
если $n = 3$, $a = 2{,}5, 2{,}0, 1{,}5$, то $a = 1{,}5, 2{,}0, 2{,}5$;
если $n = 3$, $a = 1, 2, 4$, то $a = 1, 2, 4$.

2. Сортировка по невозрастанию массива чисел a длиной n.
Вход алгоритма:
1) 1 целое положительное число n — длина массива a;
2) n чисел, организованных в виде массива a.
Выход алгоритма: n отсортированных по невозрастанию чисел массива a.
Например:
если $n = 3$, $a = 3, 1, 2$, то $a = 3, 2, 1$;
если $n = 3$, $a = 2{,}5, 2{,}0, 1{,}5$, то $a = 2{,}5, 2{,}0, 1{,}5$;
если $n = 3$, $a = 1, 2, 4$, то $a = 4, 2, 1$.

2. **Алгоритмы**

Напомним, что в этом разделе приведены не алгоритмы как предмет изучения, а технологии решения заданий по ЕГЭ по информатике.

Алгоритм 1. Выполнение алгоритма исполнителем.

1. Команды алгоритма исполняются строго последовательно без пропусков, начиная с начала алгоритма и заканчивая на его конце.

2. Повторение конкретной команды происходит до тех пор, пока условие ее повторения истинно. Как только условие повторения команды становится ложным, команда не выполняется и нужно сразу перейти к следующей команде.

3. Задачи

1°. **Запросы**

Задача 2007.A16

На городской олимпиаде по программированию предлагались задачи трех типов: A, B и C. По итогам олимпиады была составлена таблица, в колонках которой указано, сколько задач каждого типа решил участник. Вот начало таблицы:

Фамилия	A	B	C
Иванов	3	2	1

За правильное решение задачи типа A участнику начислялся 1 балл, за решение задачи типа B — 2 балла и за решение задачи типа C — 3 балла. Победитель определялся по сумме баллов, которая у всех участников оказалась разная. Для определения победителя олимпиады достаточно выполнить следующий запрос.

1) Отсортировать таблицу по возрастанию значения поля C и взять первую строку.
2) Отсортировать таблицу по убыванию значения поля C и взять первую строку.
3) Отсортировать таблицу по убыванию значения выражения A + 2B + 3C и взять первую строку.
4) Отсортировать таблицу по возрастанию значения выражения A + 2B + 3C и взять первую строку.

2°. **Исполнитель**

Задача 2006.A20

Исполнитель Черепашка перемещается на экране компьютера, оставляя след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существуют две команды:

Вперед *n*, вызывающая передвижение Черепашки на *n* шагов в направлении движения.

Направо *m*, вызывающая изменение направления движения на *m* градусов по часовой стрелке.

(Вместо *n* и *m* должны стоять целые числа).

Запись:

Повтори 5 [**Команда**1 **Команда**2]

означает, что последовательность команд в квадратных скобках повторится 5 раз.

Какое число необходимо записать вместо *n* в следующем алгоритме:

Повтори 7 [**Вперед** 40 **Направо** *n*],

чтобы на экране появился правильный шестиугольник?

1) 30 2) 45 3) 50 4) 60

Задача 2006.B3

Исполнитель Робот действует на клетчатой доске, между соседними клетками которой могут стоять стены. Робот передвигается по клеткам доски и может выполнять команды 1 (вверх), 2 (вниз), 3 (вправо), 4 (влево), переходя на соседнюю клетку в направлении, указанном в скобках. Если в этом направлении между клетками стоит стена, то Робот разрушается. Робот успешно выполнил программу 3233241.

Какую последовательность из трех команд должен выполнить Робот, чтобы вернуться в ту клетку, где он был перед началом выполнения программы, и не разрушиться вне зависимости от того, какие стены стоят на поле?

3°. Перебор вариантов исполнителем

Задача 2008.A20

Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	**вниз**	**влево**	**вправо**

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

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

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

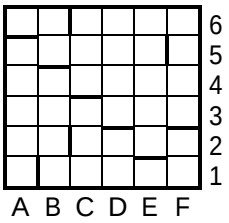
Цикл

ПОКА < *условие* > *команда*

выполняется, пока условие истинно, иначе происходит переход на следующую строку.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ остановится в той же клетке, с которой он начал движение?

НАЧАЛО
ПОКА < **справа свободно** > **вправо**
ПОКА < **сверху свободно** > **вверх**
ПОКА < **слева свободно** > **влево**
ПОКА < **снизу свободно** > **вниз**
КОНЕЦ



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

Задача 2009.А18

Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	**вниз**	**влево**	**вправо**

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

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

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

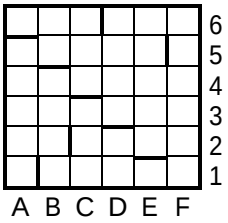
Цикл

ПОКА < *условие* > *команда*

выполняется, пока условие истинно, иначе происходит переход на следующую строку.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную ниже программу, РОБОТ остановится в той же клетке, с которой он начал движение?

НАЧАЛО
ПОКА < **снизу свободно** > **вниз**
ПОКА < **слева свободно** > **влево**
ПОКА < **сверху свободно** > **вверх**
ПОКА < **справа свободно** > **вправо**
КОНЕЦ



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

4. Ответы

1°. Запросы

Задача 2007.А16. 3) Отсортировать таблицу по убыванию значения выражения A + 2B + 3C и взять первую строку.

2°. Исполнитель

Задача 2006.А20. 4) 60.
Задача 2006.В3. 414.

3°. Перебор вариантов исполнителем

Задача 2008.А20. 4) 4.
Задача 2009.А18. 1) 1.

5. Решения

1°. **Запросы**

Рекомендации. 1. Выясняем, по какому параметру нужна сортировка.

2. Вспоминаем, что такое сортировка.

3. Ответы трудностей не вызывают.

Задача 2007.А16

Ответ: 3) Отсортировать таблицу по убыванию значения выражения A + 2B + 3C и взять первую строку.

Решение. Как следует из условия задачи, A — это количество решенных участником задач типа A, B — количество решенных задач типа B, а C — типа C. Тогда количество баллов, которые набрал участник, равно A + 2B + 3C.

Получаем, что участников нужно сортировать по значению выражения A + 2B + 3C. Поэтому подходит либо запрос 3), либо запрос 4).

И в запросе 3), и в запросе 4) берется первая строка. Поэтому сортировать нужно так, чтобы в первой строке был наибольший результат. А это сортировка по убыванию.

Подходит запрос 3).

2°. **Исполнитель**

Рекомендации. 1. Внимательно изучите команды исполнителя.

2. Решение получается механически выполнением алгоритма.

3. Ответы трудностей не вызывают.

Задача 2006.А20

Ответ: 4) 60.

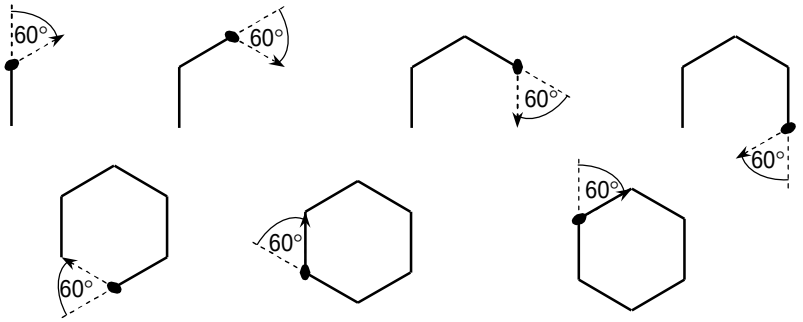
Решение. Чтобы появился правильный шестиугольник, Черепашка после шестого поворота должна вернуться на исходное направление. Другими словами, Черепашка должна за шесть поворотов повернуть на 360°. Поэтому каждый раз Черепашка должна поворачивать на 60°.

Проверим ответ, т. е. нарисуем движение черепашки по алгоритму:

Повтори 7 [**Вперед** 40 **Направо** 60]

Команды, заключенные в квадратных скобках, Черепашка повторит семь раз.

Нарисуем то, что получается после 1-го выполнения Черепашкой команд **Вперед** 40 **Направо** 60, затем то, что получается после 2-го выполнения этих команд, после 3-го, после 4-го, после 5-го, после 6-го и после 7-го.

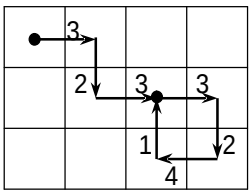


Получился шестиугольник.

Задача 2006.B3

Ответ: 414.

Решение. Нарисуем путь робота и посмотрим, как ему вернуться обратно по старой дороге.



Итак, чтобы вернуться, роботу нужно пойти налево, потом вверх и снова налево, т. е. выполнить команды 414.

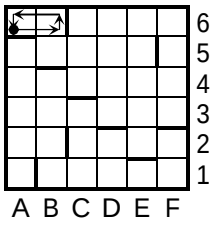
3°. **Перебор вариантов исполнителем**

Рекомендации. 1. Внимательно изучите команды исполнителя.
2. Решение получается после перебора всех вариантов.
3. Просто подсчитываем количество нужных вариантов.

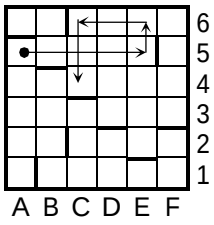
Задача 2008.А20

Ответ: 4) 4.

Решение. Необходимо проверить каждую клетку.
Нарисуем путь РОБОТа по программе для клетки A6.



Итак, начав с клетки A6, РОБОТ по данной программе возвращается в исходную клетку A6. Теперь проверим клетку A5.



Итак, начав с клетки A5, РОБОТ по данной программе приходит в клетку C4. Можно продолжать проверять все клетки, получится правильное решение. А можно заметить, что РОБОТ останавливается в той клетке, которая имеет стенку снизу. Поэтому достаточно проверить клетки A1, B1, C1, D1, E1, F1, E2, D3, F3, C4, B5, A6. Получим, что поставленному условию удовлетворяют клетки A1, C1, C4, A6, всего 4 клетки.

Задача 2009.А18

Ответ: 1) 1.

Решение. Полностью аналогично предыдущей задаче.

§ 2. Выполнение алгоритмов

1. Теория

1°. Символьная и строковая константы

Некоторые данные известны сразу на стадии разработки алгоритма. Такие данные называются *константами*. Приведем также список основных типов констант, которых достаточно для полноценного программирования.

Константа — фиксированное данное, не меняющееся при выполнении программы.

Основные типы констант — это четыре типа констант:

1) текстовые константы:

1а) символьные константы;

1б) строковые константы;

2) числовые константы

2а) целые константы;

2б) вещественные константы.

Отметим важную особенность алгоритмических языков, в том числе *Паскаля* и *Бейсика*. Компиляторы этих языков воспринимают только текст, записанный *аски-кодами*, т. е. символами американской клавиатуры (на русских компьютерах для набора в аски-кодах достаточно перейти на английский язык).

Числовые константы уже были рассмотрены в § 3 главы 1. Осталось изучить символьные константы.

Символьная константа — один байт кода, интерпретируемый как символ кодовой таблицы.

Иногда говорят, что *символьная константа* — на Паскале это любой символ, заключенный в апострофы, а на Бейсике символ, заключенный в кавычки (напоминаем, что апостроф и кавычки берутся из аски-кодов).

Рассмотрим, как на Паскале задаются символьные константы с помощью апострофов и символа, в них заключенного, а на Бейсике — с помощью кавычек и символа, в них заключенного.

Примеры на Паскале.

1. Звездочка '*'. Внутри апострофов — символ звездочки.

2. Пробел '␣'. Внутри апострофов — символ пробела.

3. Кавычки ''''. Внутри апострофов — символ кавычек.

4. Апостроф ''''''. Внутри апострофов находятся два символа апострофов подряд.

5. Пустой символ ''. Внутри апострофов ничего не находится.

Примеры на Бейсике.

1. Звездочка "*". Внутри кавычек — символ звездочки.

2. Пробел "␣". Внутри кавычек — символ пробела.

3. Апостроф """". Внутри кавычек — символ апострофа.

4. Кавычки """""". Внутри кавычек находятся два символа кавычек подряд.

5. Пустой символ "". Внутри кавычек ничего не находится.

Строковая константа — несколько последовательных байтов кода, интерпретируемых как символы кодовой таблицы.

Иногда говорят, что *строковая константа* — это последовательность символов, заключенная в апострофы на Паскале и в кавычки на Бейсике.

Строковые константы можно соединять знаком плюс.

Все изложенные факты языка программирования и те, которые еще будут изложены, подлежат обязательной проверке на компьютере по следующим причинам:

1) для практики и более полного понимания;

2) для проверки соответствия описания языка действительному положению вещей на данном компьютере.

Проверить функционирование символьных и строковых констант можно следующим образом. Например, на Паскале нужно убедиться, что при замене константы

```
'Привет,␣мир!'
```

на следующие эквивалентные ей *после выполнении программы* будет одинаковый вывод:

```
'Привет,' + ' мир!'
'Привет,' + '␣' + 'м' + 'и' + 'р' + '!'
```

Для проверки на Бейсике замените апострофы кавычами.

2°. **Имя**

Текст алгоритма или программы состоит из слов, знаков препинания и специальных символов.

Синтаксис алгоритмического языка — правила использования слов, знаков препинания и специальных символов при написании алгоритма на этом языке.

Слова языка программирования обычно делятся на два класса.

Служебное, или ключевое, или зарезервированное, слово — заранее определенное слово алгоритмического языка.

Имя — название элемента языка программирования, которое придумывает программист.

На Паскале и Бейсике в качестве имен нельзя использовать зарезервированные слова. Зарезервированными словами являются, например, такие, как end, for, to, if, else, sin, cos и другие служебные слова.

Здесь не приводится полный список зарезервированных слов Паскаля и Бейсика, поскольку не только начинающему многие из этих слов не нужны, но, возможно, они никогда не используются многими программистами. Обычно такие слова программист по интуиции не выбирает в качестве имен. Впрочем, если произойдет случайное попадание, транслятор об этом сообщит.

> Самый старый и приятный алгоритмический язык Фортран не имеет такой проблемы: не нужно зарезервированное слово — не надо, можно использовать его как имя. Нет необходимости искать полный список зарезервированных слов: если зарезервированное слово не используется по назначению, то никаких проблем при его использовании в качестве имени не возникнет.

Как и любое имя, имя в алгоритмическом языке несет как синтаксическую, так и семантическую нагрузку.

Символьный состав имени — символы, из которых состоит имя.

Содержание имени — объект программы, обозначаемый именем.

Общие правила формирования имени на алгоритмических языках следующие:

1) имя начинается с буквы или знака подчеркивания;

2) имя содержит только буквы, цифры и знак подчеркивания;

3) прописные и строчные буквы не различаются;

4) в качестве имени нельзя использовать зарезервированное слово;

5) имя не может содержать пробел;

6) имя должно быть уникально.

Рассмотрим два вопроса, связанные с созданием имени.

1. Длина и состав имени.

Рекомендуется использовать только осмысленные имена. Но при этом имя не рекомендуется делать слишком длинным.

В этой книге не будут использоваться имена длиннее 8 символов.

Вполне возможно использовать русские слова, записанные латинскими буквами.

Буквы из формул можно прямо переносить в программу в качестве имен переменных. Это также будет продемонстрировано в § 3.

Наконец, существует неофициальный стандарт, по которому в организации цикла участвуют объекты, имена которых состоят из одной буквы из следующего списка:

$$i, j, k, l, m, n.$$

2. Уникальность имени.

Мы будем изучать и создавать только достаточно короткие алгоритмы, в которых уникальность имени распространяется на весь алгоритм.

На Паскале чтобы описать содержание имени, нужно в специальном разделе алгоритма просто описать объекты, которые этими именами обозначаются.

На Бейсике заранее описывать имена не нужно. Имена определяются по мере их возникновения в алгоритме.

Очень важно, что алгоритмические языки не различают в именах прописные (заглавные) и строчные (обычные) буквы!

3°. Переменная и массив

Следующие по сложности объекты после констант — *переменные*. Без использования переменных алгоритм не имеет практического значения. Можно даже привести следующее определение алгоритма.

Алгоритм — последовательность действий, вычисляющих значения выходных переменных по значениям входных.

Переменная — именованное данное, которое может изменять свое значение в ходе выполнения программы. Переменная имеет фиксированное имя и фиксированный *тип*.

Тип — тип данного, который включает следующие составляющие:

1) множество значений данного;

2) форма кодирования значений данного в памяти компьютера;

3) допустимые операции над значениями данного.

Естественно, что переменные бывают точно таких же типов, как и константы: тестовые и числовые.

На Паскале переменные следует заранее описывать в специальном разделе описания переменных, а на Бейсике переменные определяются по мере появления в алгоритме.

Символьная переменная имеет значением один символ.

Целая переменная имеет значением одно целое число.

Вещественная переменная имеет значением одно вещественное число.

В алгоритмических языках существуют и другие типы целых и вещественных переменных, которые необходимо знать. Но они выходят за рамки книги.

Простая переменная — переменная, которую нельзя разбить на более простые.

Составная переменная — переменная, которая фактически составлена из простых переменных.

Адрес переменной — адрес ее первого байта в оперативной памяти.

Примерами простых переменных служат символьная, целая и вещественная переменные.

При выполнении программы простая переменная занимает один или несколько последовательных байтов оперативной памяти компьютера (см. рис. 1). Эти байты содержат двоичное число, сформированное в соответствии с типом переменной.

Рис. 1. Расположение переменных в оперативной памяти

Будем использовать только одну составную переменную — *массив*. Другие составные переменные не так необходимы.

Нас будут интересовать одномерные и двумерные массивы.

Массив — набор пронумерованных данных, имеющих общее имя. Эти данные можно рассматривать как простые переменные одинакового типа. Массив имеет фиксированное имя и фиксированный тип.

Элемент массива —элементарное данное массива.

Индексы элемента массива — координаты элемента массива в структуре массива.

Одномерный массив — массив, элементы которого определяются одним индексом (см. рис. 2). Для идентификации элемента *двумерного массива* нужно два индекса (см. рис. 3).

Длина, или размер, массива — количество элементов массива или, что то же самое, максимальный индекс массива в случае одномерного массива. В случае двумерного массива это произведение максимальных значений его двух индексов.

Тип массива — тип элементов массива. Все элементы массива имеют один и тот же тип.

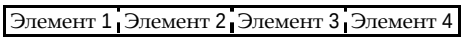


Рис. 2. Индексация одномерного массива длиной 4

Элемент 1,1	Элемент 1,2	Элемент 1,3	Элемент 1,4
Элемент 2,1	Элемент 2,2	Элемент 2,3	Элемент 2,4
Элемент 3,1	Элемент 3,2	Элемент 3,3	Элемент 3,4

Рис. 3. Индексация двумерного массива размером 4×3

4°. **Операции и выражения**

Над значениями данных, констант и переменных, можно выполнять разные *операции*.

Операция — вычисление из значений данных новых значений.

Операнды — данные, константы и переменные, которые входят в операцию.

Унарная, или одноместная, операция состоит из одного операнда.

Бинарная, или двуместная, операция включает два операнда.

Тернарная, или трехместная, операция имеет три операнда.

Выражение — запись операций. В выражение может входить одна или несколько операций.

Таким образом, результатом выполнения операции и выражения является значение.

Арифметические выражения и операции уже были рассмотрены в § 3 главы 1. Осталось изучить логические выражения и операции.

Рассмотрим простейшие логические выражения.

Мы не будем использовать ни логических констант, ни логических переменных. Зато будем пользоваться *логическими выражениями*.

Логическое выражение, или условие — выражение, которое может быть либо истинным, либо ложным.

Значениями логического выражения являются *истина* и *ложь*, которые на блок-схемах обозначают соответственно *да* и *нет*.

Также значения логического выражения обозначают на английском языке **истина** — true, **ложь** — false, или первыми буквами этих слов, или цифрами: **и**, или t, или 1, **л**, или f, или 0. Иногда на блок-схемах истину и ложь обозначают соответственно плюсом + и минусом –.

Простейшие логические выражения получаются, когда два арифметических выражения *в скобках* соединяются *операцией отношения*.

Операция отношения — операция над числами, имеющая логическое значение.

I. *Операции строгого сравнения.*

1. *Меньше* < — сравнивает два числа. Принимает значение истина, если первый операнд строго *меньше* второго, и ложь в противном случае. Например: 2 < 3 — истина, (2 * 3) < (5 + 1) — ложь.

2. *Больше* > — сравнивает два числа. Принимает значение истина, если первый операнд строго *больше* второго, и ложь в противном случае. Например: 2 > 3 — ложь, (2 * 3) > (5 + 1) — ложь.

II. *Операции нестрогого сравнения.*

1. *Меньше или равно* <= — сравнивает два числа. Принимает значение истина, если первый операнд *меньше или равен* второму, иначе ложь. Например: 2 <= 3 — истина, (2 * 3) <= (5 + 1) — истина.

2. *Больше или равно* >= — сравнивает два числа. Принимает значение истина, если первый операнд *больше или равен* второму, иначе ложь. Например: 2 >= 3 — ложь, (2 * 3) >= (5 + 1) — истина.

II. *Операции равенства.*

1. *Равно* = — сравнивает два числа. Истинно, если операнды *равны*, иначе ложно. Например: 2 = 3 — ложь, (2 * 3) = (5 + 1) — истина.

2. *Не равно* <> — сравнивает два числа. Истинно, если операнды *не равны*, иначе ложно. Например: 2 <> 3 — истина, (2 * 3) <> (5 + 1) — ложь.

Перечислены все шесть бинарных операций отношения, хотя наличие этих операций избыточно: они попарно имеют всегда противоположные значения истинности. А именно.

1. Значение выражения $a = b$ истинно тогда и только тогда, когда значение выражения $a <> b$ ложно.

Другими словами, если выражение $a = b$ ложно, то выражение $a <> b$ истинно. И наоборот: если $a <> b$ ложно, то $a = b$ истинно.

2. Значение выражения $a < b$ истинно тогда и только тогда, когда значение выражения $a >= b$ ложно.

3. Значение выражения *a* > *b* истинно тогда и только тогда, когда значение выражения *a* <= *b* ложно.

Наконец, рассмотрим полноценные логические операции над логическими выражениями.

Логическая операция — это операция над логическими выражениями, имеющая логическое значение. Будем использовать следующие логические операции.

I. *Унарная операция*.

1. *Отрицание* *not* — приписывание перед единственным операндом слова not. При этом значение истинности операнда меняется на противоположное.

Например: not ((2 * 3) < (5 + 1)) — истина.

II. *Бинарные операции*.

1. *И, или логическое «и», или* *and* равно истине тогда и только тогда, когда оба операнда истинны.

Например: (2 <> 3) and (2 < 3) — истина.

2. *Или, или логическое «или», или* *or* равно лжи тогда и только тогда, когда оба операнда ложны.

Например: (2 = 3) or (2 >= 3) — ложь.

Естественно, not по-английски означает «нет», and — «и» и or — «или».

С помощью этих трех операций, не особенно напрягаясь, можно записать логическое выражение любой сложности. (И даже с помощью двух операций: отрицания и любой из бинарных.) Однако рекомендуется программировать с использованием как можно более коротких логических выражений, разбивая длинные выражения на несколько коротких выражений.

Логические операции удобно записывать с помощью *таблиц истинности*. Также с их помощью удобно находить значения выражений, содержащих несколько логических операций.

Таблицы истинности подробно разбирались нами ранее в § 1 главы 2.

5°. **Оператор присваивания**. **Операторы ввода/вывода**

Значения переменных в ходе выполнения программы можно изменять двумя способами: с помощью *оператора присваивания* и с помощью *вызова функции*.

Присваивание — процесс изменения значения переменной.

Оператор — слово языка программирования, описывающее действие алгоритма. Может быть с параметрами или без.

Оператор присваивания — оператор, который явным образом изменяет значения переменной.

Оператор присваивания состоит из следующих частей:

1) *знака оператора присваивания, или присваивания* — на Паскале это двоеточие с равенством, а на Бейсике — только знак равенства. Знак присваивания разделяет два операнда;

2) точки с запятой, стоящей после второго операнда на Паскале, а на Бейсике — конца абзаца;

3) *первого операнда* — переменой, стоящей перед знаком оператора присваивания, значение которой меняет оператор;

4) *второго операнда* — выражения, стоящего после знака оператора присваивания, значение которого присваивается переменной, стояще слева.

Оператора присваивания отличается от равенства в математике тем, что одна и та же переменная может находиться с обеих сторон оператора присваивания. Например, увеличение значения переменной *a* на 1 записывается на Паскале

```
a := a + 1;
```

и на Бейсике

```
a = a + 1
```

При выполнении операторов присваивания из этого примера компьютер возьмет значение переменной *a*, прибавит к нему 1 и запишет новое значение в переменную *a*.

Этим свойством оператор присваивания обладает потому, что выполняется в два этапа:

1) сначала вычисляется значение выражения, стоящего *справа* от оператора присваивания;

2) после этого вычисленное значение присваивается переменной, стоящей *слева* от оператора присваивания.

Опишем способы задания входных данных алгоритма, другими словами, ввода значений *входных переменных* в алгоритм, т. е. способы присвоения переменным их первоначальных значения, которые затем будут обрабатываться алгоритмом.

Сделать это можно тремя способами:

1) оператором присваивания;

2) оператором ввода с клавиатуры;

3) оператором чтения данных из файла.

Ясно, как сделать это оператором присваивания: нужно в начале алгоритма присвоить переменным нужные значения.

Для ввода входных данных с клавиатуры используют *оператор ввода* — оператор, в котором указан список вводимых с клавиатуры переменных.

Например, ввести значения переменных *a* и *b* на языке Паскаль можно следующим оператором:

```
READLN (a, b);
```

а на языке Бейсик — следующим:

```
INPUT (a, b)
```

Здесь вводимые переменные отделены друг от друга запятой.

При выполнении компьютером операторов из этого примера сначала переменной *a*, а затем *b* будет присвоено значения двух чисел, набранных на клавиатуре. При наборе на клавиатуре между этими двумя числами следует набрать пробел, а после набора второго числа — нажать конец абзаца Enter.

Если нужно ввести еще входные данные, операторы ввода можно повторить.

Поскольку алгоритмические языки не различают в своих командах прописные и строчные буквы, в дальнейшем, как и в этом примере, будем для написания операторов использовать прописные буквы, а для переменных — строчные.

Как вводить входные данные, используя оператор чтения файла, в этом учебном пособии описывать не будем.

Опишем способы вывода вычисленных данных алгоритмом, другими словами, вывода значений как *выходных переменных*, так и *выходных констант*.

Сделать это можно двумя способами:

1) оператором вывода на экран монитора;

2) оператором записи данных в файл.

Для вывода выходных данных на экран монитора компьютера используют *оператор вывода* — оператор, в котором указан список выводимых на экран констант и переменных.

Например, вывести значения переменных *a* и *b* вместе с указанием того, что это именно переменные *a* и *b*, на языке Паскаль можно следующим оператором:

```
WRITELN ('a = ', a, 'b = ', b);
```

а на языке Бейсик — следующим:

```
PRINT "a = "; a; " b = "; b
```

В этих операторах выводимые тестовые константы и числовые переменные отделены друг от друга запятыми.

При выполнении компьютером операторов из этого примера будет выведена следующая строка. В начало строки с первой позиции будет выведена текстовая константа **a =** , причем знак равенства будет отбит с обеих сторон пробелами. Со следующей пятой позиции будет выведено число в десятичной системе, которое является значением переменной *a*.

Сразу после числа — значения *a* — будет выведена текстовая константа **b =** , причем знак равенства будет отбит с обеих сторон пробелами, а также один пробел будет выведен перед буквой **b**, чтобы отделить эту букву от предыдущего числа. Наконец, все в той же строке сразу после пробела после знака равенства будет выведено число в десятичной системе, которое является значением переменной *b*.

После выполнения оператора вывода компьютер перейдет на следующую строку на экране монитора. Если нужно вывести еще выходные данные, оператор вывода можно повторить, и тогда в следующей строке произойдет вывод указанных в операторе вывода констант и переменных.

Например, если $a = 1$, а $b = 2$, то эти операторы вывода выведут следующую строку:

```
a = 1 b = 2
```

Как выводить выходные данные, используя оператор записи файла, в этом учебном пособии описывать не будем.

6°. Структура следования. Блок-схема

Операторы присваивания и операторы ввода/вывода при выполнении компьютером алгоритма выполняются последовательно, в точности в том порядке, в каком они записаны в алгоритме.

Последовательное, линейное выполнение операторов называется *структурой следования*, или просто *следованием*, алгоритма.

А что такое структура алгоритма?

Структура алгоритма — схема изменения последовательности шагов алгоритма при условных и безусловных переходах.

Структурный элемент алгоритма, или структура алгоритма — участок алгоритма, который обладает следующими свойствами:

1) никакой шаг структуры не передает управление на шаг алгоритма, находящийся вне этой структуры;

2) никакой шаг структуры не принимает управление от шага алгоритма, находящегося вне этой структуры.

Другими словами, *структурный элемент алгоритма, или структура алгоритма* — участок алгоритма, который имеет только одну точку входа на свой первый шаг и одну точку выхода со своего последнего шага.

Приведем пример структуры следования, которая вводит значения входных переменных a и b, увеличивает их на 1 и затем выводит их значения. Запишем этот алгоритм не только на алгоритмических языках Паскаль и Бейсик, но и на языке блок-схем.

Сначала определим понятие блок-схемы и опишем некоторые ее элементы.

> Слава богу, представление о том, что блок-схемы устарели, устарело.

Линейное, или одномерное, представление алгоритма — запись шагов алгоритма в виде текстового списка.

Плоское, или двумерное, представление алгоритма, или блок-схема, или диаграмма — запись шагов алгоритма в виде плоских графических элементов, причем последовательность шагов и шаги безусловных переходов записываются стрелками.

Приведем элемент следования блок-схемы: *действие*.

Следование — элемент блок-схемы в виде прямоугольника или параллелограмма, представляющий структуру следования алгоритма. Имеет один вход и один выход (рис. 4). Если следование содержит только операторы присваивания, то рисуется прямоугольник, а если ввод или вывод — параллелограмм.

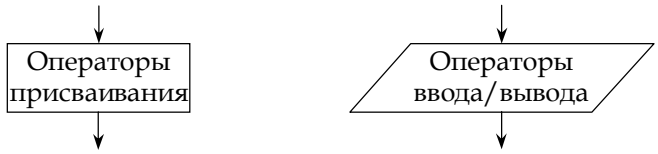


Рис. 4. Структура блок-схемы следование с операторами присваивания (слева) и операторами ввода/вывода (справа)

Теперь приведем пример структуры следования, какой мы хотели, в таблице 5.

Таблица 5

Пример структуры следования

Паскаль	Бейсик	Блок-схема
`a := 1;` `b := 2;`	`a = 1` `b = 2`	Ввод *a* и *b*
`a := a + 1;` `b := b + 1;`	`a = a + 1` `b = b + 1`	$a = a + 1$ $b = b + 1$

`WRITELN`
`('a = ', a,`
`'b = ', b);`
	`PRINT`
	`"a = "; a;`
	`"b = "; b`
		Вывод
		a и *b*

Таблица 5 составлена так, что операторы присваивания и операторы ввода/вывода, записанные на языке Паскаля, Бейсика и блок-схемы находятся, для сравнения, на одной строке.

В таблице 5 ввод данных на Паскале и Бейсике осуществлен операторами присваивания. Приведем пример ввода данных операторами ввода таблице 6.

Таблица 6

Пример структуры следования

Паскаль	Бейсик	Блок-схема
`READLN (a, b);`	`INPUT (a, b)`	Ввод *a* и *b*
`a := a + 1;` `b := b + 1;`	`a = a + 1` `b = b + 1`	$a = a + 1$ $b = b + 1$

`WRITELN`
`('a = ', a,`
`'b = ', b);`
	`PRINT`
	`"a = "; a;`
	`"b = "; b`
		Вывод *a* и *b*

В таблицах 5 и 6 операторы вывода занимают три строки по причине нехватки места. Их следует писать в одной строке!

Сделаем следующий вывод. Мы выяснили, что блок-схема удобней для записи алгоритмов по двум причинам:

1) арифметические выражения можно записывать в удобной математической форме;

2) операторы присваивания можно записывать в любой удобной форме, например, не ставить двоеточий и точек с запятой;

3) операторы ввода/вывода можно не детализировать для конкретных алгоритмических языков программирования, а записывать просто «ввод» со списком переменных ввода и «вывод» со списком переменных вывода. Всякие красоты и удобства читаемости оформляются потом при реализации алгоритма на конкретном языке программирования.

7°. **Структура цикла**. **Тестирование**. **Блок**

Смысл цикла заключается в том, что операторы, входящее в его состав, повторяются указанное число раз.

Цикл, или структура цикла, или повторения — структура алгоритма, которая организует повторение шагов от нуля до некоторого количества раз в зависимости от условий.

Изучение записи цикла начнем с языка блок-схем.

Элементарная блок-схема цикла — блок-схема элементарной структуры цикла. Операторы цикла повторяется то тех пор, пока выполняется условие цикла.

Обычно циклы используются для суммирования чисел. При этом приходится использовать вспомогательную переменную — *сумматор*, в котором накапливается вычисляемая сумма. Кроме того, используется еще одна вспомогательная переменная — *переменная цикла*.

Вспомогательная переменная — переменная, которой может не быть при описании алгоритма на обычном или математическом языке, но которая необходима для записи алгоритма на алгоритмическом языке.

Сумматор — вспомогательная переменная, сначала равная 0, к которой каждый раз при прохождении цикла прибавляется требуемое выражение.

Переменная цикла — вспомогательная переменная, каждый раз при проходе цикла меняющее свое значение на новое (обычно к переменной цикла прибавляется 1).

Переменная цикла служит для подсчета количества итераций цикла и для определения момента окончания цикла.

Например, пусть требуется просуммировать все целые числа от 1 до 3. Введем вспомогательную переменную i, пробегающую в цикле значения от 1 до 3 и обеспечивающую суммирование чисел в алгоритмическом языке (рис. 7).

Как видно из алгоритма на рисунке 7, переменную цикла (в данном случае i) можно использовать при вычислении выражений в цикле! Обратите внимание на то, что внутри цикла сначала происходит суммирование в сумматор, и только потом увеличение переменной цикла!

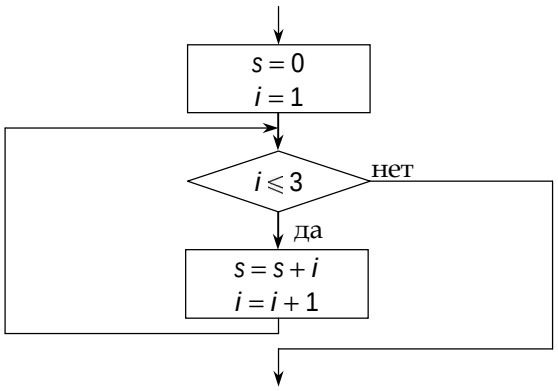


Рис. 7. Алгоритм суммирования всех целых чисел от 1 до 4

Возникает вопрос, выполняет ли построенный алгоритм то задание, которое от него требуется. Для выяснения этого вопроса проводят *тестирование алгоритма*.

Тестирование, или трассировка алгоритма,— пошаговое выполнение алгоритма.

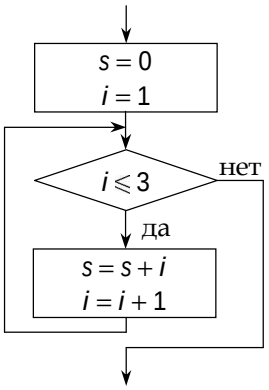
Протестируем алгоритм на рисунке 7. Другими словами, пошагово его пройдем. Если после окончания алгоритма мы получим $s = 6$, то алгоритм работает правильно.

$s = 0$.
$i = 1$.
$i \leqslant 3$? Да. (Здесь i равно 1.)
$s = s + i = 0 + 1 = 1$.
$i = i + 1 = 1 + 1 = 2$.
$i \leqslant 3$? Да. (Здесь i равно 2.)
$s = s + i = 1 + 2 = 3$.
$i = i + 1 = 2 + 1 = 3$.
$i \leqslant 3$? Да. (Здесь i равно 3.)
$s = s + i = 3 + 3 = 6$.
$i = i + 1 = 3 + 1 = 4$.
$i \leqslant 3$? Нет. (Здесь i равно 4.)
Итак, $s = 6$ и алгоритм работает правильно.

Запишем тот же самый алгоритм на Паскале и Бейсике, как показано в таблице 8.

Таблица 8

Пример структуры цикла

Паскаль	Бейсик	Блок-схема
`s := 0;`	`s = 0`	
`FOR i:=1 TO 3 DO BEGIN`	`FOR i=1 TO 3`	
`s := s + i;`	`s = s + i`	
`END;`	`NEXT i`	

После завершения цикла начинает выполняться следующий по порядку записи оператор.

Изучая таблицу 8 и сравнивая запись алгоритма на языке блок-схем и языках Паскаль и Бейсик, можно легко понять, как устроен оператор цикла в языках Паскаль и Бейсик.

На языках Паскаль и Бейсик цикл начинается с оператора FOR, в котором указываются начальное и конечное значение переменной цикла *i*.

На языке Бейсик цикл заканчивается оператором NEXT, который ограничивает операторы внутри цикла и указывает, что переменная цикла *i* увеличивается на 1 после выполнения операторов внутри цикла. Операторы FOR и NEXT составляют на языке Бейсик так называемый *блок*, внутри которого содержатся выполняемые операторы цикла.

Блок — структура следования, т. е. операторы присваивания и ввода/вывода, заключенные в специальные операторы.

На языке Паскаль *блок* всегда заключен во вспомогательные операторы BEGIN — *начало блока* и END — *конец блока*.

На языке Паскаль внутренние операторы цикла — это те операторы, которые находятся в блоке BEGIN—END сразу после оператора FOR. Точка с запятой ставится только в конце оператора цикла и после каждого оператора внутри блока, а концом оператора цикла считается конец блока цикла.

Обратите внимание, что на языках Паскаль и Бейсик переменная цикла увеличивается *после того*, как выполняется внутренние операторы цикла! На языке Бейсик это отражено в записи цикла в операторе NEXT. То, что это действительно так и на языке Паскаль. можно проверить только на практике, написав специальный тест на компьютере. Такая проверка оставляется читателю.

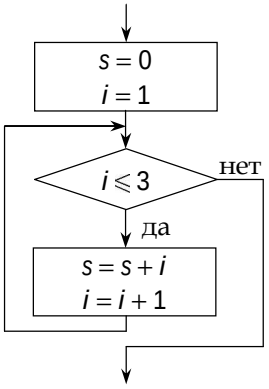
Операторы внутри любого блока следует сдвигать на несколько позиций вправо, что значительно улучшает читабельность текста, а значит, и его понимание.

Тестирование алгоритма, записанного на языках Паскаль и Бейсик, проводится точно так же, как и на языке блок-схем.

На языке Паскаль блок цикла можно опустить в том случае, если в блоке находится один единственный оператор. Перепишем алгоритм на Паскале без блока цикла, как показано в таблице 9.

Таблица 9

Пример структуры цикла

Паскаль	Бейсик	Блок-схема
`s := 0;`	`s = 0`	
`FOR i:=1 TO 3 DO`	`FOR i=1 TO 3`	
`  s := s + i;`	`  s = s + i`	
	`NEXT i`	

8°. **Структура выбора**

Структура цикла относится к *нелинейным структурам*.

Линейная структура алгоритма — структура, не меняющая последовательное выполнение записанных операторов.

Нелинейная структура алгоритма изменяет последовательность выполнения записанных операторов.

Речь в этих определениях идет, конечно, о записанных операторах. При выполнении любого алгоритма компьютером все операторы выполняются процессором последовательно, даже если в записи алгоритма есть непоследовательные структуры. (Параллельное выполнение алгоритмов на нескольких процессорах здесь не рассматривается.)

Структура цикла выполняет одни и те же записанные операторы по нескольку раз. Другими словами, структура цикла возвращает управление назад по тексту алгоритма.

Имеется еще одна нелинейная структура, которая пропускает выполнение некоторых операторы в тексте алгоритма. Это структура *выбора*.

Структура выбора, или ветвления, или альтернативы — последовательность выполнение операторов текста алгоритма, при которой, в зависимости от условия, некоторые операторы выполняются, а другие пропускаются, или просто некоторые операторы могут пропускаться.

Если происходит выбор среди операторов, то выбор называется *полным выбором* (см. рис. 10), если просто операторы могут быть пропущены, то *сокращенным выбором* (см. рис. 10).

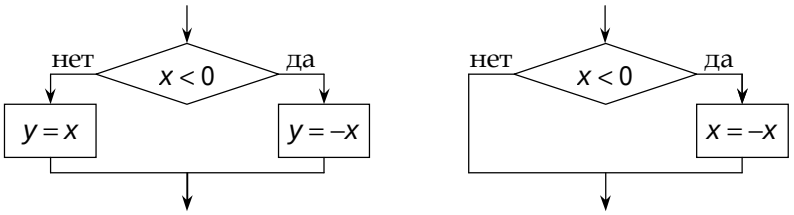


Рис. 10. Полный выбор (слева) и сокращенный (справа) на примере вычисления модуля числа

На рисунке 10 показаны два способа вычисления модуля числа $|x|$. В первом способе входная переменная x, а выходная — y, при этом используется полная структура выбора. Во втором способе входная и выходная переменная одна и та же — x, здесь удобно применить сокращенный выбор.

Как видно из рисунка 10, структура выбора имеет, как и положено, один вход и один выход.

Протестируем алгоритмы на рисунке 10. Если после окончания левого алгоритма мы получим $y = |x|$, а после окончания второго — $|x|$, то алгоритмы работают правильно.

Алгоритмы в принципе можно тестировать только на конкретных числах. На вход алгоритмов может поступить очень большое число разных чисел, но проверять все эти числа нереально. Обычно проводят минимальное тестирование.

Минимальное тестирование алгоритма — проведение по одному тесту для каждого класса входных данных.

Класс входных данных алгоритма — набор входных данных алгоритма, принципиально отличающихся от других данных.

При определении классов входных данных следует исходить только из типов входных данных без учета структуры алгоритма, т. е. алгоритм следует считать черным ящиком.

При нахождении модуля числа имеем 3 класса входных данных: 1) $x < 0$, 2) $x = 0$, 3) $x > 0$. Протестируем алгоритмы вычисления модуля числа (рис. 10) 3 раза: для $x = -1$, $x = 0$ и $x = 1$.

Полный выбор	Сокращенный выбор
$x = -1$.	$x = -1$.
$x < 0$? Да. (Здесь $x = -1$.)	$x < 0$? Да. (Здесь $x = -1$.)
$y = -x$.	$x = -x$.
$x = 0$.	$x = 0$.
$x < 0$? Нет. (Здесь $x = 0$.)	$x < 0$? Нет. (Здесь $x = 0$.)
$y = x$.	
$x = 1$.	$x = 1$.
$x < 0$? Нет. (Здесь $x = 1$.)	$x < 0$? Нет. (Здесь $x = 1$.)
$y = x$.	

Убедились, что алгоритмы работают правильно.

Запишем те же самые алгоритмы вычисления модуля числа с рисунка 10 на Паскале и Бейсике, как показано в таблицах 11 и 12.

Таблица 11

Пример структуры полного выбора

Паскаль	Бейсик	Блок-схема

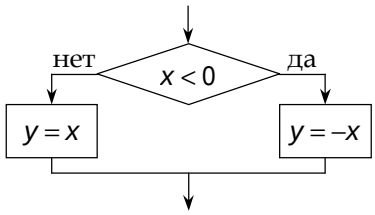
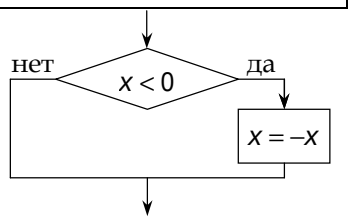
`IF (x<0) THEN`
`BEGIN`
`  y := -x;`
`END`
`ELSE`
`BEGIN`
`  y := x;`
`END;`
	`IF (x<0) THEN`
	`  y = -x`
	`ELSE`
	`  y = x`
	`ENDIF`
		

Таблица 12

Пример структуры сокращенного выбора

Паскаль	Бейсик	Блок-схема

`IF (x<0) THEN`
`BEGIN`
`  y := -x;`
`END;`
	`IF (x<0) THEN`
	`  y = -x`
	`ENDIF`
		

Изучая таблицы 11 и 12 и сравнивая запись алгоритма на языке блок-схем и языках Паскаль и Бейсик, можно легко понять, как устроен оператор выбора в языках Паскаль и Бейсик.

На языках Паскаль и Бейсик выбор начинается с оператора IF, после которого идет условие выбора и затем оператор THEN. После этого следуют операторы, которые выполняются, когда условие истинно. Если есть операторы, которые выполняются при ложном условии, то они располагаются после оператора ELSE (см. табл. 11).

При отсутствии выполняемых операторов для случая, когда условие выбора ложно, оператор ELSE можно опустить вместе с последующими операторами выбора как на Паскале, так и на Бейсике (см. табл. 12).

На языке Бейсик оператор выбора IF заканчивается оператором ENDIF. Операторы IF и ENDIF составляют на языке Бейсик так называемый *блок*, внутри которого содержатся выполняемые операторы выбора для обоих случаев значения условия, разделенные оператором ELSE (см. табл. 11).

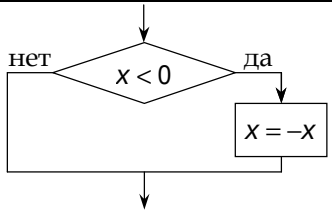
На языке Паскаль внутренние операторы выбора — это те операторы, которые находятся в блоке BEGIN—END сразу после оператора FOR для истинного значения условия выбора и в блоке BEGIN—END сразу после оператора ELSE для ложного значения условия. Точка с запятой ставится только в конце оператора выбора и после каждого оператора внутри обоих блоков, а концом оператора выбора считается конец второго блока (см. табл. 11).

Если на Паскале имеется только один выполняемый оператор в каком-нибудь блоке BEGIN—END, то ограничители этого блока BEGIN—END можно опустить (см. табл. 13).

Если на Бейсике имеется только один выполняемый оператор в выборе, относящийся к случаю, когда условие истинно, то оператор окончания выбора ENDIF можно опустить (см. табл. 13).

Таблица 13

Пример структуры сокращенного выбора в сокращенной записи

Паскаль	Бейсик	Блок-схема
`IF (x<0) THEN` `  y := -x;`	`IF (x<0) THEN` `  y = -x`	

9°. **Обработка массивов**

Структура цикла — самая мощная в компьютере. Именно она позволяет компьютеру выполнить очень много команд, выполняя алгоритм, который состоит всего из нескольких операторов.

Структура цикла позволяет обрабатывать очень много данных. Но где взять, как записать эти данные? Их можно записать в виде массива, в котором много данных имеют одно имя и различаются лишь индексом — номером расположения в массиве.

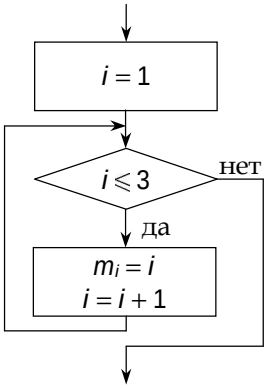
Разумеется, элементы массива обрабатываются с помощью цикла. При этом вспомогательная переменная цикла становится по совместительству вспомогательной переменной массива и пробегает все значения индекса массива.

Например, пусть требуется элементам массива m длины 3 присвоить последовательные числа 1, 2 и 3. Введем вспомогательную переменную i, пробегающую в цикле значения от 1 до 3 и обеспечивающую присваивание элементам массива нужных значений. Запишем этот алгоритм на языке блок-схем, а также на Паскале и Бейсике (см. табл.. 14).

Таблица 14

Пример присваивания значений элементам массива

Паскаль	Бейсик	Блок-схема

`FOR i:=1 TO 3 DO`
`BEGIN`
`  m[i] := i;`
`END;`
	`FOR i=1 TO 3`
	`  m(i) = i`
	`NEXT i`
		

Обратите внимание, что в алгоритме три переменные названы одним и тем же именем m и отличаются друг от друга только своим индексом. Причем индекс элемента массива указывается сразу справа от имени массива:

1) на языке блок-схем допускается указание индекса в любой форме, как Вам удобнее. В таблице 14 использован математическая форма в виде нижнего индекса, но Вы можете в блок-схемах обозначать индексы по-другому, например, как на Паскале или Бейсике;

2) на Паскале индекс записывается в квадратных скобках;

3) на Бейсике индекс записывается в круглых скобках.

Протестируем алгоритм из таблицы 14.

$i = 1$.

$i \leqslant 3$? Да. (Здесь i равно 1.)

$m(1) = 1$.

$i = i + 1 = 1 + 1 = 2$.

$i \leqslant 3$? Да. (Здесь i равно 2.)

$m(2) = 2$.

$i = i + 1 = 2 + 1 = 3$.

$i \leqslant 3$? Да. (Здесь i равно 3.)

$m(3) = 3$.

$i = i + 1 = 3 + 1 = 4$.

$i \leqslant 3$? Нет. (Здесь i равно 4.)

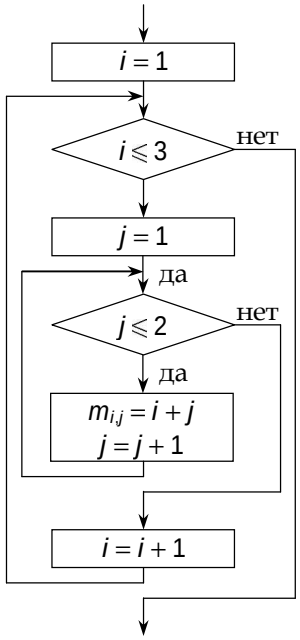
Итак, $m(1) = 1$, $m(2) = 2$, $m(3) = 3$, и алгоритм работает правильно.

Пусть теперь требуется элементам *двумерного* массива m размера 3×2 присвоить числа, равные сумме обоих их индексов. Введем вспомогательные переменные i и j, первая из которых пробегает в первом цикле значения от 1 до 3, а вторая пробегает во втором цикле значения от 1 до 2. Эти переменные обеспечат правильную индексацию элементов двумерного массива. Второй цикл будет находиться внутри первого цикла, а присвоение значений элементам массива — внутри второго цикла. Запишем этот алгоритм на языке блок-схем, а также на Паскале и Бейсике (см. табл. 14).

Таблица 15

Пример присваивания значений двумерному массиву

Паскаль	Бейсик	Блок-схема

`FOR i:=1 TO 3 DO`
`FOR j:=1 TO 2 DO`
`BEGIN`
`  m[i,j] := i + j;`
`END;`
	`FOR i=1 TO 3`
	`FOR j=1 TO 3`
	`  m(i,j) = i + j`
	`NEXT j`
	`NEXT i`
		

Протестируем алгоритм из таблицы 15.
$i = 1$.
$i \leqslant 3$? Да. (Здесь i равно 1.)

$j = 1$.
$j \leqslant 2$? Да. (Здесь j равно 1.)
$m(1, 1) = 1 + 1 = 2$.
$j = j + 1 = 1 + 1 = 2$.
$j \leqslant 2$? Да. (Здесь j равно 2.)
$m(1, 2) = 1 + 2 = 3$.
$j = j + 1 = 2 + 1 = 3$.
$j \leqslant 2$? Нет. (Здесь j равно 3.)

$i = i + 1 = 1 + 1 = 2$.
$i \leqslant 3$? Да. (Здесь i равно 2.)

$j = 1$.
$j \leqslant 2$? Да. (Здесь j равно 1.)
$m(2, 1) = 2 + 1 = 3$.
$j = j + 1 = 1 + 1 = 2$.
$j \leqslant 2$? Да. (Здесь j равно 2.)
$m(2, 2) = 2 + 2 = 4$.
$j = j + 1 = 2 + 1 = 3$.
$j \leqslant 2$? Нет. (Здесь j равно 3.)

$i = i + 1 = 2 + 1 = 3$.
$i \leqslant 3$? Да. (Здесь i равно 3.)

$j = 1$.
$j \leqslant 2$? Да. (Здесь j равно 1.)
$m(3, 1) = 3 + 1 = 4$.
$j = j + 1 = 1 + 1 = 2$.
$j \leqslant 2$? Да. (Здесь j равно 2.)
$m(3, 2) = 3 + 2 = 5$.
$j = j + 1 = 2 + 1 = 3$.
$j \leqslant 2$? Нет. (Здесь j равно 3.)

$i = i + 1 = 3 + 1 = 4$.
$i \leqslant 3$? Нет. (Здесь i равно 4.)
Итак,
$m(1, 1) = 2$, $m(1, 2) = 3$,
$m(2, 1) = 3$, $m(2, 2) = 4$,
$m(3, 1) = 4$, $m(3, 2) = 5$,
и алгоритм работает правильно.

2. Алгоритмы

Разумеется, в этом разделе находятся не те алгоритмы, которые исследуются в заданиях. Здесь приведены технологии решения заданий ЕГЭ, которые содержат алгоритмы.

Алгоритм 1. Выполнение алгоритма.

1. Выполнение алгоритма производится последовательно, по списку операторов, начиная с первого оператора.

2. В блок-схеме просто вычисляем логические условия и переходим по стрелкам. На языках программирования учитываем структуру операторов цикла и выбора.

3. Не забываем обновлять значения переменных, когда они меняются операторами присваивания.

Алгоритм 2. Тестирование.

1. Для поиска ошибки в алгоритме необходимо произвести его тестирование.

2. Для тестирования разбиваем входные данные на принципиально различные классы, например: больше нуля и меньше нуля, выше графика функции и ниже графика функции, принадлежащие разным областям на плоскости.

3. Чтобы найти одну ошибку, достаточно найти один комплект входных данных, на которых алгоритм работает неправильно.

3. Задачи

1°. **Оператор присваивания**

Задача 2006.A7

Определите значение целочисленных переменных a и b после выполнения фрагмента программы:

Бейсик	**Паскаль**	**Алгоритмический**

`a = 2468`
`b = (a MOD 1000)*10`
`a = a\1000 + b`
`'\ и MOD - операции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно`
	`a := 2468;`
	`b := (a mod 1000)*10;`
	`a := a div 1000 + b;`
	`{div и mod - операции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно}`
		`a := 2468`
		`b := mod(a, 1000)*10`
		`a := div(a, 1000) + b`
		`|div и mod - функции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно`

1) a = 22, b = 20
2) a = 4682, b = 4680
3) a = 8246, b = 246
4) a = 470, b = 468

Задача 2007.A7

Определите значение целочисленных переменных a и b после выполнения фрагмента программы:

Бейсик	**Паскаль**	**Алгоритмический**

`a = 1819`
`b = (a\100)*10 + 9`
`a = (10*b - a) MOD 100`
`'\ и MOD - операции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно`
	`a := 1819;`
	`b := (a div 100)*10 + 9;`
	`a := (10*b - a) mod 100;`
	`{div и mod - операции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно}`
		`a := 1819`
		`b := div(a,100)*10 + 9`
		`a := mod(10*b - a,100)`
		`|div и mod - функции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно|`

1) a = 81, b = 199
2) a = 81, b = 189
3) a = 71, b = 199
4) a = 71, b = 189

Задача 2008.А7

Определите значение целочисленных переменных a и b после выполнения фрагмента программы:

Бейсик	Паскаль	Алгоритмический

a = 3 + 8*4
b = (a \ 10) + 14
a = (b MOD 10) + 2
'\ и MOD - операции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно
	a:= 3 + 8*4;
	b:= (a div 10) + 14;
	a:= (b mod 10) + 2;
	{div и mod - операции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно}
		a:= 3 + 8*4
		b:= div(a,10) + 14
		a:= mod(b,10) + 2
		|div и mod - функции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно

1) a = 0, b = 18
2) a = 11, b = 19
3) a = 10, b = 18
4) a = 9, b = 17

Задача 2009.А5

Определите значение целочисленных переменной c после выполнения следующего фрагмента программы:

Бейсик	Паскаль	Алгоритмический

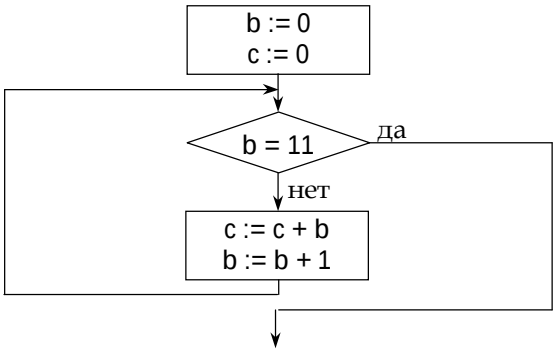
a = 5
a = a + 6
b = -a
c = a - 2*b
	a := 5;
	a := a + 6;
	b := -a;
	c := a - 2*b;
		a := 5
		a := a + 6
		b := -a
		c := a - 2*b

1) $c = -11$ 2) $c = 15$ 3) $c = 27$ 4) $c = 33$

2°. **Цикл**

Задача 2006.А6

Определите значение переменной **c** после выполнения фрагмента алгоритма:

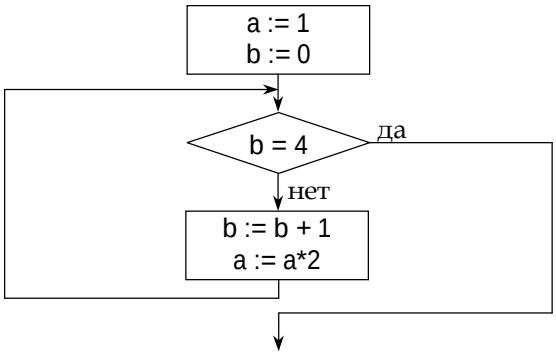


Примечание: знаком := обозначена операция присваивания.

1) 1 2) 45 3) 55 4) 66

Задача 2007.А6

Определите значение переменной a после выполнения фрагмента алгоритма:

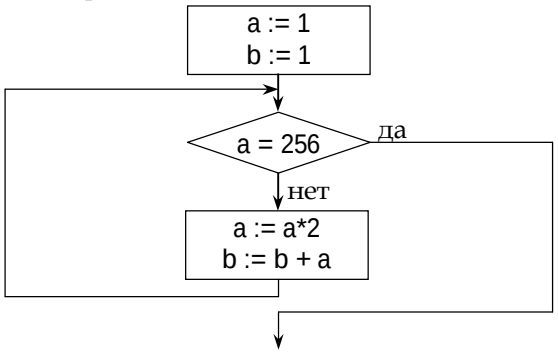


*Примечание: знаком * обозначено умножение, знаком := обозначена операция присваивания.*

1) 8 2) 16 3) 32 4) 12

Задача 2009.B2

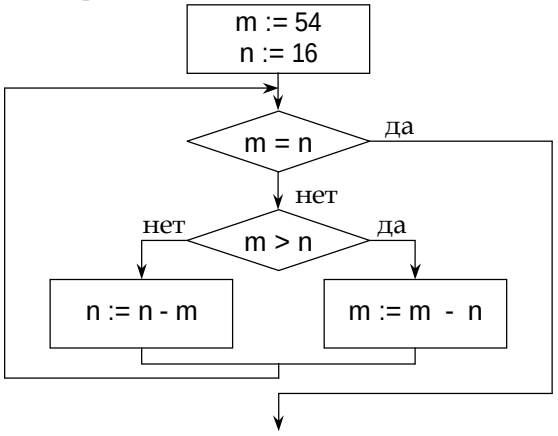
Запишите значение переменной b после выполнения фрагмента алгоритма:



*Примечание: знаком := обозначена операция присваивания, знаком * обозначена операция умножения.*

Задача 2008.A6

Определите значение переменной m после выполнения фрагмента алгоритма:



Примечание: знаком := обозначена операция присваивания.

1) 1 2) 2 3) 6 4) 16

Задача 2007.А20

В приведенном ниже фрагменте алгоритма, записанном на алгоритмическом языке, переменные *a, b, c* имеют тип «строка», а переменные *i, k* — тип «целое». Используются следующие функции.

Длина (*a*) — возвращает количество символов в строке *a*. (Тип «целое»).

Извлечь (*a, i*) — возвращает *i*-й (слева) символ в строке *a*. (Тип «строка»).

Склеить (*a, b*) — возвращает строку, в которой записаны сначала все символы строки *a*, а затем все символы строки *b*. (Тип «строка»).

Значения строк записываются в одинарных кавычках (Например, *a* := ‘дом’).

Фрагмент алгоритма:

```
i := Длина (a)
k := 2
b := ‘А’
пока i > 0
нц
c := Извлечь (a, i)
b := Склеить (b, c)
i := i – k
кц
b := Склеить (b, ‘Т’)
```

Какое значение будет у переменной *b* после выполнения вышеприведенного фрагмента алгоритма, если значение переменной *a* было ‘ПОЕЗД’?

1) ‘АДЕПТ’ 2) ‘АДЗЕОП’ 3) ‘АДТЕТПТ’ 4) ‘АДЗОТ’

3°. **Массив**

Задача 2006.А8

Значения двумерного массива размера 7×7 задаются с помощью вложенного оператора цикла в представленном фрагменте программы.

Бейсик	**Паскаль**	**Алгоритмический**

`FOR n=1 TO 7`
`FOR k=1 TO 7`
`  B(n,k) = k - n`
`NEXT k`
`NEXT n`
	`for n:=1 to 7 do`
	`for k:=1 to 7 do`
	`  B[n,k] := k - n;`
		`нц для n от 1 до 7`
		`нц для k от 1 до 7`
		`  B[n,k] = k - n`
		`кц`
		`кц`

Сколько элементов массива будут иметь положительные значения?

1) 49 2) 28 3) 21 4) 7

Задача 2007.А8

Значения двух массивов A[1..100] и B[1..100] задаются с помощью следующего фрагмента программы.

Бейсик	**Паскаль**	**Алгоритмический**

`FOR n=1 TO 100`
`  A(n) = n - 10`
`NEXT n`
`FOR n=1 TO 100`
`  B(n) = A(n)*n`
`NEXT n`
	`for n:=1 to 100 do`
	`  A[n] := n - 10;`
	`for n:=1 to 100 do`
	`  B[n] := A[n]*n;`
		`нц для n от 1 до 100`
		`  A[n] = n - 10`
		`кц`
		`нц для n от 1 до 100`
		`  B[n] = A[n]*n`
		`кц`

Сколько элементов массива B будут иметь положительные значения?

1) 10 2) 50 3) 90 4) 100

Задача 2008.A8

Значения двух массивов A[1..100] и B[1..100] задаются с помощью следующего фрагмента программы.

Бейсик	Паскаль	Алгоритмический

FOR n=1 TO 100
  A(n) = (n-80)*(n-80)
NEXT n
FOR n=1 TO 100
  B(101-n) = A(n)
NEXT n
	for n:=1 to 100 do
	  A[n] := (n-80)*(n-80);
	for n:=1 to 100 do
	  B[101-n] := A[n];
		нц для n от 1 до 100
		  A[n]=(n-80)*(n-80)
		кц
		нц для n от 1 до 100
		  B[101-n] = A[n]
		кц

Какой элемент массива B будет наибольшим?

1) B[1] 2) B[21] 3) B[80] 4) B[100]

Задача 2009.A6

Дан фрагмент программы, обрабатывающей двухмерный массив A размера $n \times n$.

Бейсик	Паскаль	Алгоритмический

k = 1
FOR i=1 TO n
  c = A(i,i)
  A(i,i) = A(k,i)
  A(k,i) = c
NEXT i
	k := 1;
	for i:=1 to n do
	begin
	  c := A[i,i];
	  A[i,i] := A[k,i];
	  A[k,i] := c;
	end;
		k := 1
		нц для i от 1 до n
		  c := A[i,i]
		  A[i,i] := A[k,i]
		  A[k,i] := c
		кц

Представим массив в виде квадратной таблицы, в которой для элемента массива A[i,j] величина i является номером строки, а величина j — номером столбца, в котором расположен элемент. Тогда данный алгоритм меняет местами:

1) два столбца в таблице;
2) две строки в таблице;
3) элементы диагонали и *k*-й строки таблицы;
4) элементы диагонали и *k*-го столбца таблицы.

4°. **Формирование строки**

Задача 2006.B6

Цепочки символов (строки) создаются по следующему правилу.

Первая строка состоит из одного символа — цифры «1».

Каждая из последующих цепочек создается такими действиями: в очередную строку дважды записывается цепочка цифр из предыдущей строки (одна за другой, подряд), а в конец приписывается еще одно число – номер строки по порядку (на *i*-м шаге дописывается число «*i*»).

Вот первые 4 строки, созданные по этому правилу:

(1) 1
(2) 112
(3) 1121123
(4) 112112311211234

Какая цифра стоит в седьмой строке на 120-м месте (считая слева направо)?

Задача 2007.B6

Цепочки символов (строки) создаются по следующему правилу.

Первая строка состоит из одного символа — цифры «1».

Каждая из последующих цепочек создается следующим действием:

в очередную строку дважды записывается предыдущая цепочка цифр (одна за другой, подряд), а в конец приписывается еще одно число — номер строки по порядку (на *i*-м шаге дописывается число «*i*»).

Вот первые 4 строки, созданные по этому правилу:

(1) 1
(2) 112
(3) 1121123
(4) 112112311211234

Сколько раз в общей сложности встречаются в восьмой строке четные цифры (2, 4, 6, 8)?

Задача 2008.B6

Цепочки символов (строки) создаются по следующему правилу.

Первая строка состоит из одного символа — цифры «1».

Каждая из последующих цепочек создается такими действиями:

в начало записывается число — номер строки по порядку (для *i*-й строки ставится число «*i*»), далее дважды подряд записывается предыдущая строка.

Вот первые 4 строки, созданные по этому правилу:

(1) 1
(2) 211
(3) 3211211
(4) 432112113211211

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

Задача 2009.B8

Строки (цепочки символов латинских букв) создаются по следующему правилу.

Первая строка состоит из одного символа — латинской буквы «A». Каждая из последующих цепочек создается такими действиями:

в очередную строку сначала записывается буква, чей порядковый номер в алфавите соответствует номеру строки (на *i*-м шаге пишется «*i*»-я буква алфавита), к ней справа дважды подряд приписывается предыдущая строка.

Вот первые 4 строки, созданные по этому правилу:

(1) A
(2) BAA
(3) CBAABAA
(4) DCBAABAACBAABAA

Латинский алфавит (для справки):

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Запишите семь символов подряд, стоящие в восьмой строке со 126-го по 132-е место (считая слева направо).

5°. **Поиск ошибки в программе**

Задача 2006.С1

Требовалось написать программу, в которой нужно было проверить, лежит ли число x на числовой оси между числами a и b («между» понимается в строгом смысле, т. е. случай $x = a$ или $x = b$ недопустим). Числа x, a, b являются натуральными, и известно, что a отлично от b (но неизвестно: $a > b$ или $b > a$). Входная информация вводится с клавиатуры, а на выходе должно быть сообщение вида «x между a и b» (если это действительно так), в противном случае никакой выходной информации не выдается.

Программист торопился и написал программу некорректно.

Программа на Паскале	Программа на Бейсике

`VAR a,b,x: INTEGER;`
`  p: INTEGER;`
`BEGIN`
`  READLN(a,b,x);`
`  IF (a>x) AND (x>b) THEN`
`  WRITELN('x между a,b');`
`END.`
	`CLS`
	`INPUT a, b, x`
	`IF (a>x) AND (x>b) THEN`
	`  PRINT "x между a, b"`
	`END`

Последовательно выполните три задания:

1) Приведите пример таких чисел a, b, x, при которых программа работает неправильно.

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

3) Укажите, как можно доработать программу, соблюдая дополнительное условие: доработанная программа не должна использовать логических операций AND или OR.

Опишите на русском языке или на одном из языков программирования алгоритм поиска второго по величине (т. е. следующего по величине за максимальным) элемента в числовом массиве из 30 различных элементов.

Задача 2007.С1

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

Программа на Паскале	Программа на Бейсике	Программа на Си

`VAR a, b, x: REAL;`
`BEGIN`
`READLN(a,b,x);`
`IF b = 0 THEN`
`WRITE('x = 0')`
`ELSE`
`IF a = 0 THEN`
`WRITE('нет решений')`
`ELSE`
`WRITE('x =',-b/a);`
`END.`
	`INPUT a, b, x`
	`IF b = 0 THEN`
	`PRINT "x = 0"`
	`ELSE`
	`IF a = 0 THEN`
	`PRINT "нет решений"`
	`ELSE`
	`PRINT "x=",-b/a`
	`ENDIF`
	`ENDIF`
	`END`
		`void main(void)`
		`{ float a,b,x;`
		`scanf("%f%f%f",`
		`&a,&b,&x);`
		`if (b==0)`
		`printf("x=0");`
		`else`
		`if (a==0)`
		`printf("нет решений");`
		`else`
		`printf("x=%f",-b/a);`
		`}`

Последовательно выполните три задания:

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

2) Укажите, какая часть программы является лишней.

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

Задача 2008.С1

Требовалось написать программу, которая решает уравнение «$a|x| = b$» относительно x для любых чисел a и b, введенных с клавиатуры. Все числа считаются действительными. Программист торопился и написал программу неправильно.

Программа на Паскале	Программа на Бейсике	Программа на Си

`VAR a, b, x: REAL;`
`BEGIN`
`READLN(a,b,x);`
`IF a = 0 THEN`
`IF b = 0 THEN`
`WRITE('любое число')`
`ELSE`
`WRITE('нет решений')`
`ELSE`
`IF b = 0 THEN`
`WRITE('x = 0')`
`ELSE`
`WRITE('x =',b/a,`
`' или x =',-b/a);`
`END.`
	`INPUT a, b, x`
	`IF a = 0 THEN`
	`IF b = 0 THEN`
	`PRINT "любое число"`
	`ELSE`
	`PRINT "нет решений"`
	`ENDIF`
	`ELSE`
	`IF b = 0 THEN`
	`PRINT "x = 0"`
	`ELSE`
	`PRINT "x =",b/a,`
	`" или x =",-b/a`
	`ENDIF`
	`ENDIF`
	`END`
		`void main(void)`
		`{ float a,b,x;`
		`scanf("%f%f%f",`
		`&a,&b,&x);`
		`if (a==0)`
		`if (b==0)`
		`printf("любое число");`
		`else`
		`printf("нет решений");`
		`else`
		`if (b==0)`
		`printf("x=0");`
		`else`
		`printf("x=%f или`
		`x=%f", b/a,-b/a);`
		`}`

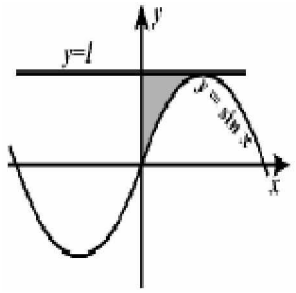
Последовательно выполните три задания:

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

2) Укажите, какая часть программы является лишней.

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

Задача 2009.C1



Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (*x*, *y* — действительные числа) и определяет принадлежность точки заштрихованной области, включая ее границы. Программист торопился и написал программу неправильно.

Программа на Паскале	Программа на Бейсике	Программа на Си

VAR x, y: REAL;
BEGIN
READLN(x,y);
IF y<=1 THEN
IF x>=0 THEN
IF y>=SIN(x) THEN
WRITE('принадлежит')
ELSE
WRITE('не принадлежит)
end.
	INPUT x, y
	IF y<=1 THEN
	IF x>=0 THEN
	IF y>=SIN(x) THEN
	PRINT "принадлежит"
	ELSE
	PRINT "не принадлежит"
	ENDIF
	ENDIF
	ENDIF
	END
		void main(void)
		{ float x,y;
		scanf("%f%f", &x,&y);
		if (y<=1)
		if (x>=0)
		if (y>=sin(x))
		printf("принадлежит");
		else
		printf("не принадлежит");
		}

Последовательно выполните следующее:

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

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

4. Ответы

1°. **Оператор присваивания**

Задача 2006.А7. 2) a = 4682, b = 4680.
Задача 2007.А7. 4) a = 71, b = 189.
Задача 2008.А7. 4) a = 9, b = 17.
Задача 2009.А5. 4) c = 33.

2°. **Цикл**

Задача 2006.А6. 3) 55.
Задача 2007.А6. 2) 16.
Задача 2009.В2. 511.
Задача 2008.А6. 2) 2.
Задача 2007.А20. 1) ‘АДЕПТ’.

3°. **Массив**

Задача 2006.А8. 3) 21.
Задача 2007.А8. 3) 90.
Задача 2008.А8. 4) B[100].
Задача 2009.А6. 3) элементы диагонали и *k*-й строки таблицы.

4°. **Формирование строки**

Задача 2006.В6. 1.
Задача 2007.В6. 85.
Задача 2008.В6. 127.
Задача 2009.В8. BAAGFED.

5°. **Поиск ошибки в программе**

Задача 2006.С1. $a = 1$, $x = 2$, $b = 3$.

Задача 2007.С1. $a = 0$, $b = 0$, $x = 0$ (значение x можно не указывать, допустим ответ, что x — любое число).

Задача 2008.С1. $a = 1$, $b = -1$, $x = 0$.

Задача 2009.С1. $x = 3$, $y = 0{,}5$ (любая пара (x, y), для которой выполняется: $y > 1$ или $x < 0$ или $(y \geqslant \sin x$ и $x > \pi/2$ и $y \leqslant 1)$).

5. Решения

1°. Оператор присваивания

Рекомендации. 1. Изучаем текст алгоритма на том языке, который знаком, а можно и сразу на всех.

2. Последовательно выполняем операторы присваивания.

3. Ответ получается автоматически.

Задача 2006.A7

Ответ: 2) a = 4682, b = 4680.

Решение. Просто вычислим b и a.

$b = (a \bmod 1000) \times 10 = (2468 \bmod 1000) \times 10 = 468 \times 10 = 4680.$

$a = a / 1000 + b = 2468 / 1000 + 4680 = 2 + 4680 = 4682.$

Задача 2007.A7

Ответ: 4) a = 71, b = 189.

Решение. Полностью аналогично решению предыдущей задачи.

Задача 2008.A7

Ответ: 4) a = 9, b = 17.

Решение. Полностью аналогично решению предыдущей задачи.

Задача 2009.A5

Ответ: 4) c = 33.

Решение. Полностью аналогично решению предыдущей задачи.

2°. Цикл

Рекомендации. 1. Изучаем алгоритма в виде блок-схемы.

2. Последовательно выполняем операторы и перемещаемся по стрелкам в соответствии с условиями.

3. Ответ получается автоматически.

Задача 2006.А6

Ответ: 3) 55.

Решение 1. Пройдем по алгоритму до конца и проследим за значениями переменных.

Начало алгоритма.
$b = 0$.
$c = 0$.

$b = 11$? Нет. (Здесь b равно 0.)
$c = c + b = 0 + 0 = 0$.
$b = b + 1 = 0 + 1 = 1$.

$b = 11$? Нет. (Здесь b равно 1.)
$c = c + b = 0 + 1 = 1$.
$b = b + 1 = 1 + 1 = 2$.

$b = 11$? Нет. (Здесь b равно 2.)
$c = c + b = 1 + 2 = 3$.
$b = b + 1 = 2 + 1 = 3$.

И т. д.

$b = 11$? Нет. (Здесь b равно 10.)
$c = c + b = 45 + 10 = 55$.
$b = b + 1 = 10 + 1 = 11$.

$b = 11$? Да. (Здесь b равно 11.)
Конец алгоритма.

Решение 2. При внимательном изучении алгоритма становится ясно, что приведенный цикл одиннадцать раз прибавляет к b единицу, причем все b суммируются в сумматор c.

Поэтому в c накапливается сумма первых натуральных чисел от 0 до 10, которая равна 45.

Задача 2007.A6

Ответ: 2) 16.

Решение 1. Полностью аналогично решению 1 предыдущей задачи.

Решение 2. При внимательном изучении алгоритма становится ясно, что цикл выполняется 4 раза.

Каждый раз в цикле число a := 1 умножается на 2. Поэтому в умножителе a накапливается число $2^4 = 16$.

Задача 2009.B2

Ответ: 511.

Решение 1. Полностью аналогично решению 1 предыдущей задачи.

Решение 2. При внимательном изучении алгоритма становится ясно, что цикл выполняется 8 раз, поскольку $256 = 2^8$.

А в сумматоре b накапливается сумма степеней двойки от 1 до 8: $1 + 2 + 4 + \ldots + 2^8 = 2^9 - 1 = 511$.

Задача 2008.A6

Ответ: 2) 2.

Решение 1. Аналогично решению 1 предыдущей задачи.

Решение 2. При внимательном изучении алгоритма становится ясно, что это алгоритм Евклида, который вычисляет наибольший общий делитель чисел 54 и 16.

Общий делитель чисел 54 и 16 равен 2, поскольку 16 — это степень 2, а 54 на 4 не делится.

Задача 2007.A20

Ответ: 1) 'АДЕПТ'.

Решение. Аналогично решению 1 предыдущей задачи.

3°. **Массив**

Рекомендации. 1. Изучаем текст алгоритма на том языке, который знаком, а можно и сразу на всех.

2. Выполняем оба цикла, пока они не закончатся.

3. Ответ получается автоматически.

Задача 2006.A8

Ответ: 3) 21.

Решение 1. Два вложенных оператора цикла FOR обеспечивают изменение обоих размерностей двумерного массива от 1 до 7, поэтому каждому элементу этого двумерного массива 7×7 при выполнении этой программы присваивается какое-нибудь значение.

Нарисуем двумерный массив в виде матрицы и заполним его числами, вычисленными по данному алгоритму.

n \ *k*	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6
2	–1	0	1	2	3	4	5
3	–2	–1	0	1	2	3	4
4	–3	–2	–1	0	1	2	3
5	–4	–3	–2	–1	0	1	2
6	–5	–4	–3	–2	–1	0	1
7	–6	–5	–4	–3	–2	–1	0

В этой таблице 21 положительное число.

Решение 2. При внимательном изучении алгоритма становится ясно, когда выражение $k - n$ положительно, если k и n пробегают все значения от 1 до 7.

При $k = 1$ таких значений нет.

При $k = 2$ имеем 1 значение при $n = 1$.

При $k = 3$ имеем 2 значения при $n = 1$ и $n = 2$.

И т. д.

При $k = 7$ имеем 6 значений при $n = 1$, $n = 2$, …, $n = 6$.

Всего получаем $1 + 2 + \ldots + 6 = 21$.

Задача 2007.А8

Ответ: 3) 90.

Решение 1. Аналогично решению 1 предыдущей задачи.

Решение 2. При внимательном изучении оператора алгоритма B(n) = A(n)*n становится ясно, что положительные значения будут иметь ровно столько же элементов массива B, сколько их будут иметь элементы массива A.

Выражение $n - 10$ при $n = 1, 2, \ldots, 100$ положительно, когда $n = 11, 12, \ldots, 100$. Получаем 90 положительных значений.

Задача 2008.А8

Ответ: 4) B[100].

Решение 1. Аналогично решению 1 предыдущей задачи.

Решение 2. При внимательном изучении оператора алгоритма B(101-n) = A(n) становится ясно, что массив B получается инвертированием массива A, т. е. первый элемент массива A — это последний B, второй A — предпоследний B и т. д.

В массив A записываются значения квадратичной функции x^2, где значения переменной x меняются от –70 при $n = 1$ до 20 при $n = 100$. Следовательно, максимальное значение будет у элемента массива A с номером 1.

Следовательно, максимальное значение будет у элемента массива B с номером 100.

Задача 2009.А6

Ответ: 3) элементы диагонали и *k*-й строки таблицы.

Решение 1. Аналогично решению 1 предыдущей задачи.

Решение 2. При внимательном изучении операторов алгоритма по обмену значений двух переменных A(i,i) и A(k,i) становится ясно следующее.

При изменении параметра цикла i от 1 до n переменная A(i,i) пробегает диагональ таблицы, а переменная A(k,i) —*k*-ю строку таблицы, как написано в условии задания.

4°. **Формирование строки**

Рекомендации. 1. Внимательно читаем текст задания.

2. Последовательно вычисляем нужные параметры строк. Находим закономерность этих вычислений.

3. Если ответ сразу не получится, то придется найти дополнительную закономерность.

Задача 2006.B6

Ответ: 1.

Решение. Сначала подсчитаем количество символов в 7-й строке. В 4-й строке 15 символов. В 5-й строке $15 + 15 + 1 = 31$ символ. В 6-й строке $31 + 31 + 1 = 63$ символа. В 7-й строке $63 + 63 + 1 = 127$ символов.

Последний символ строки совпадает с ее номером,— по построению. Предпоследний — с номером предыдущей строки,— по тому же построению. И т. д. Итак, последние 7 символов 7-й строки — это 7 символов 1234567.

Но 120-е место 7-й строки — это 8-е место с конца строки, т. е. место перед строкой 1234567. А в произвольной n-й строке (кроме первой) на $n + 1$ месте с конца всегда стоит 1.

Задача 2007.B6

Ответ: 85.

Решение. Подсчитаем количество четных цифр (2, 4, 6, 8) сначала в 1-й строке, затем во 2-й, 3-й и т. д. до 8-й. Это количество каждый раз удваивается, а если строка является четной, то после удвоения прибавляется 1.

В 1-й строке — 0 четных цифр.

Во 2-й строке — $2 \cdot 0 + 1 = 1$ четная цифра

В 3-й строке — $2 \cdot 1 = 2$ четные цифры.

В 4-й строке — $2 \cdot 2 + 1 = 5$ четных цифр.

В 5-й строке — $2 \cdot 5 = 10$ четных цифр.

В 6-й строке — $2 \cdot 10 + 1 = 21$ четная цифра.

В 7-й строке — $2 \cdot 21 = 42$ четные цифры.

В 8-й строке — $2 \cdot 42 + 1 = 85$ четных цифр.

Задача 2008.B6

Ответ: 127.

Решение. Аналогично решению предыдущей задачи.

Задача 2009.B8

Ответ: BAAGFED.

Решение. Аналогично решению задачи 2006.B6.

5°. Поиск ошибки в программе

Рекомендации. 1. Изучаем текст алгоритма на том языке, который знаком, а можно и сразу на всех. Лучше всего — на Бейсике, в заданиях алгоритмы на Бейсике записаны с лучшим показом структуры алгоритма. Весьма желательно нарисовать блок-схему алгоритма для его лучшего понимания. (Но если рисовать блок-схему для Вас затруднительно, то этого делать не надо.)

2. Проводим тестирование алгоритма. Для этого разбиваем входные данные на классы и берем по одной серии данных из каждого класса. Весьма желательно, но не обязательно, понять, как работает алгоритм.

3. Для примера входных данных, которые неправильно обрабатывает алгоритм, достаточно одного набора данных.

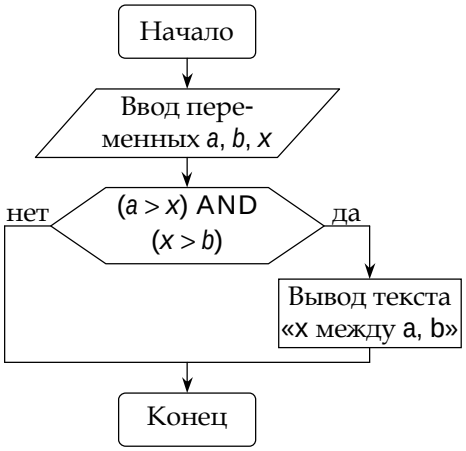
Задача 2006.C1

Ответ: $a = 1$, $x = 2$, $b = 3$.

Разумеется, в качестве ответа приведен один из возможных ответов. Для правильно решения задания этого вполне достаточно.

Найдем только такие входные данные,— числа a, b, x,— при которых программа работает неправильно. Остальную часть задания здесь выполнять не будем (задание полностью решено ниже в § 3).

Для полноты картины запишем приведенный в задаче алгоритм также на языке блок-схем.



Решение 1. Протестируем программу. Придется сначала выписать все принципиально различные случаи взаимного расположения переменных a, b и x, или хотя бы основные из них.

Основные случаи, подлежащие проверке, здесь следующие (тесты с равными числами рассматривать не будем):

1) $x < a < b$. В этом случае положим $x = 1$, $a = 2$, $b = 3$.
2) $a < x < b$. В этом случае положим $a = 1$, $x = 2$, $b = 3$.
3) $a < b < x$. В этом случае положим $a = 1$, $b = 2$, $x = 3$.
4) $x < b < a$. В этом случае положим $x = 1$, $b = 2$, $a = 3$.
5) $b < x < a$. В этом случае положим $b = 1$, $x = 2$, $a = 3$.
6) $b < a < x$. В этом случае положим $b = 1$, $a = 2$, $x = 3$.

Фактически нужно проверить один оператор:

На Паскале	На Бейсике
`IF (a>x) AND (x>b) THEN WRITELN('x между a,b');`	`IF (a>x) AND (x>b) THEN PRINT "x между a, b"`

1-й тест. Подставим в оператор значения переменных $x = 1$, $a = 2$, $b = 3$. В этом случае переменная x не лежит между a и b, и программа вывести сообщение не должна.

Получаем: **(a>x)** — истина, а **(x>b)** — ложь. Конъюнкция **(a>x) AND (x>b)** логических переменных **(a>x)** и **(x>b)** истинна только тогда, когда обе переменные истинны. Поэтому конъ-

юнкция ложна, оператор вывода не будет выполнен, сообщения не будет. В этом случае программа работает правильно.

2-й тест. Подставим в оператор значения переменных $a = 1$, $x = 2$, $b = 3$. В этом случае переменная x лежит между a и b, и программа должна вывести сообщение.

Получаем: **(a>x)** — ложь, и **(x>b)** — ложь. Конъюнкция **(a>x) AND (x>b)** ложна, и оператор вывода не будет выполнен, т. е. сообщения не будет. Программа работает неправильно!

Пример неправильной работы алгоритма найден. Проверку оставшихся тестов проводить не надо.

Решение 2. Проанализируем текст алгоритма и выясним, что на самом деле он делает.

Фактически нужно проверить один оператор:

На Паскале	На Бейсике
`IF (a>x) AND (x>b) THEN WRITELN('x между a,b');`	`IF (a>x) AND (x>b) THEN PRINT "x между a, b"`

Ясно, что логические условия **(a>x)** и **(x>b)** противоречат друг другу и не будут вместе выполнены ни при каких a, b и x. Итак, сообщение программа не выведет при *любых* значениях a, b и x, в том числе когда x лежит между a и b. Поэтому в качестве ответа подходят любые значения a, b и x, когда $a < x < b$.

Задача 2007.С1

Ответ: $a = 0$, $b = 0$, $x = 0$ (Значение x можно не указывать, допустим ответ, что x — любое число.)

Решение. Аналогично решениям предыдущей задачи.

Задача 2008.С1

Ответ: $a = 1$, $b = -1$, $x = 0$.

Решение. Аналогично решениям предыдущей задачи.

Задача 2009.С1

Ответ: $x = 3$, $y = 0{,}5$ (Любая пара (x, y), для которой выполняется: $y > 1$ или $x < 0$ или $(y \geqslant \sin x$ и $x > \pi/2$ и $y \leqslant 1)$.)

Решение. Аналогично решениям предыдущей задачи.

§ 3. Разработка алгоритмов

1. Теория

1°. Модульное программирование

Достаточно очевидно, что алгоритм и шаг — понятия относительные. Любой алгоритм может оказаться частью, т. е. шагом, более большого алгоритма. И наоборот, может оказаться, что шаг алгоритма сам является алгоритмом и его можно разбить на последовательность более мелких шагов. В этом смысле шаг алгоритма можно назвать подалгоритмом, а алгоритм по отношению к своему шагу — надалгоритмом.

Исходя из этих соображений, при проектировании алгоритма, решающего данную задачу, сначала проектируют алгоритм в целом, затем — его подалгоритмы, потом подалгоритмы подалгоритмов и т. д., пока не дойдут до элементарных структур.

Такое проектирование алгоритма называется *проектированием сверху вниз*.

Причем проектирование сверху вниз невозможно без *пошаговой детализации*.

Пошаговая детализация — замена одного шага алгоритма несколькими более простыми.

Проектирование сверху вниз — пошаговая детализация условия задачи до уровня элементарных действий для исполнителя.

Такой подход имеет два преимущества:

1) исходная задача сводится к решению более простых задач;

2) может оказаться, что некоторые подалгоритмы окажутся одинаковыми, что приведет к сокращению времени проектирования и повышению надежности окончательного алгоритма.

Итак, обучение программированию при составлении простых алгоритмов состоит из обучения двум классам методов:

1) методам пошаговой детализации;

2) методам различения действий, которые можно принять за элементарные.

При решении более сложных задач сочетают проектирование сверху вниз с проектированием снизу вверх.

Например, напишем алгоритм для задачи вычисления площади треугольника по сторонам.

1. Представим эту задачу в виде алгоритма из одного шага.

Алгоритм 1. Вычисление площади треугольника по сторонам.

1. Вычисление площади треугольника по его сторонам.

2. Детализируем первый шаг алгоритма 1.

Алгоритм 2. Вычисление площади треугольника по сторонам.

1. Задание сторон треугольника.
2. Вычисление площади треугольника по его сторонам.
3. Выдача результата.

3. Детализируем второй шаг алгоритма 2.

Алгоритм 3. Вычисление площади треугольника по сторонам.

1. Задание сторон треугольника.
2. Вычисление полупериметра треугольника.
3. Вычисление площади треугольника по формуле Герона.
4. Выдача результата.

В этом простом примере дальнейшая детализация не нужна.

Человек мыслит и сознает окружающую действительность блоками. Например, когда человек видит дерево, он представляет его в виде блока, состоящего из ствола, ветвей, листьев и т. д.

Блоки располагаются в пространстве двумя способами:

1) *последовательно*. Например, минимальными блоками книги является абзацы, которые располагаются один за другим;

2) *отдельно*. Например, в этой книге более полная информация по некоторым абзацам выделена в приложение. Кроме того, когда встречаются, например, единицы измерения байты и килобайты, то более подробную информацию следует искать в первой главе.

При проектировании алгоритма вовсе не обязательно выписывать все его шаги в виде одной длинной цепочки. Законченные по смыслу блоки удобно выделить в виде отдельных алгоритмов. Например, при вычислении синуса вовсе необязательно каждый раз выписывать его шаги; гораздо удобней описать вычисление синуса в виде отдельного алгоритма, к которому обращаться по мере необходимости.

Формализуем эти рассуждения.

Свойство дискретности алгоритма приводит к его *модульности*, без которого проектирование сверху вниз невозможно.

Заметим, что алгоритм решения задачи имеет только один вход — начало алгоритма, и только один выход — конец алгоритма.

Модуль — отрезок алгоритма, законченный по смыслу, т. е. обладающий следующими двумя свойствами:

1) имеет один вход и один выход;

2) представляется в виде отдельной задачи.

Модульность алгоритма — свойство большинства алгоритмов состоять из модулей.

Модули алгоритмов аналогичны абзацам текстов.

Черный ящик — устройство, о котором известно, что́ оно делает, но неизвестно, как.

Принцип черного ящика — представление модуля в виде черного ящика.

Проектирование сверху вниз — искусство построения алгоритма в виде иерархии модулей по принципу черного ящика.

Фактически при проектировании сверху вниз задача представляется вначале в виде последовательности нескольких модулей, которые имеют вид черных ящиков. Например, шаг 2 алгоритма 2 является модулем, имеющим вид черного ящика:

1) известен вход этого модуля (три числа — длины сторон треугольника) и его выход (одно число — площадь треугольника);

3) неизвестно, каким образом вычисляется площадь треугольника.

При дальнейшем проектировании алгоритма содержание *некоторых* черных ящиков по усмотрению программиста раскрывается. Например, в алгоритме 3 раскрыто содержание модуля конкретного вычисления площади треугольника по сторонам, состоящего из шагов 2 и 3: это формула Герона

$$s = \sqrt{p(p-a)(p-b)(p-c)},$$

где используется модуль извлечения квадратного корня, который имеет вид черного ящика: программист заставляет исполнителя использовать готовый модуль извлечения корня.

Итак, модули бывают двух видов.

Встроенный модуль — шаги алгоритма, образующие модуль, но никак от него формально не отделенные.

Отдельный модуль — шаги алгоритма, образующие модуль и оформленные в виде отдельного алгоритма.

Основной, или головной, модуль — отдельный модуль, с которого начинается алгоритм.

Примером встроенного модуля является вычисление площади треугольника по сторонам по формуле Герона — это шаги 2 и 3 в алгоритме 3. Переделаем этот алгоритм, состоящий из одного основного модуля, в алгоритм, состоящий из двух модулей: основного и отдельного.

Алгоритм 4. Вычисление площади треугольника по сторонам.

I. Основной модуль.

1. Задание сторон треугольника.
2. Передача трех чисел отдельному модулю вычисления площади треугольника по сторонам.
3. Получение одного числа от отдельного модуля вычисления площади треугольника по сторонам.
4. Выдача результата.

II. Отдельный модуль.

1. Получение трех чисел — длин сторон треугольника.
2. Вычисление полупериметра треугольника.
3. Вычисление площади треугольника по формуле Герона.
4. Возврат одного числа — площади треугольника.

В алгоритме 4 выделение отдельного модуля привело к следующим изменениям по сравнению с алгоритмом 3:

1) небольшому увеличению длины текста алгоритма;

2) небольшому увеличению толковости алгоритма.

При нескольких обращениях к одному модулю его отдельное выделение *уменьшает* текст и *повышает* степень понятности алгоритма.

Модульное программирование — представление алгоритма в виде иерархии вызывающих друг друга модулей.

Однако в этой книге используются настолько простые программы, что выделять отдельные модули мы, конечно, не будем.

2°. **Структурное программирование**

Приведенные выше три структуры: следование, выбор и цикл — являются *элементарными*. Причем элементарные структуры могут быть *вложены* друг в друга

Элементарная структура алгоритма — общее название трех структур: следования, выбора и цикла.

Вложение, или суперпозиция, элементарных структур — структура, представляющая собой элементарную структуру, у которой один шаг заменен другой элементарной структурой.

Приведем пример суперпозиции элементарных структур.

Алгоритм 5. Вычисление суммы абсолютных значений нескольких чисел.

1. Задание нескольких чисел, абсолютные значения которых нужно просуммировать, и их количества.
2. Присвоение сумме чисел значения нуль.
3. Если числа просуммированы, то шаги 4—7 пропускаются.
4. Если очередное число неотрицательно, то шаг 5 пропускается.
5. Умножение числа на –1.
6. Добавление к сумме этого числа.
7. Переход на шаг 3.
8. Выдача в качестве результата вычисленной суммы.

Современные технологии программирования используют только эти три структуры. Это не только проверено на практике, но и теоретически обосновано теоремой, которая принимается без доказательства и уточнения ее формулировки.

Эквивалентные алгоритмы — два алгоритма, решающие одинаково одну задачу, но составленные из разных шагов.

Теорема 6. Основная теорема структурного программирования.

Любой алгоритм можно переписать в виде эквивалентного алгоритма, состоящего только из трех элементарных структур.

Структурное программирование — проектирование алгоритмов только из элементарных структур, шаги которых могут быть элементарными структурами или вызовами отдельных модулей.

Структурное программирование является общепризнанным стандартом современного проектирования алгоритмов и обладает следующими преимуществами:

1) обеспечивает понятность алгоритма;
2) минимизирует количество ошибок;
3) уменьшает время проектирования;
4) облегчает модернизацию алгоритма.

Любой алгоритм можно записать при помощи только трех элементарных структур. Запишем их в форме блок-схем.

Проще всего с элементарной структурой следования. При записи алгоритма блок-схемой эта структура представляется двумя последовательными элементами «действие» (см. рис. 7).

Элементарная блок-схема следования — блок-схема элементарной структуры следования.

На рисунке 8 изображены две структуры следования, причем вторая структура вложена в первую. В получившейся блок-схеме действия 3—4 можно рассматривать как одно.

Такими вложениями можно получить цепочку из действий алгоритма любой конечной длины.

Действие структуры следования можно заменить на элемент «ввод-вывод» или «вызов отдельного модуля» (см. рис. 9).

Рис. 7. Элементарная структура следования

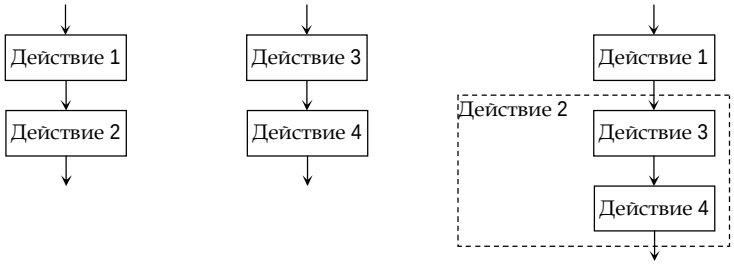


Рис. 8. Две элементарные структуры следования
и вложение второй структуры в первую

Рис. 9. Следования с элементами «ввод-вывод» и «вызов модуля»

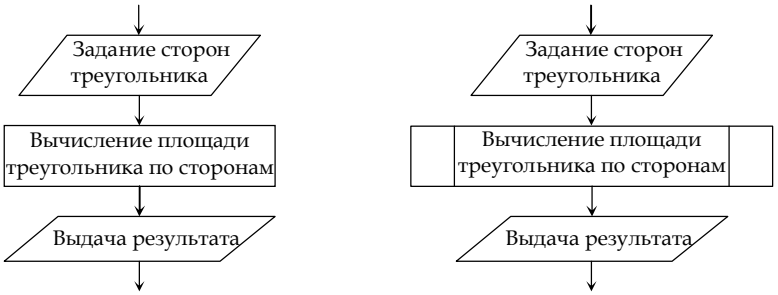


Рис. 10. Примеры представления шагов алгоритма следованием.
Шаги 1—3 алгоритма 3 (слева)
и шаги 1—4 основного модуля алгоритма 4 (справа)

Элементарная структура выбора конструируется из одного элемента «выбор» и одного или двух элементов «действие» с дополнительными стрелками. На рисунке 10 показан полный вариант элементарной структуры выбора: если при проверке условия оно истинно, то выполняется действие 2, если не выполняется — то 1.

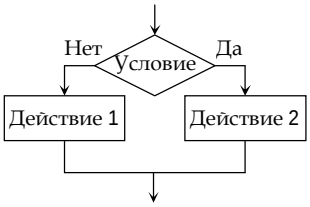


Рис. 10. Полная элементарная структура выбора

Элементарная блок-схема выбора, или ветвления, или альтернативы — блок-схема элементарной структуры выбора.

Имеются две сокращенные формы выбора, в которых отсутствует одно из действий. Таким образом, имеющееся единственное действие просто пропускается (см. рис. 11).

Все эти структуры выбора как единое целое имеют только один вход и только один выход.

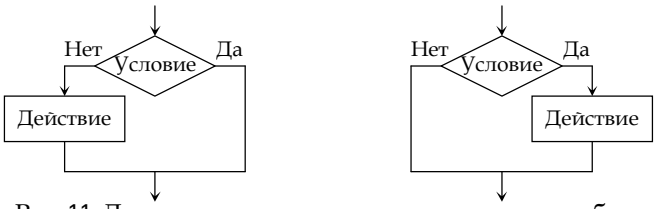


Рис. 11. Две неполные элементарные структуры выбора

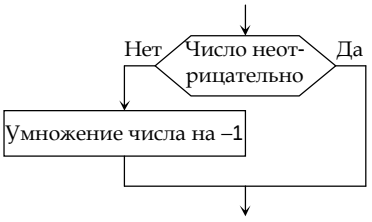


Рис. 12. Пример представления шагов алгоритма выбором. Шаги 2—3 алгоритма 5

Структура выбора может участвовать во вложении структур следующими тремя основными способами:

1) одно действие следования заменяется выбором (см. рис. 13). В этом случае всю элементарную структуру выбора можно рассматривать как одно действие, входящее в структуру следования;

2) одно действие выбора заменяется следованием (см. рис. 13). Здесь два действия, составляющие структуру следования, можно рассматривать как одно действие, входящее в структуру выбора;

3) одно действие выбора заменяется выбором (см. рис. 14). Тогда всю структуру второго, вложенного выбора можно рассматривать как одно действие, входящее в структуру первого, внешнего выбора.

Вкладывая следование и выбор друг в друга, можно получать блок-схемы алгоритмов без цикла любой сложности.

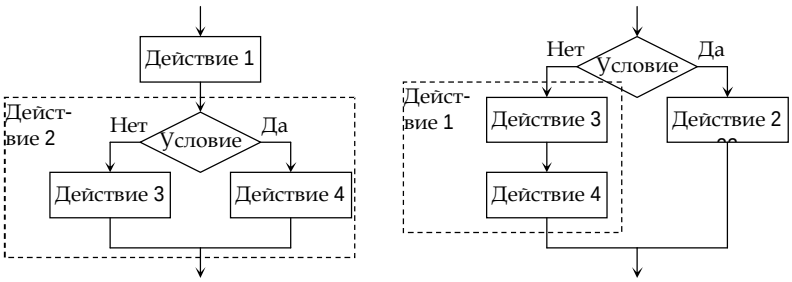


Рис. 13. Вложение выбора в следование (слева) и следования в выбор (справа)

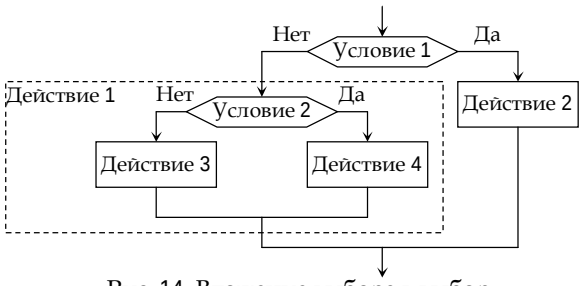


Рис. 14. Вложение выбора в выбор

Элементарная структура цикла конструируется из одного элемента «выбор с возвратом» и одного элемента «действие» с дополнительными стрелками. Смысл цикла заключается в том, что действие, входящее в его состав, может повторяться указанное число раз.

Элементарная блок-схема цикла, или повторения, или итерации — блок-схема элементарной структуры цикла. Действие цикла повторяется то тех пор, пока выполняется условие цикла.

Существует два типа способов записи цикла блок-схемой.

Элементарная блок-схема цикла с постусловием — запись в виде блок-схемы элементарной структуры цикла таким образом, что действие цикла в любом случае выполняется хотя бы один раз.

Имеется только один вариант цикла с постусловием, при котором используется элемент блок-схемы «выбор с возвратом» с началом условного перехода (см. рис. 15).

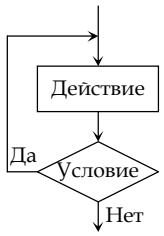


Рис. 15. Цикл с постусловием:
действие выполняется хотя бы один раз

Элементарная блок-схема цикла с предусловием — запись в виде блок-схемы элементарной структуры цикла таким образом, что действие цикла может не выполниться ни разу.

Имеется два варианта цикла с предусловием: при одном используется элемент «выбор с возвратом» с началом условного перехода, при другом — с окончанием безусловного перехода (см. рис. 16).

Вкладывая следование, выбор и цикл друг в друга, можно получить блок-схемы, которые представляют любые алгоритмы любой сложности. Пример вложения выбора в цикл показан на рисунке 18.

Все эти структуры цикла как единое целое имеют только один вход и только один выход.

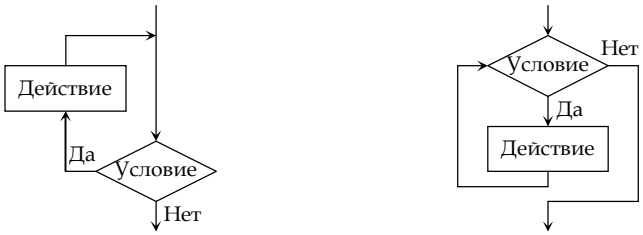


Рис. 16. Цикл с предусловием: в этом случае действие может не выполниться ни разу

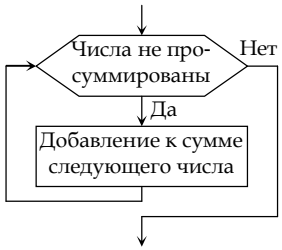


Рис. 17. Пример представления шагов алгоритма суммирования чисел циклом

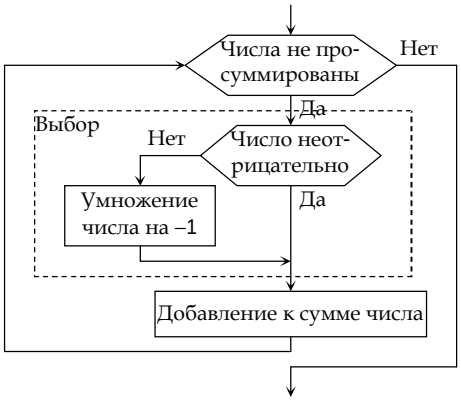


Рис. 18. Пример представления шагов алгоритма циклом и выбором. Шаги 3—7 алгоритма 5

3°. **Создание отдельных модулей**

Ранее рассматривались два способа записи алгоритмов: в виде текстового списка и в форме блок-схем. Исполнителями алгоритмов, записанных таким образом, может быть только человек.

> Имеются развитые системы записи алгоритмов в виде плоских графических фигур и стрелок. Эти системы составляются прямо на компьютере в специальных программах, и по ним компьютер затем автоматически создает код компьютерной программы.

Чтобы исполнителем алгоритма можно было сделать компьютер, алгоритм записывают на языке программирования, т. е. *кодируют*.

Кодирование, или реализация, или программирование — процесс записи алгоритма на языке программирования.

Листинг, или программа, или компьютерная программа — результат записи алгоритма на языке программирования, т. е. текст на языке программирования, кодирующий алгоритм.

Программа, или компьютерная программа — последовательность команд, которые выполняются на компьютере.

Чтобы создавать компьютерные программы, необходимо научиться хорошему *стилю программирования*.

Стиль программирования — правила проектирования и кодирования алгоритма.

Один из лучших стилей программирования состоит в следующем:

1) при проектировании алгоритма использовать:
   а) проектирование сверху вниз;
   б) модульное программирование;
   в) структурное программирование;
2) при кодировании алгоритма применять:
   а) комментарии;
   б) отступы при написании вложенных кодов;
   в) разбивку операторов на строки, отвечающую их структуре.

Дадим достаточно полное определение программирования, которое будем в дальнейшем использовать.

Программирование — система действий, алгоритм, состоящий из следующих компонентов, или шагов:

1) *анализ* — анализ проблемы;

2) *постановка* —постановка задачи;

3) *алгоритмирование* —проектирование алгоритмов;

4) *кодирование* — получение правильно работающей программы;

5) *сопровождение* — эксплуатация программы, связанная с повторением предыдущих четырех шагов.

Программист — человек, занимающийся программированием или хотя бы одним из его компонентов.

Системный аналитик — программист, специализирующийся на начальных этапах программирования: на анализе проблемы, постановке задачи и, может быть, проектировании алгоритмов.

Сложность осуществления компонентов программирования *уменьшается* от первого к четвертому.

Пятый шаг программирования рассматриваться, конечно, не будет как выходящий за рамки книги.

При создании простейших программ или только отдельных модулей достаточно использовать шаги программирования с первого по четвертый.

Технологии реализации третьего и четвертого шагов программирования рассмотрены следующих пунктах. Изучим вопрос об ошибках, возникающих при этих реализациях. Эти ошибки бывают двух видов.

Семантическая ошибка — смысловая ошибка в алгоритме.

Синтаксическая ошибка — ошибка кодирования программы, которая не позволяет компьютеру компилировать листинг.

Синтаксические ошибки бывают только на этапе кодирования программы на языке программирования. От них легко избавиться, поскольку они легко ищутся:

1) листинг программы не будет выполнен компьютером;

2) компилятор называет строку листинга с ошибкой и ее тип.

Семантические ошибки устраняются гораздо сложнее и живут в программах годами. Если в программе неожиданно выявилась существенная ошибка, то ее приходится устранять. Несущественные ошибки устранять не нужно.

Существенная ошибка — семантическая ошибка в программе, которая приводит в невозможности ее эксплуатации.

Несущественная ошибка — семантическая ошибка в программе, которая практически не мешает ее эксплуатации.

Отладка — процесс поиска и устранения синтаксических ошибок.

Тестирование — процесс поиска семантических ошибок, в случае успеха и нахождения существенной ошибки приводящий к изменению алгоритма программы.

Тестирование программы заключается в следующем.

1. Тестирование начинается на стадии алгоритмирования и продолжается на стадии кодирования, поскольку можно как неправильно закодировать алгоритм, так и выявить семантическую ошибку уже на стадии кодирования.

2. Тестирование заключается в пошаговом исполнении программы на нескольких сериях конкретных данных.

3. Тестирование очень маленькой программы, состоящей из одного основного модуля, заключается в его тестировании.

4. Тестирование модуля, вызывающего другой модуль, проводится в паре с простейшим вариантом вызываемого модуля. И наоборот. Другими словами, вызывающий и вызываемый модули взаимно тестируют друг друга.

4°. Составные части программы на Паскале и Бейсике

Паскаль, или язык программирования Паскаль (Pascal) — язык программирования, разработанный в 1970 г. профессором Цюрихского университета (Швейцария) и суперпрограммистом Никлаусом Виртом. Название языку дано в честь выдающегося французского математика, физика, литератора и философа Блеза Паскаля, сконструировавшего первый механический калькулятор.

Бейсик, или язык программирования Бейсик — язык программирования, разработанный в 1963 г. профессорами Дартмутского университета (Канада) Томасом Куртом и Джоном Кемени. (BASIC — сокращение от англ. *Beginner's All-purpose Symbolic Instruction Code* — универсальный код символических инструкций для начинающих; кроме того, англ. *basic* — основной, базовый.)

Синтаксис языка программирования — правила записи текстов на этом языке.

Описание языка программирования — описание синтаксиса языка программирования.

Листинг программы — текст на языке программирования, отвечающий правилам его синтаксиса.

Листинг на Паскале и Бейсике имеет одну составляющую, которая может встретиться почти в любом месте листинга.

Комментарий — часть листинга, которую компилятор не переводит в машинные команды и которая поэтому не влияет на выполнение программы.

Комментарий на Паскале заключается в фигурные скобки *{ }*, а на Бейсике начинается с символа апострофа ' и идет до конца строки.

Комментарий на Паскале и Бейсике можно записывать на русском языке, если, конечно, русский язык поддерживается текстовым редактором, на котором набирается листинг программы.

Комментирование — запись комментариев в программу для выполнения следующих функций:

1) добавления в листинг пояснений хода выполнения программы, пояснений данных и пояснений действий. Например, обязательно описываются назначение модулей и их входные и выходные данные;

2) временное или постоянное отключение от компиляции части листинга, которая не является комментарием. Отключение части листинга производится, например, при отладке программы.

Следующая структура листинга имеется только на Паскале.

На Паскале блок начинается со слова *begin* и заканчивается словом *end*. Другое название блока begin—end — *логические, или инструктивные, скобки*.

Begin по-английски означает «начало», end — «конец».

Листинг программы на Паскале и Бейсике состоит из последовательных листингов модулей.

Листинг отдельного модуля на Паскале и Бейсике имеет две составные синтаксические части, которые разбивают его на два последовательных раздела.

Переменная — именованное данное в программе.

Раздел описаний, или раздел описания переменных, или декларативная часть — первый раздел листинга отдельного модуля на Паскале и Бейсике. У основного модуля может сразу начинаться со слова var для Паскаля и dim для Бейсика, у Паскаля заканчивается точкой с запятой. Содержит описания переменных модуля.

Слово var является сокращением от английского variable, означающего «переменная», а dim — от английского dimensions, означающего «размеры».

Раздел описаний — первый раздел отдельного модуля на Паскале и Бейсике.

Оператор — именованное действие в программе.

Раздел операторов, или тело программы, или исполняемая часть — основной текст отдельного модуля на Паскале и Бейсике. Содержит запись алгоритма модуля с помощью операторов и переменных. На Паскале представляет собой блок самого верхнего уровня, поэтому начинается со слова begin и заканчивается словом end с точкой.

Раздел операторов — второй раздел отдельного модуля на Паскале и Бейсике. Общий вид отдельного модуля показан на листинге 19 для Паскаля и листинге 20 для Бейсика.

Простейшая программа на Паскале 21 и Бейсике 22 выводит одну строку сообщения. Она используется для тестирования системы программирования и просмотра экрана вывода системы. Программа настолько проста, что не содержит раздела описаний.

Листинг 19

Структура отдельного модуля на Паскале

```
{Раздел описаний}
VAR
    {Здесь находятся описания переменных}
;

{Раздел операторов}
BEGIN
    {Здесь находятся операторы отдельного модуля}
END.
```

Листинг 20

Структура отдельного модуля на Бейсике

```
{Раздел описаний}
DIM
    {Здесь находятся описания переменных}

{Раздел операторов}
    {Здесь находятся операторы отдельного модуля}
```

Листинг 21

Тест Паскаля

```
{Вывод одного сообщения}

BEGIN
    WRITELN ('Привет, мир!');
END.
```

Листинг 22

Тест Бейсика

```
'Вывод одного сообщения

    PRINT "Привет, мир!"
```

5°. Алгоритмы вычисления делителей целых чисел

Задача этого параграфа — научить программировать.

Здесь под программированием понимается составление алгоритма, его запись на языке блок-схем и языках программирования и, наконец, проверка правильности алгоритма, т. е. его тестирование.

Один из самых эффективных способов этому научить — изучать готовые конкретные простые алгоритмы.

Составим алгоритмы из § 1, которые описаны там как черные ящики. Словами опишем входные/выходные данные и идею алгоритма.

Алгоритмы словами описывать не будем, это примерно то же самое, как описывать словами математические формулы. Алгоритмы выпишем на языке блок-схем, на Паскале и Бейсике.

1. Вычисление наибольшего общего делителя.

Вход: 2 целых положительных числа n и m.

Выход: 1 целое положительное число $l = \text{НОД}(n, m)$.

Идея алгоритма. Проверим, являются ли делителями обоих чисел все числа от 1 до минимального из них. Максимальный делитель и будет НОД.

Алгоритм, решающий эту задачу, приведен в таблице 22. Он записан, слева направо, на Паскале, Бейсике и языке блок-схем. Операторы, которые соответствуют друг другу в разных языках, синхронны, т. е. находятся в одних и тех же строках.

Обратите внимание, что в блок-схемах можно не вводить конкретные данные, а в программах, записанных на Паскале и Бейсике, это делать необходимо, иначе программа не будет работать на компьютере. Мы ввели данные из первого теста.

Чтобы понять, как работает алгоритм, нужно просто внимательно разобрать, как работают модули, из которых он состоит. Каждый модуль снабжен коротким комментарием, т. е. заголовком. Это следующие последовательные модули:

1) ввод n и m;

2) вычисление $k = \min(n, m)$;

3) вычисление $l = \text{НОД}(n, m)$;

4) вывод $l = \text{НОД}(n, m)$.

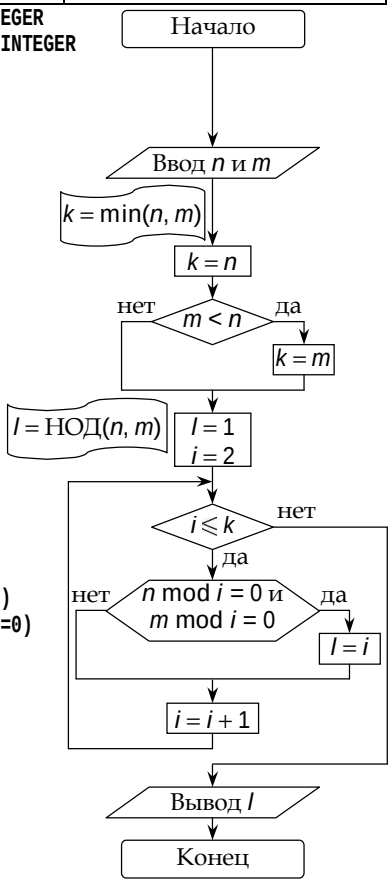
Таблица 22

Вычисление наибольшего общего делителя

Паскаль	Бейсик	Блок-схема

```
VAR n, m: INTEGER;
  l, k, i: INTEGER;

BEGIN
{Ввод n и m}
  n := 2;
  m := 2;

{k = min(n, m)}
  k := n;


  if (m<n) k := m;

{l = НОД(n, m)}
  l := 1;


  FOR i:=2 TO k DO
  BEGIN

    IF (n MOD i=0)
     AND (m MOD i=0)
     THEN l := i;

  END;

{Вывод l}
  WRITELN ('l =', l);

END.
```

```
DIM n, m AS INTEGER
DIM l, k, i AS INTEGER


'Ввод n и m
n = 2
m = 2

'k = min(n, m)
k = n


if (m<n) k := m

'l = НОД(n, m)
l = 1


FOR i=2 TO k


  IF (n MOD i=0)
   AND (m MOD i=0)
   THEN l := i

NEXT i

'Вывод l
PRINT "l ="; l
```



В этом разделе входные данные в языках программирования будем вводить оператором присваивания.

Алгоритм в виде блок-схемы всегда начинается с элемента «Начало» и заканчивается элементом «Конец».

Заметим, что поиск минимума n и m можно опустить, и использовать цикл до любого из двух чисел, например, до n. Просто если n окажется больше m, то цикл будет крутиться дольше, а сам алгоритм работать больше времени.

Этот поиск минимума добавлен для *оптимизации алгоритма*, т. е. для ускорения времени его работы.

Осталось протестировать наш алгоритм. Тестирование — это не только проверка того, правильно ли работает алгоритм.

Правильное тестирование — это очень хороший способ понять, как алгоритм работает.

Под правильным тестированием понимается, разумеется, честное, т. е. бесхитростное, т. е. тупое прохождение алгоритма, а не подонка результата к известному ответу.

Воспользуемся готовыми честными тестами из § 1, где было выделено четыре класса входных данных и величина НОД, которая должна получиться в каждом случае:

1) если $n = m$, то $l = n = m$;
2) если n делится на m, то $l = m$;
3) если m делится на n, то $l = n$;
4) если $n = 4$, а $m = 6$, то $l = 2$;
5) если $n = 10$, а $m = 4$, то $l = 2$.

Возьмем из каждого класса по одной серии данных, причем возьмем числа поменьше, и честно пройдем с этими данными наш алгоритм. Сведем полученное таким образом тестирование по пяти тестам в одну таблицу 23.

В каждом из пяти тестов получено правильное значение НОД. Можно сделать вывод, что алгоритм, скорее всего, работает правильно.

Напоминаем, что со стопроцентной уверенностью в правильной работе алгоритма можно быть уверенным, если протестировать его на всех возможных данных. И то только для алгоритма, записанного в виде блок-схемы.

Для конкретных реализаций алгоритма на языках программирования никогда нельзя быть уверенным, что они будут работать правильно для любых входных данных на любых компьютерах.

Таблица 23

Тестирование вычисления наибольшего общего делителя

1-й тест	2-й тест	3-й тест	4-й тест	5-й тест
$n = 2$ $m = 2$ $k = 2$ $m < n$? Нет $l = 1$ $i = 2$ $i \leqslant k$? Да n mod $i = 0$ и m mod $i = 0$? Да $l = 2$ $i = 3$ $i \leqslant k$? Нет Вывод $l = 2$	$n = 4$ $m = 2$ $k = 4$ $m < n$? Да $k = 2$ $l = 1$ $i = 2$ $i \leqslant k$? Да n mod $i = 0$ и m mod $i = 0$? Да $l = 2$ $i = 3$ $i \leqslant k$? Нет Вывод $l = 2$	$n = 2$ $m = 6$ $k = 2$ $m < n$? Нет $l = 1$ $i = 2$ $i \leqslant k$? Да n mod $i = 0$ и m mod $i = 0$? Да $l = 2$ $i = 3$ $i \leqslant k$? Нет Вывод $l = 2$	$n = 4$ $m = 6$ $k = 4$ $m < n$? Нет $l = 1$ $i = 2$ $i \leqslant k$? Да n mod $i = 0$ и m mod $i = 0$? Да $l = 2$ $i = 3$ $i \leqslant k$? Да n mod $i = 0$ и m mod $i = 0$? Нет $i = 4$ $i \leqslant k$? Да n mod $i = 0$ и m mod $i = 0$? Нет Вывод $l = 2$	$n = 10$ $m = 4$ $k = 10$ $m < n$? Да $k = 4$ $l = 1$ $i = 2$ $i \leqslant k$? Да n mod $i = 0$ и m mod $i = 0$? Да $l = 2$ $i = 3$ $i \leqslant k$? Да n mod $i = 0$ и m mod $i = 0$? Нет $i = 4$ $i \leqslant k$? Да n mod $i = 0$ и m mod $i = 0$? Нет Вывод $l = 2$

2. Вычисление наибольшего общего делителя.

Вход: 2 целых положительных числа n и m.

Выход: 1 целое положительное число $l = \text{НОД}(n, m)$.

Идея алгоритма. Найдем НОД более эффективным методом, а именно алгоритмом Евклида.

Напомним его суть. Если данные числа n и m делятся на одно и то же число l, т. е. $n = la$ и $m = lb$, тогда на то же число l делится и разность этих чисел: $n - m = la - lb = l(a - b)$.

Тогда НОД двух чисел можно искать, вычитая из большего меньшее и заменяя большее на разность до тех пор, пока разность не будет равна 0.

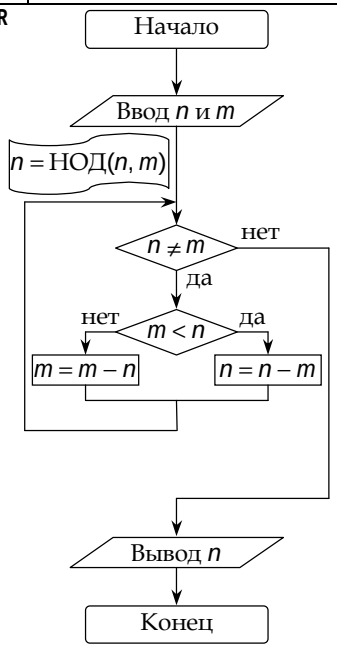
Проверим правильность этого алгоритма, правильно написав компьютерный алгоритм, который его реализует. Алгоритм состоит из трех модулей:

1) ввод n и m;
2) вычисление l = НОД(n, m);
3) вывод l = НОД(n, m).

Таблица 24

Вычисление наибольшего общего делителя

Паскаль	Бейсик	Блок-схема

`VAR n, m: INTEGER;`
 	
`BEGIN`
`{Ввод n и m}`
`  n := 2;`
`  m := 2;`
 	
`{n = НОД(n, m)}`
`  WHILE n<>m DO`
`  BEGIN`
 	
`    IF (m<n) THEN`
`      n := n - m;`
`    ELSE`
`      m := m - n;`
`    END;`
 	
`  END;`
 	
`{Вывод n}`
`  WRITELN ('n =', n);`
 	
`END.`
	`DIM n, m AS INTEGER`
	
	`'Ввод n и m`
	`n = 2`
	`m = 2`
	
	`'n = НОД(n, m)`
	`WHILE n<>m`
	
	`  IF (m<n) THEN`
	`    n := n - m`
	`  ELSE`
	`    m := m - n`
	`  ENDIF`
	
	`WEND`
	
	`'Вывод n`
	`PRINT "n ="; n`
		

Обратите внимание, что на Паскале и Бейсике пришлось применить другую форму цикла с предусловием — цикл while, т. к. заранее неизвестно количество проходов цикла.

Воспользуемся готовыми честными тестами из § 1, где было выделено четыре класса входных данных и величина НОД, которая должна получиться в каждом случае:

1) если $n = m$, то $l = n = m$;
2) если n делится на m, то $l = m$;
3) если m делится на n, то $l = n$;
4) если $n = 4$, а $m = 6$, то $l = 2$;
5) если $n = 10$, а $m = 4$, то $l = 2$.

В каждом из пяти тестов получено правильное значение НОД. Можно сделать вывод, что алгоритм, скорее всего, работает правильно.

Таблица 25

Тестирование вычисления наибольшего общего делителя

1-й тест	2-й тест	3-й тест	4-й тест	5-й тест
$n = 2$	$n = 4$	$n = 2$	$n = 4$	$n = 10$
$m = 2$	$m = 2$	$m = 6$	$m = 6$	$m = 4$
$n \neq m$? Нет	$n \neq m$? Да	$n \neq m$? Да	$n \neq m$? Да	$n \neq m$? Да
	$m < n$? Да	$m < n$? Нет	$m < n$? Нет	$m < n$? Да
$n = 2$	$n = n - m = 2$	$m = m - n = 4$	$m = m - n = 2$	$n = n - m = 6$
Вывод $n = 2$				
	$n \neq m$? Нет	$n \neq m$? Да	$n \neq m$? Да	$n \neq m$? Да
		$m < n$? Нет	$m < n$? Да	$m < n$? Да
	$n = 2$	$m = m - n = 2$	$n = n - m = 2$	$n = n - m = 2$
	Вывод $n = 2$			
		$n \neq m$? Нет	$n \neq m$? Нет	$n \neq m$? Да
				$m < n$? Нет
		$n = 2$	$n = 2$	$m = m - n = 2$
		Вывод $n = 2$	Вывод $n = 2$	
				$n \neq m$? Нет
				$n = 2$
				Вывод $n = 2$

3. Вычисление всех делителей.

Вход: 1 целое положительное число n.

Выход: 1) 1 целое m — количество всех делителей числа n;

2) массив l из m целых чисел — делителей числа n.

Идея алгоритма. Проверим все числа от 1 до n на делители n.

Алгоритм состоит из трех модулей:

1) ввод n;

2) вычисление массива l с делителями n и его длины m;

3) вывод массива l и его длины m.

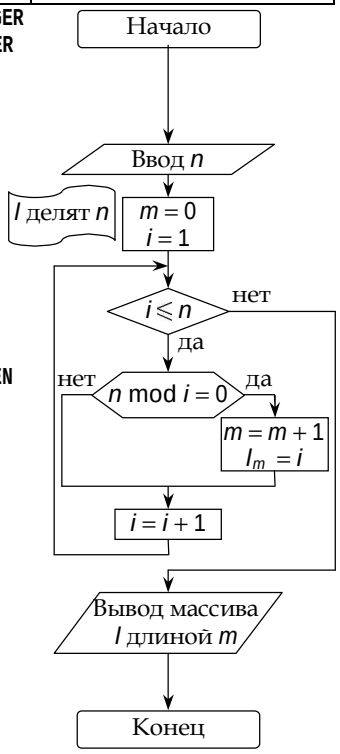
Таблица 26

Вычисление всех делителей

Паскаль	Бейсик	Блок-схема

```
VAR n, m, i: INTEGER;
    l: ARRAY [1..100]
       OF INTEGER;

BEGIN
{Ввод n}
 n := 2;

{l - делители n}
 m := 0;
 FOR i:=1 TO n DO
 BEGIN

  IF (n MOD i=0) THEN
  BEGIN
   m := m + 1;
   l[m] := i;
   END;

 END;

{Вывод мас. l дл. m}
 WRITELN ('m =', m);
 FOR i:=1 TO m DO
  WRITELN (l[i]);

END.
```

```
DIM n, m, i AS INTEGER
DIM l(100) AS INTEGER



'Ввод n
n = 2

'l - делители n
m = 0
FOR i=1 TO n


  IF (n MOD i=0) THEN

    m = m + 1
    l(m) = i
  ENDIF

NEXT i

'Вывод мас. l дл. m
PRINT "m ="; m
FOR i:=1 TO m
  PRINT l(i)
```



Обратите внимание, как описываются целые простые переменные n, m, i и массив l в разделе описания переменных Паскаля и Бейсика. Причем формальная длина массива (здесь 100) должна быть не меньше, чем количество используемых его элементов n.

Обратите также внимание на то, что у размерности массива в языках программирования (на блок-схеме этого нет) задан фиксированный размер массива: 100. В программировании размерность массивов нужно планировать заранее. В наших программах это приводит к ограничению на величину входного числа n, которое не должно превышать размерность массива: 100.

Воспользуемся готовыми честными тестами из § 1. Все они относятся к одну классу, поэтому оставим два из них:

1) если $n = 3$, то $m = 2$, $l = 1, 3$;

2) если $n = 4$, то $m = 3$, $l = 1, 2, 4$;

В каждом из двух тестов получено правильное значение делителей чисел и их количества. Можно сделать вывод, что алгоритм, скорее всего, работает правильно.

Таблица 27

Тестирование вычисления всех делителей

1-й тест		2-й тест	
$n = 3$		$n = 4$	
$m = 0$	$i = 3$	$m = 0$	$i = 3$
	$i \leqslant n$? Да		$i \leqslant n$? Да
$i = 1$	n mod $i = 0$? Да	$i = 1$	n mod $i = 0$? Нет
$i \leqslant n$? Да	$m = 2$	$i \leqslant n$? Да	
n mod $i = 0$? Да	$l_2 = 3$	n mod $i = 0$? Да	$i = 4$
$m = 1$		$m = 1$	$i \leqslant n$? Да
$l_1 = 1$	Вывод	$l_1 = 1$	n mod $i = 0$? Да
	Количество 2		$m = 3$
$i = 2$	Делители 1, 3	$i = 2$	$l_3 = 4$
$i \leqslant n$? Да		$i \leqslant n$? Да	
n mod $i = 0$? Нет		n mod $i = 0$? Да	Вывод
		$m = 2$	Количество 3
		$l_2 = 2$	Делители 1, 2, 4

6°. Алгоритмы решения уравнений

1. Решение линейного уравнения.

Вход: 2 числа: a — коэффициент при x и b — свободный коэффициент.

Выход: либо сообщение «любой x» (если $a = 0$, $b = 0$);
либо сообщение «решений нет» (если $a = 0$, $b \neq 0$);
либо число $x = -b / a$ (если $a \neq 0$).

Идея алгоритма. Проверим все указанные в выходе варианты.

Алгоритм состоит из двух модулей: 1) ввод a и b;
2) вычисление и вывод x.

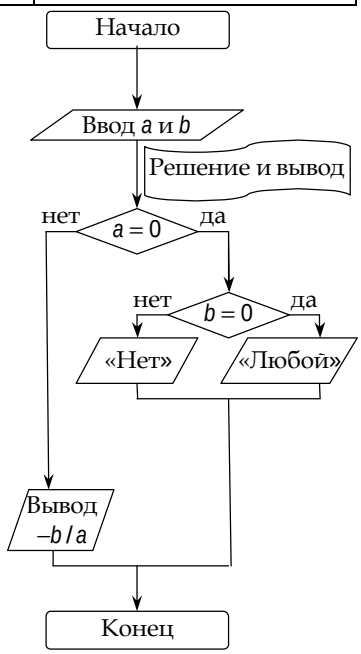
Таблица 28

Решение линейного уравнения

Паскаль	Бейсик	Блок-схема

```
VAR a, b: REAL;

BEGIN
{Ввод a, b}
 a := 0;
 b := 0;

{Решение и вывод}
 IF (a=0.0) THEN
 BEGIN

  IF (b=0.0) THEN
   WRITELN ('Любой');
  ELSE
   WRITELN ('Нет');

 ELSE

  WRITELN (-b/a);

 END;

END.
```

```
DIM a, b AS REAL


'Ввод a, b
a = 0
b = 0

'Решение и вывод
IF (a=0.0) THEN


  IF (b=0.0) THEN
    PRINT "Любой"
  ELSE
    PRINT "Нет"
  ENDIF
ELSE

  PRINT -b/a

ENDIF
```



Обратите внимание, как описываются вещественные простые переменные a и b в разделе описания переменных.

Воспользуемся готовыми честными тестами из § 1, относящихся к четырем классам:

1) если $a = 0{,}0$, $b = 0{,}0$, то вывод сообщения «любой x»;

2) если $a = 0{,}0$, $b = 1{,}0$, то вывод сообщения «решений нет»;

3) если $a = 1{,}0$, $b = 0{,}0$, то $x = 0{,}0$;

4) если $a = 1{,}0$, $b = 1{,}0$, то $x = -1{,}0$.

В каждом из четырех тестов получено правильное значение делителей чисел и их количества. Можно сделать вывод, что алгоритм, скорее всего, работает правильно.

Таблица 29

Тестирование решения линейного уравнения

1-й тест	2-й тест	3-й тест	4-й тест
$a = 0.0$ $b = 0.0$ $a = 0.0$? Да $b = 0.0$? Да Вывод «Любой»	$a = 0.0$ $b = 1.0$ $a = 0.0$? Да $b = 0.0$? Нет Вывод «Нет»	$a = 1.0$ $b = 0.0$ $a = 0.0$? Нет Вывод $-b / a = 0.0$	$a = 1.0$ $b = 1.0$ $a = 0.0$? Нет Вывод $-b / a = -1.0$

2. Решение квадратного уравнения.

Вход: 3 числа: a — коэффициент при x^2, b — коэффициент при x и c — свободный коэффициент.

Выход:

либо сообщение «любой x», если $a = 0$, $b = 0$, $c = 0$;

либо сообщение «решений нет», если $a = 0$, $b = 0$, $c \neq 0$;

либо число $x = -c / b$, если $a = 0$, $b \neq 0$;

либо сообщение «решений нет», если $a \neq 0$, $b^2 - 4ac < 0$;

либо число $x = -b / 2a$, если $a \neq 0$, $b^2 - 4ac = 0$;

либо 2 числа $x_1 = \dfrac{-b - \sqrt{b^2 - 4ac}}{2a}$, $x_2 = \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$, если $a \neq 0$, $b^2 - 4ac > 0$.

Идея алгоритма. Проверим все указанные в выходе варианты.

Алгоритм состоит из двух модулей: 1) ввод a и b;

2) вычисление и вывод x.

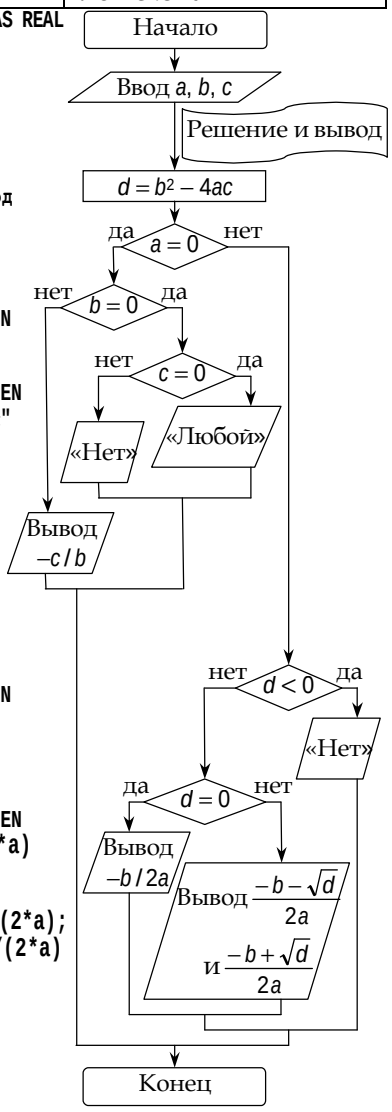
Таблица 30

Решение квадратного уравнения

Паскаль	Бейсик	Блок-схема

```
VAR a, b, c, d: REAL;
BEGIN
{Ввод a, b, c}
 a := 0;
 b := 0;
 c := 0;

{Решение и вывод}
 d := b*b-4*a*c;
 IF (a=0.0) THEN
 BEGIN

  IF (b=0.0) THEN
  BEGIN

   IF (c=0.0) THEN
    WRITELN ('Любой');
   ELSE
    WRITELN ('Нет');

  ELSE
   WRITELN (-c/b);

 END
 ELSE
 BEGIN

  IF (d<0.0) THEN
  BEGIN
   WRITELN ('Нет');
  ELSE

   IF (d=0.0) THEN
    WRITELN (-b/(2*a));
   ELSE
    WRITELN
    (-b-sqrt(d)/(2*a),
    -b+sqrt(d)/(2*a));


  END;

 END;
END.
```

```
DIM a, b, c, d AS REAL

'Ввод a, b, c
a = 0
b = 0
c = 0

'Решение и вывод
d = b*b-4*a*c
IF (a=0.0) THEN


 IF (b=0.0) THEN


  IF (c=0.0) THEN
   PRINT "Любой"
  ELSE
   PRINT "Нет"
  ENDIF
 ELSE
  PRINT -c/b
 ENDIF

ELSE


 IF (d<0.0) THEN


  PRINT "Нет"
 ELSE


  IF (d=0.0) THEN
   PRINT -b/(2*a)
  ELSE
   PRINT
    -b-sqrt(d)/(2*a);
    -b+sqrt(d)/(2*a)
  ENDIF


 ENDIF

ENDIF
```



Обратите внимание, что для уменьшения времени работы программы и для сокращения строк листингов, т. е. для увеличения легкости читаемости листингов, введена вспомогательная переменная d, равная дискриминанту квадратного уравнения. Воспользуемся готовыми честными тестами из § 1:

1) если $a = 0{,}0$, $b = 0{,}0$, $c = 0{,}0$, то вывод «любой x»;
2) если $a = 0{,}0$, $b = 0{,}0$, $c = 1{,}0$, то вывод «решений нет»;
3) если $a = 0{,}0$, $b = 1{,}0$, $c = 0{,}0$, то $x = 0{,}0$;
4) если $a = 0{,}0$, $b = 1{,}0$, $c = 1{,}0$, то $x = -1{,}0$;
5) если $a = 1{,}0$, $b = 0{,}0$, $c = 0{,}0$, то $x = 0{,}0$;
6) если $a = 1{,}0$, $b = 0{,}0$, $c = 1{,}0$, то вывод «решений нет»;
7) если $a = 1{,}0$, $b = 1{,}0$, $c = 0{,}0$, то $x_1 = -1{,}0$, $x_2 = 0{,}0$;
8) если $a = 1{,}0$, $b = 1{,}0$, $c = 1{,}0$, то вывод «решений нет».

Таблица 31

Тестирование решения квадратного уравнения

1-й тест	2-й тест	3-й тест	4-й тест
$a = 0.0$ $b = 0.0$ $c = 0.0$ $d = 0.0$ $a = 0.0$? Да $b = 0.0$? Да $c = 0.0$? Да Вывод «Любой»	$a = 0.0$ $b = 0.0$ $c = 1.0$ $d = 0.0$ $a = 0.0$? Да $b = 0.0$? Да $c = 0.0$? Нет Вывод «Нет»	$a = 0.0$ $b = 1.0$ $c = 0.0$ $d = 1.0$ $a = 0.0$? Да $b = 0.0$? Нет Вывод $-c / b = 0.0$	$a = 0.0$ $b = 1.0$ $c = 1.0$ $d = 1.0$ $a = 0.0$? Да $b = 0.0$? Нет Вывод $-c / b = -1.0$
5-й тест	6-й тест	7-й тест	8-й тест
$a = 1.0$ $b = 0.0$ $c = 0.0$ $d = 0.0$ $a = 0.0$? Нет $d < 0.0$? Нет $d = 0.0$? Да Вывод $-b / 2a = 0.0$	$a = 1.0$ $b = 0.0$ $c = 1.0$ $d = -4.0$ $a = 0.0$? Нет $d < 0.0$? Да Вывод «Нет»	$a = 1.0$ $b = 1.0$ $c = 0.0$ $d = 1.0$ $a = 0.0$? Нет $d < 0.0$? Нет $d = 0.0$? Нет Вывод $\frac{-b-\sqrt{d}}{2a} = -1$ и $\frac{-b+\sqrt{d}}{2a} = 0$	$a = 1.0$ $b = 1.0$ $c = 1.0$ $d = -3.0$ $a = 0.0$? Нет $d < 0.0$? Да Вывод «Нет»

7°. **Алгоритмы вычисления суммы чисел**

1. Вычисление суммы значений функции.

В общем виде математически это записывается так:

$$S = f(1) + f(2) + \mathrm{h} + f(n-1) + f(n) = \sum_{i=1}^{n} f(i).$$

Вход: 1 целое положительное число n — количество слагаемых в сумме.

Выход: 1 число S — сумма значений функции f при параметре i, пробегающем все значения от 1 до n.

Идея алгоритма. Введем сумматор S и просуммируем в цикле нужные значения функции. Функция $f = 1/i$ задается арифметическим выражением в цикле.

Алгоритм состоит из трех модулей:

1) ввод n;

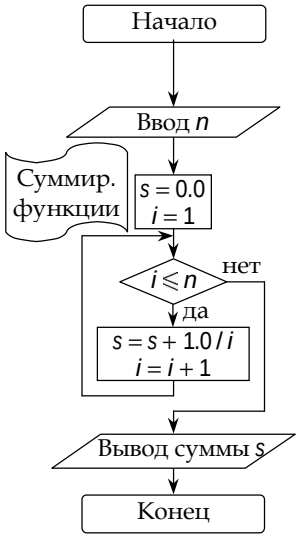
2) вычисление S;

3) вывод S.

Таблица 32

Вычисление суммы значений функции

Паскаль	Бейсик	Блок-схема

`VAR n, i: INTEGER;`
`  s: REAL;`
 	
`BEGIN`
`{Ввод n}`
`  n := 2;`
 	
`{Суммирование функции}`
`  s := 0.0;`
 	
`  FOR i:=1 TO n DO`
`  BEGIN`
`  s := s + 1.0/i;`
`  END;`
 	
`{Вывод суммы}`
`  WRITELN ('S =', S);`
 	
`END.`
	`DIM n, i AS INTEGER`
	`DIM s AS REAL`
	
	`'Ввод n`
	`n = 2`
	
	`'Суммирование функции`
	`s = 0.0`
	
	`FOR i=1 TO n`
	
	`  s = s + 1.0/i`
	`NEXT i`
	
	`'Вывод суммы`
	`PRINT "s ="; s`
		

Воспользуемся готовыми честными тестами из § 1:

1) если $f = 1 / i$, $n = 2$, то $s = 1{,}5$;

2) если $f = 1 / i$, $n = 3$, то $s = 11 / 6$.

Таблица 33

Тестирование вычисления суммы значений функции

1-й тест		2-й тест	
$n = 2$		$n = 3$	
	$i = 3$		$i = 3$
$s = 0$	$i \leqslant n$? Нет	$s = 0$	$i \leqslant n$? Да
$i = 1$		$i = 1$	$s = 1{,}5 + 1{,}0 / 3 =$
$i \leqslant n$? Да	Вывод 1,5	$i \leqslant n$? Да	$= 3 / 2 + 1 / 3 =$
$s = 0{,}0 + 1{,}0 / 1 =$		$s = 0{,}0 + 1{,}0 / 1 =$	$= 11 / 6$
$= 1{,}0$		$= 1{,}0$	
			$i = 4$
$i = 2$		$i = 2$	$i \leqslant n$? Нет
$i \leqslant n$? Да		$i \leqslant n$? Да	
$s = 1{,}0 + 1{,}0 / 2 = 1{,}5$		$s = 1{,}0 + 1{,}0 / 2 = 1{,}5$	Вывод $11 / 6$

2. Вычисление суммы массива.

В общем виде математически это записывается так:

$$s = a_1 + a_2 + \text{h} + a_{n-1} + a_n = \sum_{i=1}^{n} a_i.$$

Вход:

1) 1 целое положительное число n — длина массива a;

2) n чисел, организованных в виде массива a.

Выход: 1 число s — сумма массива a.

Идея алгоритма. Введем сумматор s и просуммируем в цикле значения элементов массива.

Алгоритм состоит из трех модулей:

1) ввод n элементов массива и его длины n;

2) вычисление s;

3) вывод s.

Воспользуемся готовыми честными тестами из § 1:

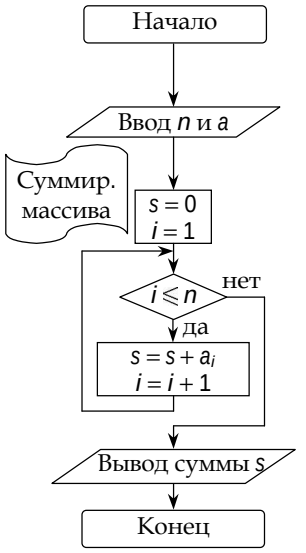
1) если $n = 2$, $a = 1, 2$, то $s = 3$;

2) если $n = 2$, $a = 2{,}5, 2{,}0$, то $s = 4{,}5$.

Таблица 34

Вычисление суммы массива

Паскаль	Бейсик	Блок-схема

`VAR n, s, i: INTEGER;`
`  a: ARRAY [1..100]`
`  OF INTEGER;`
`BEGIN`
`{Ввод n и массива a}`
`  n := 2;`
`  a[1] := 1;`
`  a[2] := 2;`
 	
`{Суммирование массива}`
`  s := 0;`
`  FOR i:=1 TO n DO`
`  BEGIN`
`  s := s + a[i];`
`  END;`
 	
`{Вывод суммы массива}`
`  WRITELN ('s =', s);`
 	
`END.`
	`DIM n, s, i AS INTEGER`
	`DIM a(100) AS INTEGER`
	
	`'Ввод n и массива a`
	`n = 2`
	`a(1) = 1`
	`a(2) = 2`
	
	`'Суммирование массива`
	`s = 0`
	`FOR i=1 TO n`
	
	`  s = s + a(i)`
	`NEXT i`
	
	`'Вывод суммы массива`
	`PRINT "s ="; s`
		

При переходе от суммирования целых чисел к суммированию вещественных следует в листингах заменить описания целых массива *a* и сумматора *s* на описания вещественных массива и сумматора.

Таблица 35

Тестирование вычисления суммы массива

1-й тест		2-й тест	
n = 2		*n* = 2	
a[1] = 1	*i* = 2	a[1] = 2,5	*i* = 2
a[2] = 2	*i* ⩽ *n*? Да	a[2] = 2,0	*i* ⩽ *n*? Да
	s = 1 + 2 = 3		*s* = 2,5 + 2,0 = 4,5
s = 0		*s* = 0,0	
i = 1	*i* = 3	*i* = 1	*i* = 3
i ⩽ *n*? Да	*i* ⩽ *n*? Нет	*i* ⩽ *n*? Да	*i* ⩽ *n*? Нет
s = 0 + 1 = 1	Вывод 3	*s* = 0,0 + 2,5 = 2,5	Вывод 4,5

8°. **Алгоритмы поиска**

1. Поиск заданного числа в массиве.

Вход: 1) 1 целое положительное число n — длина массива a;

2) n чисел, организованных в виде массива a;

3) 1 число b.

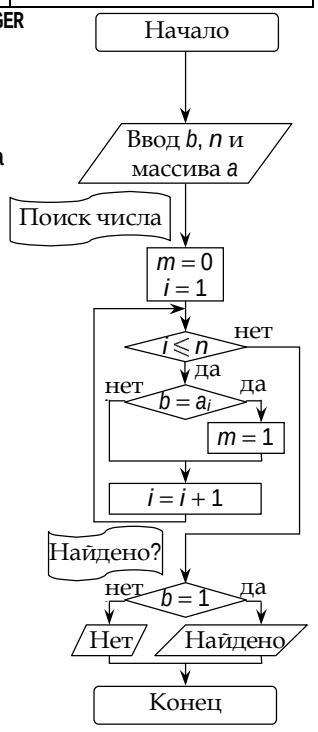
Выход: вывод сообщения «найдено», если число b входит в массив, и вывод сообщения «не найдено» в противном случае.

Идея алгоритма. Просмотрим в цикле все элементы массива и сравним их с заданным числом. Если элемент найден, то присвоим $m = 1$, если не найден — то оставим $m = 0$.

Таблица 36

Поиск заданного числа в массиве

Паскаль	Бейсик	Блок-схема

`VAR n, b, i, m: INTEGER;`
`  a: ARRAY [1..100]`
`  OF INTEGER;`
 	
`BEGIN`
`{Ввод b, n, массива a}`
`b := 2;`
`n := 3;`
`a[1] := 1;`
`a[2] := 3;`
`a[3] := 2;`
 	
`{Поиск числа}`
`m := 0;`
`FOR i:=1 TO n DO`
`IF (b = a[i]) THEN`
`m := 1;`
 	
`{Вывод результата}`
`IF (m = 1) THEN`
`WRITELN ('Найдено');`
`ELSE`
`WRITELN ('Нет');`
 	
`END.`
	`DIM n, b, i, m AS INTEGER`
	`DIM a(100) AS INTEGER`
	
	`'Ввод b, n и массива a`
	`b = 2`
	`n = 3`
	`a(1) = 1`
	`a(2) = 3`
	`a(3) = 2`
	
	`'Поиск числа`
	`m = 0`
	`FOR i=1 TO n`
	`IF (b = a(i)) THEN`
	`m = 1`
	`NEXT i`
	
	`'Вывод результата`
	`IF (m = 1) THEN`
	`PRINT "Найдено"`
	`ELSE`
	`PRINT "Нет"`
	`ENDIF`
		

Алгоритм состоит из трех модулей:

1) ввод элементов массива a, его длины n и данного числа b;

2) вычисление значения переменной m;

3) вывод «найдено» при $m = 1$ и «не найдено» при $m = 0$.

Воспользуемся готовыми честными тестами из § 1:

если $n = 3$, $a = 1, 3, 2$, $b = 2$, то «найдено»;

если $n = 3$, $a = 2{,}5, 2{,}0, 1{,}5$, $b = 0{,}5$, то «не найдено»;

если $n = 3$, $a = 1, 2, 4$, $b = 2$, то «найдено».

Примеров тестирования алгоритмов выше было уже достаточно много. Поэтому теперь тестирование алгоритмов предоставляется читателю.

Естественно, что массив может состоять не только из чисел, но и из символов. Поиск заданного символа в таком массиве оставляется читателю. Возможно, при этом потребуются консультации.

2. Поиск индексов вхождений числа в массив.

Вход:

1) 1 целое положительное число n — длина массива a;

2) n чисел, организованных в виде массива a;

3) 1 число b.

Выход:

1) 1 целое положительное число k — длина массива l;

2) k чисел, организованных в виде массива l.

Идея алгоритма. Просмотрим в цикле все элементы входного массива и сравним их с заданным числом. Если элемент равен числу, то сохраним его индекс в выходном массиве.

Воспользуемся готовыми честными тестами из § 1:

если $n = 3$, $a = 1, 2, 2$, $b = 2$, то $k = 2$, $l = 2, 3$;

если $n = 3$, $a = 2{,}5, 2{,}0, 2{,}5$, $b = 0{,}5$, то $k = 0$;

если $n = 3$, $a = 1, 2, 4$, $b = 2$, то $k = 1$, $l = 2$.

Естественно, что массив может состоять не только из чисел, но и из символов. Поиск заданного символа в таком массиве и накопление его индексов оставляется читателю.

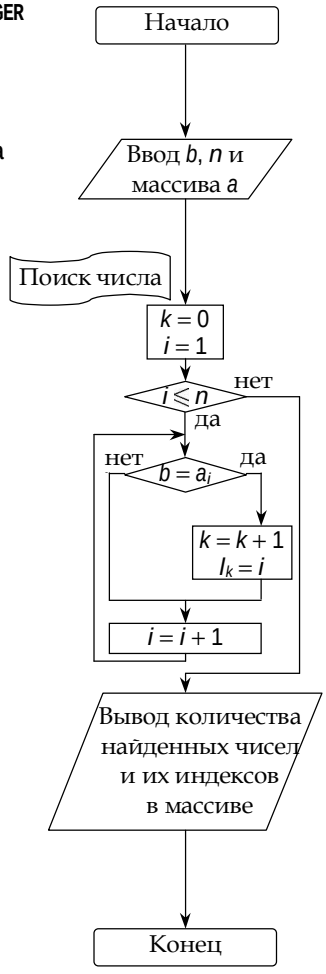
Таблица 37

Поиск индексов вхождений числа в массив

Паскаль	Бейсик	Блок-схема

```
VAR n, b, k, i: INTEGER;
  a, l: ARRAY [1..100]
        OF INTEGER;

BEGIN
{Ввод b, n, массива a}
 b := 2;
 n := 3;
 a[1] := 1;
 a[2] := 3;
 a[3] := 2;

{Поиск числа}
 k := 0;
 FOR i:=1 TO n DO
 BEGIN

  IF (b = a[i]) THEN
  BEGIN
   k := k + 1;
   l[k] := i;
  END;

 END;

{Вывод результата}
 WRITELN
  ('Всего чисел', k);

 WRITELN ('Индексы:');

 FOR i:=1 TO k DO
  WRITELN (l[i]);

END.
```

```
DIM n, b, k, i AS INTEGER
DIM a(100) AS INTEGER
DIM l(100) AS INTEGER


'Ввод b, n и массива a
b = 2
n = 3
a(1) = 1
a(2) = 3
a(3) = 2

'Поиск числа
k = 0
FOR i=1 TO n


  IF (b = a(i)) THEN

    k = k + 1
    l(k) = i
ENDIF

NEXT i

'Вывод результата
  PRINT
"Всего чисел"; k

  PRINT 'Индексы:'

FOR i=1 TO k
  PRINT l(i)
NEXT i
```



9°. Алгоритмы вычисления экстремальных значений

1. Вычисление максимального элемента массива.

Математически: $m = \max(a_1, a_2, \ldots, a_{n-1}, a_n)$.

Вход: 1) 1 целое положительное число n — длина массива a;

2) n чисел, организованных в виде массива a.

Выход: 1 число m — максимум массива a.

Идея алгоритма. Возьмем за максимум первый элемент массива и сравним его со всеми остальными, при этом если найдем больший элемент, то заменим максимум.

Алгоритм состоит из трех модулей:

1) ввод n элементов массива a и его длины n;

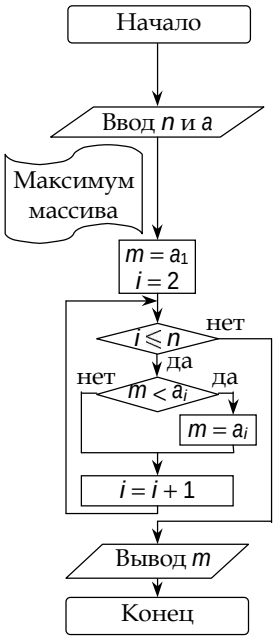
2) вычисление m;

3) вывод m.

Таблица 38

Вычисление максимума массива

Паскаль	Бейсик	Блок-схема

`VAR n, m, i: INTEGER;`
`  a: ARRAY [1..100]`
`  OF INTEGER;`
`BEGIN`
`{Ввод n и массива a}`
`  n := 3;`
`  a[1] := 1;`
`  a[2] := 3;`
`  a[3] := 2;`
 	
`{Поиск максимума}`
`  m := a[1];`
`  FOR i:=2 TO n DO`
`  BEGIN`
`  IF (m < a[i]) THEN`
`  m := a[i];`
 	
`  END;`
 	
`{Вывод максимума}`
`  WRITELN ('m =', m);`
 	
`END.`
	`DIM n, m, i AS INTEGER`
	`DIM a(100) AS INTEGER`
	
	`'Ввод n и массива a`
	`n = 3`
	`a(1) = 1`
	`a(2) = 3`
	`a(3) = 2`
	
	`'Поиск максимума`
	`m = a(1)`
	`FOR i=2 TO n`
	
	`  IF (m < a(i)) THEN`
	`  m = a(i)`
	
	`NEXT i`
	
	`'Вывод максимума`
	`PRINT "m ="; m`
		

При переходе от поиска максимума целых чисел к вещественным следует в листингах заменить описания целого массива *a* и целого максимума *m* на описания вещественных.

Воспользуемся готовыми честными тестами из § 1:

если $n = 3$, $a = 1, 3, 2$, то $m = 3$;

если $n = 3$, $a = 2{,}5, 2{,}0, 1{,}5$, то $m = 2{,}5$;

если $n = 3$, $a = 1, 2, 4$, то $m = 4$.

Аналогично можно найти минимум массива. Эта задача также предоставляется читателю.

2. Вычисление индекса максимального элемента массива.

Выше был найден максимум массива в виде максимального элемента. При этом информация о местонахождении элемента в массиве, т. е. его индексе, была утеряна. Если искать и сохранять индекс максимума массива, то и место максимума будет вычислено, и сам максимум тоже.

Вход: 1) 1 целое положительное число *n* — длина массива *a*;

2) *n* чисел, организованных в виде массива *a*.

Выход: 1 число *l* — индекс максимума массива *a*.

Идея алгоритма. Возьмем за максимум первый элемент массива и сравним его со всеми остальными, при этом если найдем больший элемент, то заменим максимум. Максимальный элемент будет сохранять в виде его индекса.

Алгоритм состоит из трех модулей:

1) ввод *n* элементов массива *a* и его длины *n*;

2) вычисление индекса *l*;

3) вывод максимума и его индекса *l*.

Воспользуемся теми же самыми готовыми честными тестами из § 1, что и в предыдущем алгоритме:

если $n = 3$, $a = 1, 3, 2$, то $l = 3$;

если $n = 3$, $a = 2{,}5, 2{,}0, 1{,}5$, то $l = 2{,}5$;

если $n = 3$, $a = 1, 2, 4$, то $l = 4$.

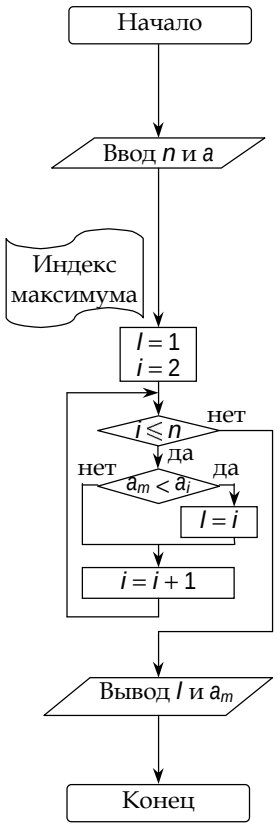
Кстати, представленный алгоритм поиска индекса максимума находит *последний*, самый большой индекс максимума в массиве. Составление алгоритма поиска *первого* индекса максимума массива оставляется читателю.

Кстати, эти задачи поиска первого и последнего индексов имеют несколько решений, которые, скорей всего, потребуют консультаций специалиста.

Таблица 39

Вычисление индекса максимума массива

Паскаль	Бейсик	Блок-схема

`VAR n, l, i: INTEGER;`
`  a: ARRAY [1..100]`
`  OF INTEGER;`
 	
`BEGIN`
`{Ввод n и массива a}`
`  n := 3;`
`  a[1] := 1;`
`  a[2] := 3;`
`  a[3] := 2;`
 	
`{Поиск индекса}`
`{последнего максимума}`
`  l := 1;`
 	
`  FOR i:=2 TO n DO`
`  BEGIN`
`  IF (a[l] < a[i])`
`  THEN`
`  l := i;`
 	
`  END;`
 	
`{Вывод максимума}`
`{и его индекса}`
`  WRITELN`
`  ('индекс =', l,`
`  'макс. =', a[l]);`
 	
`END.`
	`DIM n, l, i AS INTEGER`
	`DIM a(100) AS INTEGER`
	
	`'Ввод n и массива a`
	`n = 3`
	`a(1) = 1`
	`a(2) = 3`
	`a(3) = 2`
	
	`'Поиск индекса`
	`'последнего максимума`
	`l = 1`
	
	`FOR i=2 TO n`
	
	`  IF (a(l) < a(i))`
	`THEN`
	`  l = i`
	
	`NEXT i`
	
	`'Вывод максимума`
	`'и его индекса`
	`PRINT`
	`  "индекс ="; l;`
	`  "макс. ="; a[l]`
		

3. Вычисление всех индексов максимальных элементов массива.

Итак, мы узнали, как в массиве можно найти один из максимумов.

А теперь найдем сразу все максимумы массива, т. е. все места, где они находятся. Сделать это можно одним способом: собрать индексы всех максимумов массива вместе, в одном выходном массиве.

Вход: 1) 1 целое положительное число n — длина массива a;

2) n чисел, организованных в виде массива a.

Выход: 1) 1 целое положительное число k — длина массива l;

2) k чисел, организованных в виде массива l.

Идея алгоритма. Сначала найдем максимум, а потом все его индексы. Другими словами, цикл с просмотром массива будет использован дважды, один за другим.

Итак, алгоритм будет состоять уже не из трех, а из четырех модулей:

1) ввод n элементов массива a и его длины n;

2) вычисление максимума m;

3) вычисление индексов максимумов l и их количества k;

4) вывод максимума m, его индексов l и их количества k.

Этот алгоритм включает два прохода входного цикла, в котором ищутся максимумы:

1) при первом проходе находится величина максимума массива;

2) при втором проходе находятся индексы элементов массива, в которых находится в массиве вычисленный ранее максимум, и их количество. Этот алгоритм был уже описан выше как вычисление индексов всех вхождений заданного числа в заданный массив.

Воспользуемся готовыми честными тестами из § 1:

если $n = 3$, $a = 1, 2, 2$, то $k = 2$, $l = 2, 3$;

если $n = 3$, $a = 2{,}5, 2{,}0, 2{,}5$, то $k = 2$, $l = 1, 3$;

если $n = 3$, $a = 1, 2, 4$, то $k = 1$, $l = 3$.

Если хорошо подумать, то можно написать алгоритм, который вычисляет все индексы максимума массива за один проход массива. С одной стороны, такой алгоритм будет немного более быстро работать. Но с другой стороны, структура алгоритма существенно усложнится. Такое усовершенствование алгоритма остается на разработку читателя.

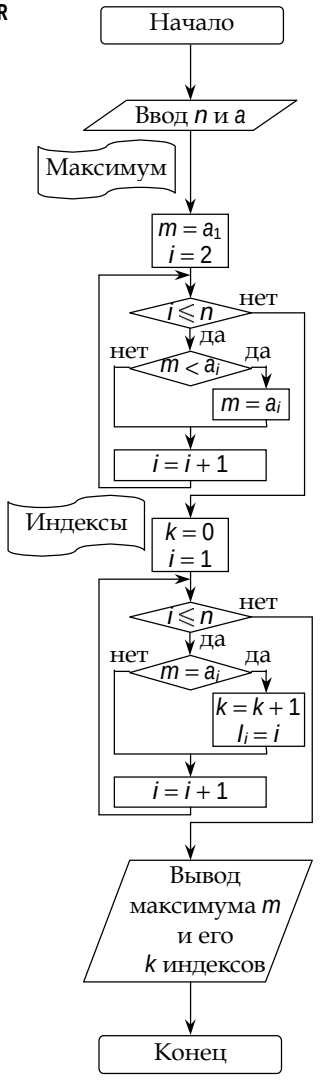
Таблица 40

Вычисление всех индексов максимумов массива

Паскаль	Бейсик	Блок-схема

```
VAR n, m, i: INTEGER;
  a, l: ARRAY [1..100]
       OF INTEGER;
BEGIN
{Ввод n и массива a}
  n := 3;
  a[1] := 1;
  a[2] := 2;
  a[3] := 2;

{Поиск максимума}
  m := a[1];
  FOR i:=2 TO n DO
  BEGIN
    IF (m < a[i]) THEN
      m := a[i];

  END;

{Поиск индексов}
  k := 0;
  FOR i:=1 TO n DO
  BEGIN
    IF (m = a[i]) THEN
    BEGIN
      k := k + 1;
      l[k] := i;
    END;

  END;

{Вывод макс. и инд.}
  WRITELN
    ('макс. =', m);
  WRITELN
    (k, ' индексов');
  FOR i:=1 TO k DO
    WRITELN (l[i]);
END.
```

```
DIM n, m, i AS INTEGER
DIM a(100) AS INTEGER
DIM l(100) AS INTEGER

'Ввод n и массива a
n = 3
a(1) = 1
a(2) = 2
a(3) = 2

'Поиск максимума
m = a(1)
FOR i=2 TO n

  IF (m < a(i)) THEN
    m = a(i)

NEXT i

'Поиск индексов
k = 0
FOR i=1 TO n

  IF (m = a(i)) THEN

    k = k + 1
    l(k) = i
ENDIF

NEXT i

'Вывод макс. и инд.
PRINT
  "макс. ="; m
PRINT
  k; " индексов"
FOR i:=1 TO k
  PRINT l(i)
```



10°. **Алгоритмы сортировки**

1. Сортировка массива.

Вход:

1) 1 целое положительное число n — длина массива a;

2) n чисел, организованных в виде массива a.

Выход: n отсортированных чисел массива a.

Идея алгоритма. Сначала найдем индекс максимума. Потом переставим максимум в коней массива. Снова найдем индекс максимума, но уже без последнего элемента, и переставим этот максимум на предпоследнее место. И т. д. до двух первых элементов массива. Организовать это «и т. д.» можно только с помощью внешнего цикла, внешнего по отношению к циклу поиска максимума.

В алгоритме встречается перестановка элементов массива. Это частный случай обмена значениями двух переменных. Две переменные обмениваются значениями с использованием вспомогательной переменной.

Также обратите внимание на то, что переменная j в алгоритме сортировки из таблицы 41 действует только внутри второго, внутреннего цикла. А переменная i работает внутри первого цикла, захватывая и второй.

Очень важно, что каждый цикл работает со своей переменной цикла. Эти переменные смешивать нельзя.

Итак, алгоритм будет состоять из трех модулей:

1) ввод n элементов массива a и его длины n;

2) сортировка n элементов массива a;

3) вывод n элементов массива a.

Совсем неочевидно, как работает этот алгоритм, поэтому очень полезно протестировать его на самых разнообразных тестах.

По крайней мере, необходимо воспользоваться готовыми честными тестами из § 1:

если $n = 3$, $a = 3, 1, 2$, то $a = 1, 2, 3$;

если $n = 3$, $a = 2{,}5, 2{,}0, 1{,}5$, то $a = 1{,}5, 2{,}0, 2{,}5$;

если $n = 3$, $a = 1, 2, 4$, то $a = 1, 2, 4$.

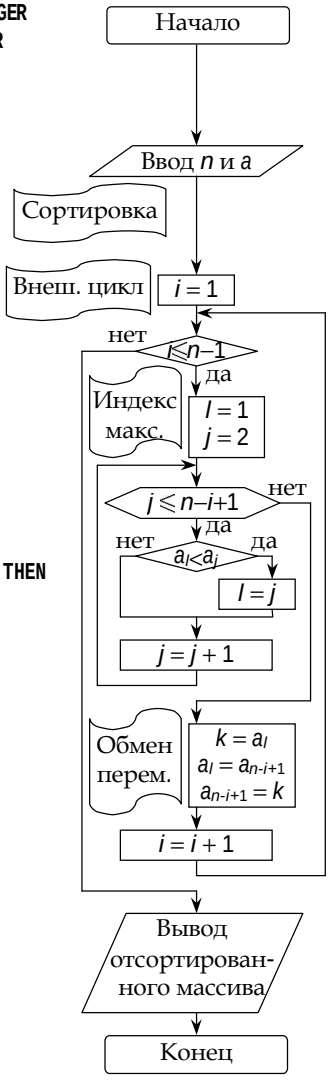
Таблица 41

Сортировка массива

Паскаль	Бейсик	Блок-схема

```
VAR n, i, j, k: INTEGER;
    a: ARRAY [1..100]
       OF INTEGER;

BEGIN
{Ввод n и массива a}
 n := 3;
 a[1] := 1;
 a[2] := 3;
 a[3] := 2;

{Сортировка}
{Внешний цикл}
 FOR i:=1 TO n-1 DO

{Поиск индекса макс.}
{в части массива}
  l := i;
  FOR j=2 TO n-i+1 DO
  BEGIN
   IF (a[l] < a[j]) THEN
      l := j;

  END;

{Обмен переменных}
  k := a[l];
  a[l] := a[n-i+1];
  a(n-i+1) := k;

 END;

{Вывод отсортиров.}
{массива}
  FOR i:=1 TO n DO
    WRITELN (a[i]);

END.
```

```
DIM n, i, j, k AS INTEGER
DIM a(100) AS INTEGER



'Ввод n и массива a
n = 3
a(1) = 1
a(2) = 3
a(3) = 2

'Сортировка
'Внешний цикл
FOR i=1 TO n-1

'Поиск индекса макс.
'в части массива
  l = i
  FOR j=2 TO n-i+1

    IF (a(l) < a(j)) THEN
      l = j

  NEXT j

'Обмен переменных
k = a(l)
a(l) = a(n-i+1)
a(n-i+1) = k

NEXT i

'Вывод отсортиров.
'массива
FOR i:=1 TO n
  PRINT a(i)
```



2. Сортировка массива по невозрастанию.

Вход:

1) 1 целое положительное число n — длина массива a;

2) n чисел, организованных в виде массива a.

Выход: n отсортированных по невозрастанию чисел массива a.

Идея алгоритма. Аналогично предыдущей сортировке по неубыванию, только максимум всего массива помещается не в конец массива, а в начало. И т. д.

Итак, алгоритм будет состоять из трех модулей:

1) ввод n элементов массива a и его длины n;

2) сортировка n элементов массива a по невозрастанию;

3) вывод n элементов массива a.

Совсем неочевидно, как работает этот алгоритм, поэтому очень полезно протестировать его на самых разнообразных тестах.

По крайней мере, необходимо воспользоваться готовыми честными тестами из § 1:

если $n = 3$, $a = 3, 1, 2$, то $a = 3, 2, 1$;

если $n = 3$, $a = 2{,}5, 2{,}0, 1{,}5$, то $a = 2{,}5, 2{,}0, 1{,}5$;

если $n = 3$, $a = 1, 2, 4$, то $a = 4, 2, 1$.

В рассмотренных алгоритмах сортировки был использован поиск максимума. В алгоритмах сортировки также можно использовать поиск минимума. Эти алгоритмы оставляем на разработку читателя.

В рассмотренных алгоритмах сортировки были использованы циклы, в которых переменные цикла возрастали. Иногда удобнее использовать циклы, в которых переменные цикла убывают. Или даже в одном цикле убывают, а во втором возрастают, или наоборот. Разработку подобных алгоритмов оставляем читателю.

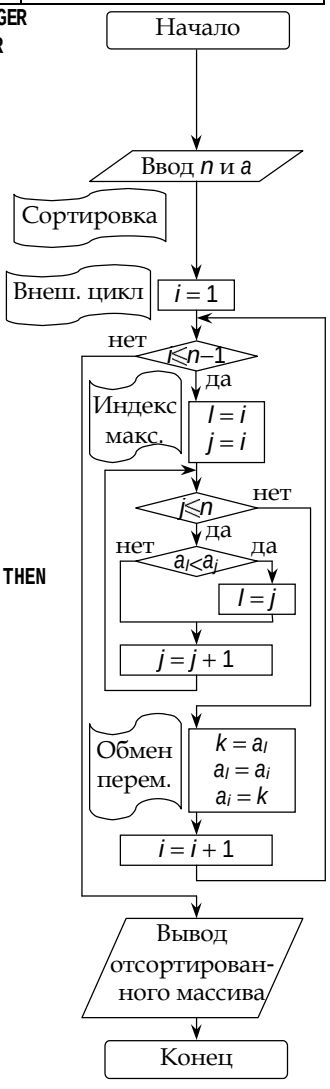
Таблица 42

Сортировка массива по невозрастанию

Паскаль	Бейсик	Блок-схема

```
VAR n, i, j, k: INTEGER;
    a: ARRAY [1..100]
       OF INTEGER;

BEGIN
{Ввод n и массива a}
 n := 3;
 a[1] := 1;
 a[2] := 3;
 a[3] := 2;

{Сортировка}
{Внешний цикл}
 FOR i:=1 TO n-1 DO

{Поиск индекса макс.}
{в части массива}
  l := i;
  FOR j=i TO n DO
  BEGIN
   IF (a[l] < a[j]) THEN
      l := j;

  END;

{Обмен переменных}
  k := a[l];
  a[l] := a[i];
  a(i) := k;

 END;

{Вывод отсортиров.}
{массива}
  FOR i:=1 TO n DO
    WRITELN (a[i]);

END.
```

```
DIM n, i, j, k AS INTEGER
DIM a(100) AS INTEGER



'Ввод n и массива a
n = 3
a(1) = 1
a(2) = 3
a(3) = 2

'Сортировка
'Внешний цикл
FOR i=1 TO n-1

'Поиск индекса макс.
'в части массива
  l = i
  FOR j=i TO n

    IF (a(l) < a(j)) THEN
      l = j

  NEXT j

'Обмен переменных
k = a(l)
a(l) = a(i)
a(i) = k

NEXT i

'Вывод отсортиров.
'массива
FOR i:=1 TO n
  PRINT a(i)
```



2. Алгоритмы

Алгоритм 1. Проектирование алгоритма сверху вниз.

1. Исходная задача представляется в виде одного шага алгоритма.

2. Этот шаг заменяют несколькими более простыми шагами.

3. Те шаги, которые готовы для выполнения исполнителем, оставляются. Остальные заменяют несколькими более простыми шагами.

4. Шаг 3 повторяется то тех пор, пока все шаги не будут готовы для выполнения исполнителем.

Алгоритм 2. Тестирование алгоритма.

1. Алгоритм выполняется для нескольких наборов входных данных (в идеале — для всех возможных наборов).

2. Разные наборы входных данных могут принципиально отличаться друг от друга, разбиваясь тем самым на классы входных данных. Необходимо выполнить алгоритм для всех классов входных данных, взяв хотя бы по одному набору данных из каждого класса.

3. Если на каком-то наборе входных данных алгоритм выдаст неверные выходные данные, т. е. посчитает не то, что планировалось, тогда говорят, что тестирование прошло успешно и ошибка найдена. В этом случае алгоритм переписывается и ошибка в нем устраняется. После этого необходимо начать тестирование заново.

4. Если ни один тест не выявил ошибки, то говорят, что тестирование окончилось неудачей и алгоритм работает правильно.

3. Задачи

1°. **Поиск ошибок в программе**

Задача 2006.C1

Требовалось написать программу, в которой нужно было проверить, лежит ли число x на числовой оси между числами a и b («между» понимается в строгом смысле, т. е. случай $x = a$ или $x = b$ недопустим). Числа x, a, b являются натуральными, и известно, что a отлично от b (но неизвестно: $a > b$ или $b > a$). Входная информация вводится с клавиатуры, а на выходе должно быть сообщение вида «x между a и b» (если это действительно так), в противном случае никакой выходной информации не выдается.

Программист торопился и написал программу некорректно.

Программа на Паскале	Программа на Бейсике

`VAR a,b,x: integer;`
`  p: integer;`
`BEGIN`
`  readln(a,b,x);`
`  if (a>x) AND (x>b) then`
`  writeln('x между a,b');`
`END.`
	`CLS`
	`INPUT a, b, x`
	`IF (a>x) AND (x>b) THEN`
	`  PRINT "x между a, b"`
	`END`

Последовательно выполните три задания:

1) Приведите пример таких чисел a, b, x, при которых программа работает неправильно.

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

3) Укажите, как можно доработать программу, соблюдая дополнительное условие: доработанная программа не должна использовать логических операций AND или OR.

Задача 2007.С1

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

Программа на Паскале	Программа на Бейсике	Программа на Си

`var a, b, x: real;`
`begin`
`readln(a,b,x);`
`if b = 0 then`
`write('x = 0')`
`else`
`if a = 0 then`
`write('нет решений')`
`else`
`write('x =',-b/a);`
`end.`
	`INPUT a, b, x`
	`IF b = 0 THEN`
	`PRINT "x = 0"`
	`ELSE`
	`IF a = 0 THEN`
	`PRINT "нет решений"`
	`ELSE`
	`PRINT "x=",-b/a`
	`ENDIF`
	`ENDIF`
	`END`
		`void main(void)`
		`{ float a,b,x;`
		`scanf("%f%f%f",`
		`&a,&b,&x);`
		`if (b==0)`
		`printf("x=0");`
		`else`
		`if (a==0)`
		`printf("нет решений");`
		`else`
		`printf("x=%f",-b/a);`
		`}`

Последовательно выполните три задания:

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

2) Укажите, какая часть программы является лишней.

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

Задача 2008.С1

Требовалось написать программу, которая решает уравнение «$a|x| = b$» относительно x для любых чисел a и b, введенных с клавиатуры. Все числа считаются действительными. Программист торопился и написал программу неправильно.

Программа на Паскале	Программа на Бейсике	Программа на Си

var a, b, x: real;
begin
readln(a,b,x);
if a = 0 then
if b = 0 then
write('любое число')
else
write('нет решений')
else
if b = 0 then
write('x = 0')
else
write('x =',b/a,' или
x =',-b/a);
end.
	INPUT a, b, x
	IF a = 0 THEN
	IF b = 0 THEN
	PRINT "любое число"
	ELSE
	PRINT "нет решений"
	ENDIF
	ELSE
	IF b = 0 THEN
	PRINT "x = 0"
	ELSE
	PRINT "x =",b/a," или
	x =",-b/a
	ENDIF
	ENDIF
	END
		void main(void)
		{ float a,b,x;
		scanf("%f%f%f",
		&a,&b,&x);
		if (a==0)
		if (b==0)
		printf("любое число");
		else
		printf("нет решений");
		else
		if (b==0)
		printf("x=0");
		else
		printf("x=%f или x=%f",
		b/a,-b/a);
		}

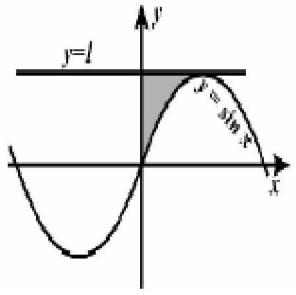
Последовательно выполните три задания:

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

2) Укажите, какая часть программы является лишней.

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

Задача 2009.С1



Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (*x*, *y* — действительные числа) и определяет принадлежность точки заштрихованной области, включая ее границы. Программист торопился и написал программу неправильно.

Программа на Паскале	Программа на Бейсике	Программа на Си

`var x, y: real;`
`begin`
`readln(x,y);`
`if y<=1 then`
`if x>=0 then`
`if y>=sin(x) then`
`write('принадлежит')`
`else`
`write('не принадлежит)`
`end.`
	`INPUT x, y`
	`IF y<=1 THEN`
	`IF x>=0 THEN`
	`IF y>=SIN(x) THEN`
	`PRINT "принадлежит"`
	`ELSE`
	`PRINT "не принадлежит"`
	`ENDIF`
	`ENDIF`
	`ENDIF`
	`END`
		`void main(void)`
		`{ float x,y;`
		`scanf("%f%f", &x,&y);`
		`if (y<=1)`
		`if (x>=0)`
		`if (y>=sin(x))`
		`printf("принадлежит");`
		`else`
		`printf("не принадлежит");`
		`}`

Последовательно выполните следующее:

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

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

2°. **Составление алгоритма**

Задача 2006.С2

Опишите на русском языке или на одном из языков программирования алгоритм поиска второго по величине (т. е. следующего по величине за максимальным) элемента в числовом массиве из 30 различных элементов.

Задача 2007.С2

Опишите на русском языке или одном из языков программирования алгоритм поиска номера первого из двух последовательных элементов в целочисленном массиве из 30 элементов, сумма которых максимальна (если таких пар несколько, то можно выбрать любую из них).

Задача 2008.С2

Опишите на русском языке или одном из языков программирования алгоритм подсчета максимального количества подряд идущих совпадающих элементов в целочисленном массиве длины 30.

Задача 2009.С2

Опишите на русском языке или одном из языков программирования алгоритм получения из заданного целочисленного массива размером 30 элементов другого массива, который будет содержать модули значений элементов первого массива (не используя специальной функции, вычисляющей модуль числа).

3°. **Написание работающей компьютерной программы**

Задача 2006.С4

Вступительные испытания в некоторый вуз состоят из трех экзаменов: математика (максимальный балл — 9), информатика (максимальный балл — 9), литература (максимальный балл — 5). На вход программе подаются сведения о сдаче этих экзаменов абитуриентами. В первой строке вводится количество абитуриентов N, во второй — количество мест K ($K < N$) на которые эти абитуриенты претендуют. Каждая из следующих N строк имеет следующий формат:

<Фамилия> <оценка1> <оценка2> <оценка3>,

где <Фамилия> — строка, состоящая не более, чем из 20 символов, оценки — числа от 0 до максимальной оценки по предмету соответственно. (Ноль ставится в случае, если экзамен не сдавался, например, после полученной на предыдущем экзамене двойки. Все баллы, большие 2, считаются удовлетворительными). Пример входных строк:

Иванов 8 9 3

Петров 2 0 0

Требуется написать программу на языке Паскаль или Бейсик, которая определяла бы по имеющимся данным количество абитуриентов, набравших полупроходной балл в данный вуз или сообщала, что такой балл отсутствует. (Полупроходным называется такой балл, что лишь часть абитуриентов, набравших такой балл и не получивших ни одной неудовлетворительной оценки, попадает в K лучших, которые должны быть зачислены на 1 курс) Считается, что абитуриенты, получившие только удовлетворительные оценки, обязательно присутствуют.

Задача 2007.С4

На вход программе подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке сообщается количество учеников *N*, которое не меньше 10, но не превосходит 100, каждая из следующих *N* строк имеет следующий формат:

<Фамилия> <Имя> <оценки>,

где <Фамилия> — строка, состоящая не более чем из 20 символов, <Имя> — строка, состоящая не более чем из 15 символов, <оценки> — через пробел три целых числа, соответствующие оценкам по пятибалльной системе. <Фамилия> и <Имя>, а также <Имя> и <оценки> разделены одним пробелом. Пример входной строки:

Иванов Петр 4 5 4

Требуется написать программу, которая будет выводить на экран фамилии и имена трех лучших по среднему баллу учеников. Если среди остальных есть ученики, набравшие тот же средний балл, что и один из трех лучших, то следует вывести и их фамилии и имена. Требуемые имена и фамилии можно выводить в произвольном порядке.

Задача 2008.С4

На вход программе подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке сообщается количество учеников *N*, которое не меньше 10, но не превосходит 100, каждая из следующих *N* строк имеет следующий формат:

<Фамилия> <Имя> <оценки>,

где <Фамилия> — строка, состоящая не более чем из 20 символов, <Имя> — строка, состоящая не более чем из 15 символов, <оценки> — через пробел три целых числа, соответствующие оценкам по пятибалльной системе. <Фамилия> и <Имя>, а также <Имя> и <оценки> разделены одним пробелом. Пример входной строки:

Иванов Петр 4 5 3

Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет выводить на экран фамилии и имена трех худших по среднему баллу учеников. Если среди остальных есть ученики, набравшие тот же средний балл, что и один из трех худших, то следует вывести и их фамилии и имена.

Задача 2009.C4

На вход программе подаются сведения о номерах школ учащихся, участвовавших в олимпиаде. В первой строке сообщается количество учащихся *N*, каждая из следующих *N* строк имеет формат:

<Фамилия> <Инициалы> <номер школы>,

где <Фамилия> — строка, состоящая не более чем из 20 символов, <Инициалы> — строка, состоящая из 4 символов (буква, точка, буква, точка), <номер школы> — не более чем двузначный номер. <Фамилия> и <Инициалы>, а также <Инициалы> и <номер школы> разделены одним пробелом. Пример входной строки:

Иванов П.С. 57

Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет выводить на экран информацию, из какой школы было меньше всего участников (таких школ может быть несколько). При этом необходимо вывести информацию только по школам, пославшим хотя бы одного участника.

Следует учитывать, что *N*>=1000.

4. Ответы

1°. Поиск ошибок в программе

Задача 2006.С1. 1) Пример: $a = 1$, $x = 2$, $b = 3$.

2)—3) Продолжением ответа являются две программы, полученные в результате решения.

Задача 2007.С1. 1) $a = 0$, $b = 0$, $x = 0$. (Значение x можно не указывать, допустим ответ, что x — любое число.)

2)—3) Продолжением ответа являются две программы, полученные в результате решения.

Задача 2008.С1. 1) $a = 1$, $b = -1$, $x = 0$.

2)—3) Продолжением ответа являются две программы, полученные в результате решения.

Задача 2009.С1. 1) $x = 3{,}0$, $y = 0{,}5$. (Любая пара (x, y), для которой выполняется: $y > 1$ или $x < 0$ или $(y \geqslant \sin x$ и $x > \pi/2$ и $y \leqslant 1)$.)

2) Продолжением ответа является программа, полученная в результате решения.

2°. Составление алгоритма

Задача 2006.С2. Ответом является алгоритм, полученный при решении.

Задача 2007.С2. Ответом является алгоритм, полученный при решении.

Задача 2008.С2. Ответом является алгоритм, полученный при решении.

Задача 2009.С2. Ответом является алгоритм, полученный при решении.

3°. Написание работающей компьютерной программы

Задача 2006.С4. Ответ — компьютерная программа.

Задача 2007.С4. Ответ — компьютерная программа.

Задача 2008.С4. Ответ — компьютерная программа.

Задача 2009.С4. Ответ — компьютерная программа.

5. Решения

1°. Поиск ошибок в программе

Замечание. В качестве достаточного решения после авторского будет приведено решение, которое рекомендуется составителями заданий.

Рекомендации. 1. Выясняем, на какие классы распадаются входные данные.

2. Разбираемся в сути работы алгоритма.

3. В качестве примера достаточно указать один набор данных, на котором программа работает неправильно. В качестве доработанных программ также достаточно указать один вариант правильной программы.

Задача 2006.С1

Ответ: 1) Пример: $a = 1$, $x = 2$, $b = 3$.

2)—3) Продолжением ответа являются две программы, полученные в результате решения.

Решение 1. Поиск ошибочного решения уже был описан выше в § 2. Осталось решить два оставшихся пункта задания.

Чтобы программа выдавала сообщение о том, что значение x попадает между числами a и b при $a < b$, в ней должно быть правильное условие для выбора if:

$$(a < x) \text{ и } (x < b).$$

А чтобы программа выдавала сообщение при $a > b$, нужно просто оставить старое условие для выбора if. Получаем:

Программа на Паскале	Программа на Бейсике

`VAR a,b,x: INTEGER;`
`  p: INTEGER;`
`BEGIN`
`  READLN(a,b,x);`
`  IF (a>x) AND (x>b) THEN`
`  WRITELN('x между b, a');`
`  IF (a<x) AND (x<b) THEN`
`  WRITELN('x между a, b');`
`END.`
	`CLS`
	`INPUT a, b, x`
	`IF (a>x) AND (x>b) THEN`
	`  PRINT "x между b, a"`
	`IF (a<x) AND (x<b) THEN`
	`  PRINT "x между a, b"`
	`END`

Избавиться от логической операции AND можно даже двумя способами. В ответе следует указать только один из этих способов.

При первом способе один оператор выбора if заменяется на два вложенных, например, так:

Программа на Паскале	Программа на Бейсике

`VAR a,b,x: INTEGER;`
`  p: INTEGER;`
`BEGIN`
`  READLN(a,b,x);`
`  IF (a>x) THEN`
`  IF (x>b) THEN`
`  WRITELN('x между b, a');`
`  IF (a<x) THEN`
`  IF (x<b) THEN`
`  WRITELN ('x между a,b');`
`END.`
	`CLS`
	`INPUT a, b, x`
	`IF (a>x) THEN`
	`  IF (x>b) THEN`
	`  PRINT "x между b, a"`
	`IF (a<x) THEN`
	`  IF (x<b) THEN`
	`  PRINT "x между a, b"`
	`END`

Удостовериться в правильности программы можно при помощи тестирования по всем классам входных данных. В данном случае возможно провести так называемое абсолютное тестирование без выполнения программы. Протестируем только условные операторы, причем математически.

Самое трудное здесь — это перебрать все классы входных данных, другими словами, все математические случаи. Будем обозначать истину буквой И, а ложь — буквой Л. Один вывод в программе должен быть только в двух случаях:

1) $a < x < b$; 2) $b < x < a$.

$a < b$				
$x < a < b$	$x = a < b$	$a < x < b$	$a < b = x$	$a < b < x$
$a > x$ = И, но $x > b$ = Л. Вывода нет.	$a > x$ = Л. Вывода нет.	$a > x$ = Л. Вывода нет.	$a > x$ = Л. Вывода нет.	$a > x$ = Л. Вывода нет.
$a < x$ = Л. Вывода нет.	$a < x$ = Л. Вывода нет.	$a < x$ = И, и $x < b$ = И. *Вывод есть.*	$a < x$ = И, но $x < b$ = Л. Вывода нет.	$a < x$ = И, но $x < b$ = Л. Вывода нет.

$b < a$				
$x < b < a$	$x = b < a$	$b < x < a$	$b < a = x$	$b < a < x$
$a > x$ = И, но $x > b$ = Л. Вывода нет.	$a > x$ = И, но $x > b$ = Л. Вывода нет.	$a > x$ = И, и $x > b$ = И. *Вывод есть.*	$a > x$ = Л. Вывода нет.	$a > x$ = Л. Вывода нет.
$a < x$ = Л. Вывода нет.	$a < x$ = Л. Вывода нет.	$a < x$ = Л. Вывода нет.	$a < x$ = Л. Вывода нет.	$a < x$ = И, но $x < b$ = Л. Вывода нет.

$a = b$		
$x < a = b$	$a = x = b$	$a = b < x$
$a > x$ = И, но $x > b$ = Л. Вывода нет.	$a > x$ = Л. Вывода нет.	$a > x$ = Л. Вывода нет.
$a < x$ = Л. Вывода нет.	$a < x$ = Л. Вывода нет.	$a < x$ = И, но $x < b$ = Л. Вывода нет.

Второй способ математический. Нужно заменить два условия двух операторов выбора

$$(a < x) \text{ и } (x < b);\ (a > x) \text{ и } (x > b)$$

на два математических выражения, а желательно на одно. Стандартный способ — это заменить конъюнкцию AND на арифметическое умножение, например, так:

$$0 < (a - x)(x - b).$$

В этом случае достаточно одного условия выбора!

Программа на Паскале	Программа на Бейсике

`VAR a,b,x: INTEGER;`
`  p: INTEGER;`
`BEGIN`
`  READLN(a,b,x);`
`  if (0<(a-x)*(x-b)) THEN`
`  WRITELN('x между a,b');`
`END.`
	`CLS`
	`INPUT a, b, x`
	`IF (0<(a-x)*(x-b)) THEN`
	`  PRINT "x между a, b"`
	`END`

Удостовериться в правильности программы можно при помощи тестирования по всем классам входных данных.

Снова проведем так называемое абсолютное тестирование без выполнения программы. Протестируем только условные операторы, причем математически.

Один вывод в программе должен быть только в двух случаях:

1) $a < x < b$; 2) $b < x < a$.

$a < b$				
$x < a < b$	$x = a < b$	$a < x < b$	$a < b = x$	$a < b < x$
$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (+)\times(-) \Leftrightarrow$ $0 < (-) =$ Л. Вывода нет.	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < 0\times(-) \Leftrightarrow$ $0 < 0 =$ Л. Вывода нет.	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (-)\times(-) \Leftrightarrow$ $0 < (+) =$ И. *Вывод есть.*	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (-)\times 0 \Leftrightarrow$ $0 < 0 =$ Л. Вывода нет.	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (-)\times(+) \Leftrightarrow$ $0 < (-) =$ Л. Вывода нет.

$b < a$				
$x < b < a$	$x = b < a$	$b < x < a$	$b < a = x$	$b < a < x$
$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (+)\times(-) \Leftrightarrow$ $0 < (-) =$ Л. Вывода нет.	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (+)\times 0 \Leftrightarrow$ $0 < 0 =$ Л. Вывода нет.	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (+)\times(+) \Leftrightarrow$ $0 < (+) =$ И. *Вывод есть.*	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < 0\times(+) \Leftrightarrow$ $0 < 0 =$ Л. Вывода нет.	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (-)\times(+) \Leftrightarrow$ $0 < (-) =$ Л. Вывода нет.

$a = b$		
$x < a = b$	$a = x = b$	$a = b < x$
$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (+)\times(-) \Leftrightarrow$ $0 < (-) =$ Л. Вывода нет.	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < 0\times 0 \Leftrightarrow$ $0 < 0 =$ Л. Вывода нет.	$0 < (a - x)\times$ $\times(x - b) \Leftrightarrow$ $0 < (-)\times(+) \Leftrightarrow$ $0 < (-) =$ Л. Вывода нет.

Решение 2. Приведем решение, которое предлагается представлять на экзамене по ЕГЭ согласно разработчикам заданий, вместе с указаниями по оцениванию.

Содержание верного ответа и указания по оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)	**Баллы**

Элементы ответа:
 	
1) Пример: $a = 1$, $x = 2$, $b = 3$.
 	
2) Возможная доработка:
`if a<b then begin p:=a; a:=b; b:=p end;`
`if (a>x) AND (x>b) then`
`writeln(' x между a,b');`
(могут быть и другие правильные способы доработки).
 	
3) Возможная доработка без использования логических операций AND, OR:
`p:=(x-a)*(x-b); if p<0 then`
`writeln(' x между a,b');`
(могут быть и другие способы доработки с соблюдением дополнительного условия).
При оценке других вариантов доработки программы нужно проверять, что поставленная цель достигается.	

Указания по оцениванию	
Правильно выполнены п. 1 + п. 3 задания (т. к. выполнение п. 3 «покрывает» и пункт 2), или правильно выполнены все 3 пункта задания, при этом в работе (во фрагментах программ) допускается не более одной пунктуационной ошибки.	3
Правильно выполнены 2 пункта задания: 1 + 2 или 2 + 3, (причем способы доработки в п. 2 и п. 3 различны). При этом в сданной работе допускается не более двух синтаксических ошибок (пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования).	2
Правильно выполнен только один пункт задания, при этом, если это был п. 2 или п. 3, то в нем допускается не более двух синтаксических ошибок (пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования).	1
Все пункты задания выполнены неверно.	0
Максимальный балл	3

Задача 2007.С1

Ответ: 1) $a = 0$, $b = 0$, $x = 0$. (Значение x можно не указывать, допустим ответ, что x — любое число.)

2)—3) Продолжением ответа являются две программы, полученные в результате решения.

Решение. Аналогично решению предыдущей задачи.

Также приведем рекомендации разработчиков задания.

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

Элементы ответа:
1) $a = 0$, $b = 0$, $x = 0$. (Значение x можно не указывать, допустим ответ, что x — любое число.)
2) Лишняя часть: не нужно вводить x с клавиатуры
верно: readln(a,b);
3) Возможная доработка:

```
readln(a,b);
if a = 0 then
 if b = 0 then
   write('любое число')
 else
   write('нет решений')
 else
   write('x=',-b / a);
```

(могут быть и другие способы доработки).
При оценке других вариантов доработки программы нужно проверять, что поставленная цель достигается.

Указания по оцениванию	**Баллы**
Правильно выполнены все 3 пункта задания, при этом в работе (во фрагментах программ) допускается не более одной синтаксической ошибки.	3
Правильно выполнены 2 пункта задания. При этом в сданной работе допускается не более двух синтаксических ошибок (пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования).	2
Правильно выполнен только один пункт задания, при этом если это был п. 3, то в нем допускается не более трех синтаксических ошибок (пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования).	1
Все пункты задания выполнены неверно.	0
Максимальный балл	3

Задача 2008.С1

Ответ: 1) $a = 1$, $b = -1$, $x = 0$. 2)—3) Продолжением ответа являются две программы, полученные в результате решения.

Решение. Аналогично решению предыдущей задачи.

Также приведем рекомендации разработчиков задания.

Содержание верного ответа и указания к оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

Элементы ответа:
1) $a = 1$, $b = -1$, $x = 0$. (Значение x может быть не указано. Значения a и b могут быть любыми ненулевыми числами с разными знаками. Также допустим ответ, что программа работает неправильно при любых ненулевых a и b, имеющих разные знаки.)
2) Лишняя часть: не нужно вводить x с клавиатуры
верно: readln(a,b);
3) Возможная доработка:

```
readln(a,b);
if a = 0 then
  if b = 0 then write('любое число')
  else write('нет решений')
else
  if b / a > 0 then
    write('x=',b / a, ' или x=',-b / a)
  else
    if b=0 then write('x=0')
    else write('нет решений');
```

(могут быть и другие способы доработки).
При оценке других вариантов доработки программы нужно проверять, что поставленная цель достигается.

Указания по оцениванию	**Баллы**
Правильно выполнены все 3 пункта задания, при этом в работе (во фрагментах программ) допускается не более одной синтаксической ошибки.	3
Правильно выполнены 2 пункта задания. При этом в сданной работе допускается не более двух синтаксических ошибок (пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования).	2
Правильно выполнен только один пункт задания, при этом если это был п. 3, то в нем допускается не более трех синтаксических ошибок (пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования).	1
Все пункты задания выполнены неверно.	0
Максимальный балл	3

Задача 2009.С1

Ответ: 1) $x = 3{,}0$, $y = 0{,}5$. (Любая пара (x, y), для которой выполняется: $y > 1$ или $x < 0$ или $(y \geqslant \sin x$ и $x > \pi/2$ и $y \leqslant 1)$.)

2) Продолжением ответа является программа, полученная в результате решения.

Решение. Аналогично решению предыдущей задачи.

Приведем решение, которое предлагается представлять на экзамене по ЕГЭ согласно разработчикам заданий, вместе с указаниями по оцениванию.

Содержание верного ответа и указания по оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)	

Элементы ответа:
 	
1) Пример: $x = 3{,}0$, $y = 0{,}5$. (Любая пара (x, y), для которой выполняется: $y > 1$ или $x < 0$ или $(y \geqslant \sin x$ и $x > \pi/2$ и $y \leqslant 1)$.)
 	
2) Возможная доработка (Паскаль):
`if (y<=1) and (x>=0) and (y>=sin(x)) and (x<=3,14/2) then`
`  write('принадлежит')`
`else`
`  write('не принадлежит')`
(могут быть и другие способы доработки).	

Указания по оцениванию	**Баллы**
Обратите внимание! В задаче требовалось выполнить **три** действия: указать пример входных данных, при которых программа работает неверно, и исправить две ошибки. 1. Неправильное использование условного оператора, в результате чего при невыполнении первого или второго условия программа не выдавала ничего (отсутствуют случаи ELSE). 2. Приведенным трем ограничениям удовлетворяют также те точки плоскости, у которых $(y \geqslant \sin x$ и $x > \pi/2$ и $y \leqslant 1)$.	
Правильно выполнены оба пункта задания. Исправлены обе ошибки. Допускается замена числа π на 3,14 или другую константу. В работе (во фрагментах программ) допускается не более одной синтаксической ошибки.	3

Указания по оцениванию	**Баллы**

Правильно выполнены 2 пункта задания из трех (исправлены обе ошибки, но не указан/неправильно указан пример требуемых входных данных, либо правильно указан пример входных данных, программа правильно работает при большем числе случаев, чем исходная, но не при всех).
Например, выдает «принадлежит» для точек, у которых ($y \geqslant \sin x$ и $x > \pi/2$ и $y \leqslant 1$).
Допускается, например, такое решение:
 	
`if y<=1 then`
`  if x>=0 then`
`    if y>=sin(x) then`
`      write('принадлежит')`
`    else`
`      write('не принадлежит')`
`  else`
`    write('не принадлежит')`
`else`
`  write('не принадлежит')`
 	
При этом в сданной работе допускается не более двух синтаксических ошибок (пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования).
	2

Правильно выполнен только один пункт задания. То есть, только приведен пример входных данных, либо он не приведен, но имеется программа, корректно работающая при большем количестве входных данных, чем исходная. При этом, если приведена программа, то в ней допускается не более трех синтаксических ошибок (пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования).	1
Все пункты задания выполнены неверно (пример входных данных не указан или указан неверно, программа не приведена, либо приведенная программа корректно работает в не большем количестве случаев, чем исходная).	0
Максимальный балл	3

2°. **Составление алгоритма**

Замечание. В качестве достаточного решения после авторского будет приведено решение, которое рекомендуется составителями заданий.

Рекомендации. 1. Выясняем, какими известными алгоритмами и алгоритмическими структурами можно воспользоваться для решения задачи.

2. Лучше всего придумывать алгоритм в виде блок-схемы. Это избавит от ненужных подробностей языков программирования. Если опыта недостаточно, то желательно письменно протестировать полученный алгоритм.

Детализация алгоритма должна быть полная, до стандартных операторов языков программирования. (Например, нельзя просто написать «найдем максимум массива», нужно детально расписать этот поиск.)

3. В ответе записываем блок-схему и сразу переводим ее на русский язык. При переводе описываются алгоритмические структуры и поясняется, что они делают.

Задача 2006.C2

Ответ: ответом является алгоритм, полученный при решении.

Решение 1. Воспользуемся известным алгоритмом поиска индекса максимума массива. Найдем индекс максимума массива. И затем переставим максимум в конец массива!

Затем снова найдем максимум массива, но уже без последнего элемента. Это и будет второй по величине максимум.

Найти второй по величине максимум можно многими другими интересными способами. Эти различные способы оставляются читателю на самостоятельную разработку.

Нарисуем подробную блок-схему описанного алгоритма и переведем ее на русский язык. При переводе опишем циклы алгоритма и диапазоны изменения переменных цикла, а также напишем, зачем они нужны. Далее опишем, что происходит внутри циклов, и что — вне циклов.

Имеется альтернатива: вместо русского языка блок-схему можно перевести на какой-нибудь язык программирования.

Русский язык	Блок-схема

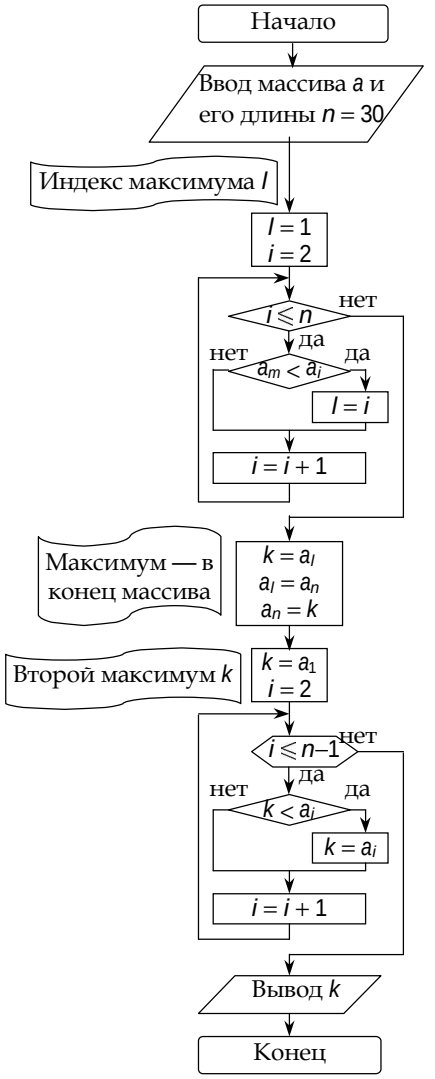
Вводим массив длины 30. Положим переменную n равной 30.

Ищем индекс максимума массива. Для этого примем 1 за первоначальный индекс l максимума массива. Далее в цикле от 2 до n проверяем, не больше ли текущий элемент элемента с индексом l. Если больше, то заменяем значение l на индекс текущего элемента.

Меняем местами максимум массива и его последний элемент.

Ищем максимум массива. Для этого примем значение первого элемента за первоначальный максимум k массива. Далее в цикле от 2 до $n-1$ проверяем, не больше ли текущий элемент максимума k. Если больше, то заменяем максимум k на значение текущего элемента.

Выводим второй по величине максимум k.



Удостовериться в правильности программы можно при помощи тестирования по всем классам входных данных.

В данном случае алгоритм составлен из пяти модулей. Три из них являются алгоритмическими, правильность которых мы проверяли ранее тестированием. И поиск индекса максимума массива, и обмен значениями двух переменных, и поиск величины максимума массива уже были разобраны. Поэтому, если эти три алгоритма записаны так же правильно, как и раньше, то можно сделать вывод, что программа работает, скорее всего, правильно.

Решение 2. Приведем решение, которое предлагается представлять на экзамене по ЕГЭ согласно разработчикам заданий, вместе с указаниями по оцениванию.

Содержание верного ответа и указания по оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)	**Баллы**
Введем числовые переменные *Max1* и *Max2*, в которых будем хранить соответственно максимальный и следующий за максимальным элемент в уже просмотренной части массива. Затем в цикле до конца массива сравниваем очередной элемент массива с двумя максимальными, и если он больше одного из них или обоих, то меняем два отобранных элемента. По окончании цикла переменная *Max2* содержит второй по величине элемент массива.	
Указания по оцениванию	
Предложен правильный алгоритм, выдающий верное значение (в том числе и алгоритм, требующий двукратного прохода по массиву). Возможно использование числа 30 вместо константы. Возможно наличие отдельных синтаксических ошибок (пропущенные «;», неверная запись оператора присваивания и т. п.), не искажающих замысла автора программы. В качестве примера правильного и эффективного алгоритма приведен фрагмент программы:	2

На языке Паскаль	На языке Бейсик	

const N=30;
var a:array[1..N] of real;
Max1, Max2, i: real;
begin
Max1:=a[1];
Max2:=a[1];
if a[2]>Max1 then Max1:=a[2]
else Max2:=a[2];
for i:=3 to N do
begin
if a[i]>Max1 then
begin
Max2:=Max1;
Max1:=a[i];
end
else
if a[i]>Max2 then
Max2:=a[i];
end;
writeln(Max2);
end.
	N=30
	DIM i, Max1, Max2, a(N) AS REAL
	Max1=a(1)
	Max2=a(1)
	IF a(2)>Max1 THEN Max1=a(2)
	ELSE Max2=a(2)
	FOR i = 3 TO N
	IF a(i)>Max1 THEN
	Max2=Max1
	Max1=a(i)
	ELSE
	IF a(i)>Max2 THEN
	Max2=a(i)
	ENDIF
	ENDIF
	NEXT i
	PRINT Max2
	END	

Имеется не более двух ошибок из числа следующих: Не задано или неверно задано первое значение *Max1.* Неверно вычисляется первое значение переменной *Max2.* Не указано условие завершения цикла. Программа не выводит результат. Индексная переменная в цикле не увеличивается (при использовании циклов while или repeat-until). В программе на Паскале неверно расставлены операторные скобки.		1
Ошибок, перечисленных выше, больше двух или алгоритм сформулирован неверно (в частности, не хранится следующий за максимальным элемент).		0
	Максимальный балл	2

Задача 2007.С2

Ответ: Ответом является алгоритм, полученный при решении.

Решение. Аналогично решению предыдущей задачи.

Приведем решение, которое предлагается представлять на экзамене по ЕГЭ согласно разработчикам заданий, вместе с указаниями по оцениванию.

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

Введем целочисленную переменную MaxSum, в которую будем заносить максимальную сумму двух последовательных элементов в просмотренной части массива, и переменную MaxNum, в которую будем заносить номер первого элемента в этой паре. Первоначально в эти переменные занесем сумму первых двух элементов и номер 1 соответственно. В цикле до конца массива: проверяем, превосходит ли сумма элементов очередной пары уже найденный максимум; если да, то заносим в переменную MaxSum новую сумму, а в переменную MaxNum — номер первого элемента пары. По окончании цикла выводим значение переменной MaxNum.

Пример правильной и эффективной программы (на основе алгоритма, использующего однократный проход по массиву):

На языке Паскаль

```
const N=30;
var a:array[1..N] of integer;
MaxSum, MaxNum, i: integer;
begin
  MaxNum:=1;
  MaxSum:=a[1]+a[2];
  for i:=2 to N-1 do
  begin
    if a[i]+a[i+1]>MaxSum then
    begin
      MaxNum:=i;
      MaxSum:=a[i]+a[i+1];
    end
  end;
  writeln(MaxNum);
end.
```

На языке Бейсик

```
N=30
DIM i, MaxSum, MaxNum, a(N) AS
INTEGER
MaxNum=1
MaxSum=a(1)+a(2)
FOR i = 2 TO N-1
  IF a(i)+a(i+1)>MaxSum THEN
    MaxNum=i
    MaxSum=a(i)+a(i+1)
  ENDIF
NEXT i
PRINT MaxNum
END
```

Указания по оцениванию	**Баллы**
Предложен правильный алгоритм, выдающий верное значение (в том числе и алгоритм, требующий двукратного прохода по массиву или создания массива сумм пар). Возможно использование числа 30 вместо константы. Возможно наличие отдельных синтаксических ошибок (пропущенные «;», неверная запись оператора присваивания и т.п.), не искажающих замысла автора программы.	2
Имеется не более двух ошибок из числа следующих: 1) не задано первое значение MaxNum; 2) неверно задается первое значение MaxSum; 3) не указано или неверно указано условие завершения цикла; 4) индексная переменная в цикле не меняется; 5) неверно расставлены операторные скобки.	1
Ошибок, перечисленных выше, больше двух, или алгоритм сформулирован неверно.	0
Максимальный балл	2

Задача 2008.C2

Ответ: Ответ — алгоритм, полученный при решении.

Решение. Аналогично решению предыдущей задачи.

Приведем решение разработчиков заданий.

Содержание верного ответа и указания к оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)
Пример правильного описания алгоритма на русском языке. Заводим переменную MaxCoin для хранения максимального количества подряд идущих совпадающих элементов и счетчик NumCoin для хранения числа элементов в последней группе совпадающих элементов. Просматривая элементы массива, сравниваем очередной элемент со следующим за ним. Если значения совпадают, увеличиваем счетчик NumCoin на единицу. Если очередной элемент массива оказывается не равным предыдущему, то сравниваем текущее значение счетчика со значением переменной MaxCoin; если он больше, то заменяем значение переменной MaxCoin значением счетчика. После сравнения записываем в счетчик NumCoin единицу. Так повторяем до конца массива. В конце работы нужно еще раз сравнить значение счетчика со значением переменной MaxCoin и переопределить ее, если счетчик больше. При оценке других вариантов алгоритма решения необходимо проверить, что поставленная цель достигается. Пример правильной и эффективной программы (на основе алгоритма, использующего однократный проход по массиву):

На языке Паскаль	На языке Бейсик

const N = 30;
var a:array[1..N] of integer;
MaxCoin, NumCoin, i: integer;
begin
  MaxCoin: = 1;
  NumCoin: = 1;
  for i:= 2 to N do
  begin
  if a[i]=a[i-1] then
  NumCoin:=NumCoin+1;
  else begin
  if NumCoin> MaxCoin then
  MaxCoin:=NumCoin;
  NumCoin:=1;
  end;
  end;
  if NumCoin> MaxCoin then
  MaxCoin:= NumCoin;
  writeln(MaxCoin);
end.
	N=30
	DIM i, MaxCoin, NumCoin, a(N) AS INTEGER
	MaxCoin = 1
	NumCoin = 1
	FOR i = 2 TO N
	  IF a(i) = a(i-1) THEN
	  NumCoin=NumCoin+1
	  ELSE
	  IF NumCoin>MaxCoin THEN
	  MaxCoin = NumCoin
	  END IF
	  NumCoin = 1
	  END IF
	  NEXT i
	  IF NumCoin>MaxCoin THEN
	  MaxCoin = NumCoin
	  END IF
	PRINT MaxCoin
	END

Указания по оцениванию	**Баллы**
Предложен правильный алгоритм, выдающий верное значение (в том числе и алгоритм, требующий двукратного прохода по массиву). Возможно использование числа 30 вместо константы. Возможно наличие отдельных синтаксических ошибок (пропущенные «;», неверная запись оператора присваивания и т.п.), не искажающих замысла автора программы.	2
Имеется не более двух ошибок из числа следующих: 1) не задано начальное значение MaxCoin и/или NumCoin; 2) не указано или неверно указано условие завершения цикла; 3) программа не выводит результат; 4) индексная переменная в цикле не увеличивается; 5) в программе на Паскале неверно расставлены операторные скобки.	1
Ошибок, перечисленных выше, больше двух, или алгоритм сформулирован неверно (в частности, переменная NumCoin не приравнивается единице в случае прекращения последовательности одинаковых элементов или нет проверки после завершения цикла в варианте решения, аналогичном предложенному).	0
Максимальный балл	2

Задача 2009.С2

Ответ: Ответ — алгоритм, полученный при решении.

Приведем решение разработчиков заданий.

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

Заводим новый целочисленный массив той же длины. В цикле от первого элемента до последнего сравниваем элементы исходного массива с нулем и отрицательным элементам меняем знак. Записываем значения в элементы второго массива с тем же номером. Печатать значения массива не обязательно.

Пример правильной и эффективной программы (на основе алгоритма, использующего однократный проход по массиву):

На языке Паскаль

```
const N=30;
var a, b:array[1..N] of integer;
i: integer;
begin
  for i:=1 to N do read(a[i]);
    for i:=1 to N do
    if a [i] < 0
    then b[i]:= - a[i]
    else b[i]:= a[i];
end.
```

На языке Бейсик

```
FOR I=1 to N
  INPUT A(I)
NEXT I
N=30
DIM I, A(N), B(N) AS INTEGER
FOR I = 1 TO N
   IF A(I) < 0 THEN
     B(I) = - A(I)
   ELSE
     B(I) = A(I)
   ENDIF
NEXT I
END
```

Указания по оцениванию	**Баллы**
Предложен правильный алгоритм, выдающий верное значение. Возможно использование числа 30 вместо константы. Возможно использование операции «больше» (так как –0 = 0). Возможно использование двух индексных переменных. Возможно наличие отдельных синтаксических ошибок (пропущенные «;», неверная запись оператора присваивания и т. п.), не искажающих замысла автора программы. Алгоритм может не содержать ввода-вывода данных. Алгоритм может не содержать объявления массивов.	2
Имеется не более одной ошибки из числа следующих: 1) не инициализируется или неверно инициализируется индексная переменная в цикле; 2) не указано или неверно указано условие завершения цикла; 3) индексная переменная в цикле не меняется; 4) неверно расставлены операторные скобки.	1
Ошибок, перечисленных выше, две или больше, или алгоритм сформулирован неверно.	0
Максимальный балл	2

3°. **Написание работающей компьютерной программы**

Замечание. Приведено только решение, которое рекомендуется составителями заданий.

Рекомендации. 1. Выясняем, какими известными алгоритмами и алгоритмическими структурами можно воспользоваться для решения задачи.

2. Лучше всего придумывать алгоритм в виде блок-схемы. Это избавит от ненужных подробностей языков программирования. Если опыта недостаточно, то желательно письменно протестировать полученный алгоритм.

3. В ответе записываем перевод блок-схемы на один из языков программирования.

Задача 2006.С4

Ответ: ответ — компьютерная программа.

Решение. Сразу приведем решение, которое предлагается представлять на экзамене по ЕГЭ согласно разработчикам заданий, вместе с указаниями по оцениванию.

Содержание верного ответа и указания по оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)	**Баллы**
Программа верно читает входные данные, не запоминая их все, а сразу подсчитывая в массиве, хранящем 24 целых числа, количество абитуриентов, набравших тот или иной балл (от 0 до 23). Если при этом абитуриент получил хотя бы одну двойку, то удобно считать, что его общий балл равен 0. Затем вычисляется сумма элементов этого массива, начиная с 23-го, до тех пор пока она не превосходит K. Индекс первого элемента массива, который не войдет в эту сумму и будет искомым полупроходным баллом. Если проходной балл набрали ровно K абитуриентов, то программа сообщает, что полупроходной балл отсутствует. Баллы начисляются только за программу, которая решает задачу хотя бы для частного случая (например, проходной балл набрали строго меньше K абитуриентов).	

Указания по оцениванию	Баллы
Программа работает верно, т. е. корректно выделяет из входных данных оценки абитуриентов, верно учитывает результаты абитуриентов, получивших двойки, не содержит вложенных циклов (от 1 до *N* и от 0 до 23). Допускается наличие в тексте программы одной пунктуационной ошибки.	4

Пример правильной и эффективной программы на языке Паскаль:
 	
`var m:array[0..23] of integer;`
`  c:char;`
`  i, K, N, S, m1, m2, m3:integer;`
 	
`begin`
`  readln(N); readln(K);`
`  for i:=0 to 23 do m[i]:=0;`
`  for i:=1 to N do`
`  begin`
 	
`  repeat`
`  read(c)`
`  until c=' '; {считана фамилия абитуриента}`
`  readln(m1, m2, m3);`
`  if (m1<3)or(m2<3)or(m3<3) then s:=0`
`  else s:=m1+m2+m3;`
`  m[s]:=m[s]+1 {учитываем абитуриента в элементе массива, соответствующем его баллам}`
 	
`  end;`
`  s:=m[23]; i:=23;`
`  while s+m[i-1]<=K and`
`  (i>9) {9 - минимально возможный балл} do`
`  begin`
`  i:=i-1;`
`  s:=s+m[i]`
`  end;`
`  if (s<K)and(i>9) then`
`  writeln('полупроходной балл набрали', m[i-1], ' человек')`
`  else writeln('полупроходной балл отсутствует');`
`  readln`
`end.`	

Указания по оцениванию	Баллы

Пример правильной программы на языке Бейсик:
 	
`DIM i, j, k, n, m1, m2, m3, s, m(23) AS INTEGER`
`DIM ss AS STRING`
 	
`FOR i = 1 TO 23`
`  m(i) = 0`
`NEXT i`
`INPUT n`
`INPUT k`
 	
`FOR j = 1 TO n`
`LINE INPUT ss`
`i = 1`
`c$ = MID$(ss, i, 1)`
`WHILE NOT (c$ = " ")`
` i = i + 1`
` c$ = MID$(ss, i, 1)`
`WEND`
`ss = MID$(ss, i + 1, 5)`
`m1 = ASC(MID$(ss, 1, 1)) - ASC("0")`
`m2 = ASC(MID$(ss, 3, 1)) - ASC("0")`
`m3 = ASC(MID$(ss, 5, 1)) - ASC("0")`
`IF (m1 < 3) OR (m2 < 3) OR (m3 < 3) THEN`
`  s = 1`
`  ELSE s = m1 + m2 + m3`
`END IF`
`m(s) = m(s) + 1`
`NEXT j`
`s = m(23): i = 23`
`WHILE (s + m(i - 1) <= k) AND (i > 9)`
` i = i - 1`
` s = s + m(i)`
`WEND`
`IF (s < k) AND (i > 9) THEN`
`PRINT "Полупроходной балл набрали"; m(i - 1);" человек"`
`ELSE PRINT "Полупроходной балл отсутствует"`
`END IF`
`END`	

Указания по оцениванию	Баллы
Программа работает верно, но содержит вложенные циклы (от 0 до 23 и от 1 до *N*) или несколько операторов IF (по количеству возможных баллов у абитуриента) или оператор CASE, обрабатывающий различные варианты количества баллов абитуриента. Возможно, сохраняет все входные данные в массиве абитуриентов. Допускается наличие от одной до трех различных синтаксических ошибок: пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования, не описана или неверно описана переменная, применяется операция, недопустимая для соответствующего типа данных.	3
Программа не учитывает случай, когда ровно *K* абитуриентов набрали проходной балл или что количество абитуриентов, получивших удовлетворительные оценки, может оказаться меньше *K*. Возможно, в реализации алгоритма содержатся 1—2 ошибки (используется знак «<» вместо «<=», «or» вместо «and», выражение на 1 отличается от верного и т. п.). Допускается наличие от одной до пяти различных синтаксических ошибок.	2
Программа неверно работает при некоторых входных данных, возможно, содержит ошибку при учете баллов абитуриентов, получивших неудовлетворительные оценки или в выделении оценок из строки входных данных или в логике определения полупроходного балла. Допускается наличие от одной до семи различных синтаксических ошибок.	1
Задание выполнено неверно.	0
Максимальный балл	4

Задача 2007.С4

Ответ: ответ — компьютерная программа.

Решение. Сразу приведем решение разработчиков ЕГЭ.

Содержание верного ответа и указания по оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)
Программа верно читает входные данные, запоминая фамилии, имена и сумму баллов в массиве записей (или в нескольких массивах), сразу или за дополнительный просмотр подсчитывая три лучших по величине суммы баллов (так как количество экзаменов у всех учеников одинаковое, лучший средний балл соответствует лучшей сумме баллов). Затем за дополнительный просмотр этого массива распечатывается информация о тех учениках, которые набрали в сумме баллов не меньше третьей по величине суммы. Баллы начисляются только за программу, которая решает задачу хотя бы для частного случая (например, все ученики набрали различный средний балл).

Пример правильной и эффективной программы на языке Паскаль:

```
var p:array[1..100] of record
                         name:string;
                         sum:integer;
                       end;
    c:char;
    i,j,N,s1,s2,s3,m:integer;
begin
  readln(N);
  for i:=1 to N do
  begin
    p[i].name:='';
    repeat
      read(c);
      p[i].name:=p[i].name+c
    until c=' '; {считана фамилия}
    repeat
      read(c);
      p[i].name:=p[i].name+c
    until c=' '; {считано имя}
    p[i].sum:=0;
    for j:=1 to 3 do
    begin
      read(m);
      p[i].sum:=p[i].sum+m
    end; {подсчитана сумма баллов}
    readln;
  end;
  s1:=0; s2:=0; s3:=0;
  for i:=1 to N do
  begin
    if p[i].sum>s1 then
      begin
        s3:=s2; s2:=s1;
        s1:=p[i].sum
      end else
    if p[i].sum>s2 then
      begin
        s3:=s2; s2:=p[i].sum
      end else
    if p[i].sum>s3 then s3:=p[i].sum;
  end;
  for i:=1 to N do
    if p[i].sum>=s3 then writeln(p[i].name);
end.
```

Пример правильной программы на языке Бейсик:

```
DIM i, j, n, s1, s2, s3, sum(100) AS INTEGER
DIM s AS STRING
DIM nm(100) AS STRING
INPUT n
FOR j = 1 TO n
LINE INPUT s
c$ = MID$(s, 1, 1)
i = 1
WHILE NOT (c$ = " ")
 i = i + 1
 c$ = MID$(s, i, 1)
WEND
i = i + 1
c$ = MID$(s, i, 1)
WHILE NOT (c$ = " ")
 i = i + 1
 c$ = MID$(s, i, 1)
WEND
nm(j) = MID$(s, 1, i)
sum(j) = ASC(MID$(s, i + 1, 1)) - ASC("0")
sum(j)=sum(j)+(ASC(MID$(s,i+3,1))-ASC("0"))
sum(j)=sum(j)+(ASC(MID$(s,i+5,1))-ASC("0"))
NEXT j
s1 = 0: s2 = 0: s3 = 0
FOR j = 1 TO n
  IF sum(j) > s1 THEN
    s3 = s2: s2 = s1
    s1 = sum(j)
  ELSE
  IF sum(j) > s2 THEN
    s3 = s2: s2 = sum(j)
  ELSE
    IF sum(j) > s3 THEN s3 = sum(j)
  END IF
  END IF
NEXT j
FOR j = 1 TO n
IF sum(j) >= s3 THEN PRINT nm(j)
NEXT j
END
```

Указания по оцениванию	**Баллы**
Программа работает верно, т.е. корректно выделяет из входных данных оценки, ищет три лучших суммы баллов и распечатывает учеников, набравших эти суммы. Допускается наличие в тексте программы одной синтаксической ошибки.	4
Программа работает в целом верно, но содержит по крайней мере две из следующих неточностей (нерациональностей): сохраняются не суммы баллов (средние баллы), а сами баллы и суммы перевычисляются несколько раз заново; явно вычисляются средние баллы, что приводит к сравнению вещественных чисел; при нахождении трех максимальных значений элементы массива переставляются местами; при печати сравнения производятся с каждым из трех максимальных элементов. Допускается наличие от одной до трех синтаксических ошибок: пропущен или неверно указан знак пунктуации, неверно написано или пропущено зарезервированное слово языка программирования, не описана или неверно описана переменная, применяется операция, недопустимая для соответствующего типа данных.	3
Программа работает в целом верно, но выводит только трех лучших учеников, даже если кто-то еще сдал экзамены не хуже. Возможно, в реализации алгоритма содержатся 1–2 ошибки (используется знак “<” вместо “>”, “or” вместо “and” и т.п.). Возможно, некорректно организовано считывание входных данных. Допускается наличие до пяти синтаксических ошибок: пропущен или неверно указан знак пунктуации, неверно написано или пропущено зарезервированное слово языка программирования, не описана или неверно описана переменная, применяется операция, недопустимая для соответствующего типа данных.	2
Программа неверно работает при некоторых входных данных и, возможно, содержит ошибку в алгоритме поиска трех максимальных элементов. Допускается до 4 различных ошибок в ходе решения задачи, в том числе описанных в критериях присвоения двух баллов. Допускается наличие от одной до семи синтаксических ошибок: пропущен или неверно указан знак пунктуации, неверно написано или пропущено зарезервированное слово языка программирования, не описана или неверно описана переменная, применяется операция, недопустимая для соответствующего типа данных.	1
Задание выполнено неверно	0
Максимальный балл	4

Задача 2008.С4

Ответ: ответ — компьютерная программа.

Решение. Сразу приведем решение, которое предлагается представлять на экзамене по ЕГЭ согласно разработчикам заданий, вместе с указаниями по оцениванию.

Содержание верного ответа и указания к оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)
Программа верно читает входные данные, запоминая фамилии, имена и сумму баллов в массиве записей (или в нескольких массивах), сразу или за дополнительный просмотр подсчитывая три худших по величине суммы баллов (так как количество экзаменов у всех учеников одинаковое, лучший средний балл соответствует лучшей сумме баллов). Затем за дополнительный просмотр этого массива распечатывается информация о тех учениках, которые набрали в сумме баллов не больше третьей по величине суммы. Баллы начисляются только за программу, которая решает задачу хотя бы для частного случая (например, все ученики набрали различный средний балл).

Пример правильной и эффективной программы на языке Паскаль:

```
var p:array[1..100] of record
                         name:string;
                         sum:integer;
                       end;
    c:char;
    i,j,N,s1,s2,s3,m:integer;
begin
  readln(N);
  for i:=1 to N do
  begin
    p[i].name:='';
    repeat
      read(c);
      p[i].name:=p[i].name+c
    until c=' '; {считана фамилия}
    repeat
      read(c);
      p[i].name:=p[i].name+
    until c=' '; {считано имя}
    p[i].sum:=0;
    for j:=1 to 3 do
    begin
      read(m);
      p[i].sum:=p[i].sum+m
    end; {подсчитана сумма баллов}
    readln;
  end;
    s1:=20; s2:=20; s3:=20;
  for i:=1 to N do
  begin
    if p[i].sum<s1 then
      begin
        s3:=s2; s2:=s1;
        s1:=p[i].sum
      end else
    if p[i].sum<s2 then
      begin
        s3:=s2; s2:=p[i].sum
      end else
    if p[i].sum<s3 then s3:=p[i].sum;
  end;
  for i:=1 to N do
    if p[i].sum<=s3 then writeln(p[i].name);
end.
```

Пример правильной программы на языке Бейсик:

```
DIM i, j, n, s1, s2, s3, sum(100) AS INTEGER
DIM s AS STRING
DIM nm(100) AS STRING
INPUT n
FOR j = 1 TO n
LINE INPUT s
c$ = MID$(s, 1, 1)
i = 1
WHILE NOT (c$ = " ")
 i = i + 1
 c$ = MID$(s, i, 1)
WEND
i = i + 1
c$ = MID$(s, i, 1)
WHILE NOT (c$ = " ")
 i = i + 1
 c$ = MID$(s, i, 1)
WEND
nm(j) = MID$(s, 1, i)
sum(j) = ASC(MID$(s, i + 1, 1)) - ASC("0")
sum(j)=sum(j)+(ASC(MID$(s,i+3,1))-ASC("0"))
sum(j)=sum(j)+(ASC(MID$(s,i+5,1))-ASC("0"))
NEXT j
s1 = 20: s2 = 20: s3 = 20
FOR j = 1 TO n
  IF sum(j) < s1 THEN
    s3 = s2: s2 = s1
    s1 = sum(j)
  ELSE
  IF sum(j) < s2 THEN
    s3 = s2: s2 = sum(j)
  ELSE
    IF sum(j) < s3 THEN s3 = sum(j)
  END IF
  END IF
NEXT j
FOR j = 1 TO n
IF sum(j) <= s3 THEN PRINT nm(j)
NEXT j
END.
```

Указания по оцениванию	**Баллы**
Программа работает верно, т. е. корректно выделяет из входных данных оценки, ищет три худших суммы баллов и распечатывает учеников, набравших эти суммы. Допускается наличие в тексте программы одной синтаксической ошибки.	4
Программа работает в целом верно, но содержит по крайней мере две из следующих неточностей (нерациональностей): сохраняются не суммы баллов (средние баллы), а сами баллы, и суммы перевычисляются несколько раз заново; явно вычисляются средние баллы, что приводит к сравнению вещественных чисел; при нахождении трех минимальных значений элементы массива переставляются местами; при печати сравнения производятся с каждым из трех минимальных элементов. Допускается наличие от одной до трех синтаксических ошибок: пропущен или неверно указан знак пунктуации, неверно написано или пропущено зарезервированное слово языка программирования, не описана или неверно описана переменная, применяется операция, недопустимая для соответствующего типа данных.	3
Программа работает в целом верно, но выводит только трех худших учеников, даже если кто-то еще сдал экзамены так же. Возможно, в реализации алгоритма содержатся 1—2 ошибки (используется знак «<» вместо «>», «or» вместо «and» и т. п.). Возможно, некорректно организовано считывание входных данных. Допускается наличие до пяти синтаксических ошибок: пропущен или неверно указан знак пунктуации, неверно написано или пропущено зарезервированное слово языка программирования, не описана или неверно описана переменная, применяется операция, недопустимая для соответствующего типа данных.	2
Программа неверно работает при некоторых входных данных и, возможно, содержит ошибку в алгоритме поиска трех минимальных элементов. Допускается до 4 различных ошибок в ходе решения задачи, в том числе описанных в критериях присвоения двух баллов. Допускается наличие от одной до семи синтаксических ошибок: пропущен или неверно указан знак пунктуации, неверно написано или пропущено зарезервированное слово языка программирования, не описана или неверно описана переменная, применяется операция, недопустимая для соответствующего типа данных.	1
Задание выполнено неверно	0
Максимальный балл	4

Задача 2009.С4

Ответ: ответ — компьютерная программа.

Решение. Сразу приведем решение, которое предлагается представлять на экзамене по ЕГЭ согласно разработчикам заданий, вместе с указаниями по оцениванию.

Содержание верного ответа и указания к оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)	
Программа верно читает входные данные, не запоминая их все, а сразу подсчитывая в массиве, хранящем 99 целых чисел согласно номерам школ, количество участников олимпиады из каждой школы. Затем с использованием ненулевых элементов этого массива ищется минимальный элемент, затем распечатываются номера соответствующих школ. Баллы начисляются только за программу, которая решает задачу хотя бы для частного случая.	
Указания по оцениванию	**Баллы**
Программа работает верно, т. е. корректно выделяет из входных данных номера школ, не содержит вложенных циклов, в тексте программы не анализируется каждая школа в отдельности, все считанные номера не запоминаются в массиве. Допускается наличие в тексте программы одной синтаксической ошибки.	4

Указания по оцениванию	**Баллы**

Пример правильной и эффективной программы:
 	
var nc:array[1..99] of integer;
  p:1..99;
  c:char;
  i, k, N, min: integer;
 	
begin
 readln(N);
 for i:=0 to 99 do nc[i]:=0;
 for i:=1 to N do
 begin
 	
  repeat
  read(c)
  until c=' '; {считана фамилия}
 	
  repeat
  read(c)
  until c=' '; {считаны инициалы}
 	
  readln(p);
  nc[p]:=nc[p]+1;
 	
 end;
 min:=N;
 	
 for i:=1 to 99 do
  if nc[i]>0 then
  begin
  if nc[i]<min then min:=nc[i];
  end;
 for i:=1 to 99 do
  if nc[i]=min then
  writeln(i);
 readln
end.
	4

Указания по оцениванию	**Баллы**
Программа работает верно, но содержит вложенные циклы (от 1 до *N* и от 1 до 99) или обрабатывает каждую школу явным образом (99 операторов IF или оператор CASE, содержащий 99 вариантов номеров, в бланке ответа допускаются многоточия), или номера всех школ сохраняются в массиве с последующим поиском наименее часто встречающегося в массиве элемента. Допускается наличие от одной до трех различных синтаксических ошибок: пропущен или неверно указан знак пунктуации, неверно написано зарезервированное слово языка программирования, не описана или неверно описана переменная, применяется операция, недопустимая для соответствующего типа данных.	3
Программа работает в целом верно, но, возможно, некорректно обрабатывает номера школ, ученики которых во входных данных отсутствуют. Возможно, в реализации алгоритма содержатся 1—2 ошибки (используется знак «<» вместо «>», «or» вместо «and», выражение на 1 отличается от верного и т. п.). Допускается наличие от одной до пяти различных синтаксических ошибок.	2
Программа неверно работает при некоторых входных данных и, возможно, содержит ошибку в алгоритме нахождения минимума. Допускается наличие от одной до семи различных синтаксических ошибок.	1
Задание выполнено неверно.	0
Максимальный балл	4

Приложения

§ 1. **Числа**

1. 2-, 3-, 4-, 8-, 10- **и** 16-**ричные числа**

Числа					
2-чные	3-чные	4-ричные	8-ричные	10-ричные	16-ричные
0	0	0	0	0	0
1	1	1	1	1	1
10_2	2	2	2	2	2
11_2	10_3	3	3	3	3
100_2	11_3	10_4	4	4	4
101_2	12_3	11_4	5	5	5
110_2	20_3	12_4	6	6	6
111_2	21_3	13_4	7	7	7
1000_2	22_3	20_4	10_8	8	8
1001_2	$1\ 00_3$	21_4	11_8	9	9
1010_2	$1\ 01_3$	22_4	12_8	10_{10}	A
1011_2	$1\ 02_3$	23_4	13_8	11_{10}	B
1100_2	$1\ 10_3$	30_4	14_8	12_{10}	C
1101_2	$1\ 11_3$	31_4	15_8	13_{10}	D
1110_2	$1\ 12_3$	32_4	16_8	14_{10}	E
1111_2	$1\ 20_3$	33_4	17_8	15_{10}	F
$1\ 0000_2$	$1\ 21_3$	$1\ 00_4$	20_8	16_{10}	10_{16}
$1\ 0001_2$	$1\ 22_3$	$1\ 01_4$	21_8	17_{10}	11_{16}
$1\ 0010_2$	$2\ 00_3$	$1\ 02_4$	22_8	18_{10}	12_{16}
$1\ 0011_2$	$2\ 01_3$	$1\ 03_4$	23_8	19_{10}	13_{16}
$1\ 0100_2$	$2\ 02_3$	$1\ 10_4$	24_8	20_{10}	14_{16}
$1\ 0101_2$	$2\ 10_3$	$1\ 11_4$	25_8	21_{10}	15_{16}
$1\ 0110_2$	$2\ 11_3$	$1\ 12_4$	26_8	22_{10}	16_{16}
$1\ 0111_2$	$2\ 12_3$	$1\ 13_4$	27_8	23_{10}	17_{16}
$1\ 1000_2$	$2\ 20_3$	$1\ 20_4$	30_8	24_{10}	18_{16}
$1\ 1001_2$	$2\ 21_3$	$1\ 21_4$	31_8	25_{10}	19_{16}
$1\ 1010_2$	$2\ 22_3$	$1\ 22_4$	32_8	26_{10}	1A
$1\ 1011_2$	$10\ 00_3$	$1\ 23_4$	33_8	27_{10}	1B
$1\ 1100_2$	$10\ 01_3$	$1\ 30_4$	34_8	28_{10}	1C
$1\ 1101_2$	$10\ 02_3$	$1\ 31_4$	35_8	29_{10}	1D
$1\ 1110_2$	$10\ 10_3$	$1\ 32_4$	36_8	30_{10}	1E
$1\ 1111_2$	$10\ 11_3$	$1\ 33_4$	37_8	31_{10}	1F
$10\ 0000_2$	$10\ 12_3$	$2\ 00_4$	40_8	32_{10}	20_{16}

Числа					
2-чные	3-чные	4-ричные	8-ричные	10-ричные	16-ричные
$10\ 0001_2$	$10\ 20_3$	$2\ 01_4$	41_8	33_{10}	21_{16}
$10\ 0010_2$	$10\ 21_3$	$2\ 02_4$	42_8	34_{10}	22_{16}
$10\ 0011_2$	$10\ 22_3$	$2\ 03_4$	43_8	35_{10}	23_{16}
$10\ 0100_2$	$11\ 00_3$	$2\ 10_4$	44_8	36_{10}	24_{16}
$10\ 0101_2$	$11\ 01_3$	$2\ 11_4$	45_8	37_{10}	25_{16}
$10\ 0110_2$	$11\ 02_3$	$2\ 12_4$	46_8	38_{10}	26_{16}
$10\ 0111_2$	$11\ 10_3$	$2\ 13_4$	47_8	39_{10}	27_{16}
$10\ 1000_2$	$11\ 11_3$	$2\ 20_4$	50_8	40_{10}	28_{16}
$10\ 1001_2$	$11\ 12_3$	$2\ 21_4$	51_8	41_{10}	29_{16}
$10\ 1010_2$	$11\ 20_3$	$2\ 22_4$	52_8	42_{10}	$2A$
$10\ 1011_2$	$11\ 21_3$	$2\ 23_4$	53_8	43_{10}	$2B$
$10\ 1100_2$	$11\ 22_3$	$2\ 30_4$	54_8	44_{10}	$2C$
$10\ 1101_2$	$12\ 00_3$	$2\ 31_4$	55_8	45_{10}	$2D$
$10\ 1110_2$	$12\ 01_3$	$2\ 32_4$	56_8	46_{10}	$2E$
$10\ 1111_2$	$12\ 02_3$	$2\ 33_4$	57_8	47_{10}	$2F$
$11\ 0000_2$	$12\ 10_3$	$3\ 00_4$	60_8	48_{10}	30_{16}
$11\ 0001_2$	$12\ 11_3$	$3\ 01_4$	61_8	49_{10}	31_{16}
$11\ 0010_2$	$12\ 12_3$	$3\ 02_4$	62_8	50_{10}	32_{16}
$11\ 0011_2$	$12\ 20_3$	$3\ 03_4$	63_8	51_{10}	33_{16}
$11\ 0100_2$	$12\ 21_3$	$3\ 10_4$	64_8	52_{10}	34_{16}
$11\ 0101_2$	$12\ 22_3$	$3\ 11_4$	65_8	53_{10}	35_{16}
$11\ 0110_2$	$20\ 00_3$	$3\ 12_4$	66_8	54_{10}	36_{16}
$11\ 0111_2$	$20\ 01_3$	$3\ 13_4$	67_8	55_{10}	37_{16}
$11\ 1000_2$	$20\ 02_3$	$3\ 20_4$	70_8	56_{10}	38_{16}
$11\ 1001_2$	$20\ 10_3$	$3\ 21_4$	71_8	57_{10}	39_{16}
$11\ 1010_2$	$20\ 11_3$	$3\ 22_4$	72_8	58_{10}	$3A$
$11\ 1011_2$	$20\ 12_3$	$3\ 23_4$	73_8	59_{10}	$3B$
$11\ 1100_2$	$20\ 20_3$	$3\ 30_4$	74_8	60_{10}	$3C$
$11\ 1101_2$	$20\ 21_3$	$3\ 31_4$	75_8	61_{10}	$3D$
$11\ 1110_2$	$20\ 22_3$	$3\ 32_4$	76_8	62_{10}	$3E$
$11\ 1111_2$	$21\ 00_3$	$3\ 33_4$	77_8	63_{10}	$3F$
$100\ 0000_2$	$21\ 01_3$	$10\ 00_4$	100_8	64_{10}	40_{16}

Числа					
2-чные	3-чные	4-ричные	8-ричные	10-ричные	16-ричные
$100\ 0001_2$	$21\ 02_3$	$10\ 01_4$	101_8	65_{10}	41_{16}
$100\ 0010_2$	$21\ 10_3$	$10\ 02_4$	102_8	66_{10}	42_{16}
$100\ 0011_2$	$21\ 11_3$	$10\ 03_4$	103_8	67_{10}	43_{16}
$100\ 0100_2$	$21\ 12_3$	$10\ 10_4$	104_8	68_{10}	44_{16}
$100\ 0101_2$	$21\ 20_3$	$10\ 11_4$	105_8	69_{10}	45_{16}
$100\ 0110_2$	$21\ 21_3$	$10\ 12_4$	106_8	70_{10}	46_{16}
$100\ 0111_2$	$21\ 22_3$	$10\ 13_4$	107_8	71_{10}	47_{16}
$100\ 1000_2$	$22\ 00_3$	$10\ 20_4$	110_8	72_{10}	48_{16}
$100\ 1001_2$	$22\ 01_3$	$10\ 21_4$	111_8	73_{10}	49_{16}
$100\ 1010_2$	$22\ 02_3$	$10\ 22_4$	112_8	74_{10}	$4A$
$100\ 1011_2$	$22\ 10_3$	$10\ 23_4$	113_8	75_{10}	$4B$
$100\ 1100_2$	$22\ 11_3$	$10\ 30_4$	114_8	76_{10}	$4C$
$100\ 1101_2$	$22\ 12_3$	$10\ 31_4$	115_8	77_{10}	$4D$
$100\ 1110_2$	$22\ 20_3$	$10\ 32_4$	116_8	78_{10}	$4E$
$100\ 1111_2$	$22\ 21_3$	$10\ 33_4$	117_8	79_{10}	$4F$
$101\ 0000_2$	$22\ 22_3$	$11\ 00_4$	120_8	80_{10}	50_{16}
$101\ 0001_2$	$1\ 00\ 00_3$	$11\ 01_4$	121_8	81_{10}	51_{16}
$101\ 0010_2$	$1\ 00\ 01_3$	$11\ 02_4$	122_8	82_{10}	52_{16}
$101\ 0011_2$	$1\ 00\ 02_3$	$11\ 03_4$	123_8	83_{10}	53_{16}
$101\ 0100_2$	$1\ 00\ 10_3$	$11\ 10_4$	124_8	84_{10}	54_{16}
$101\ 0101_2$	$1\ 00\ 11_3$	$11\ 11_4$	125_8	85_{10}	55_{16}
$101\ 0110_2$	$1\ 00\ 12_3$	$11\ 12_4$	126_8	86_{10}	56_{16}
$101\ 0111_2$	$1\ 00\ 20_3$	$11\ 13_4$	127_8	87_{10}	57_{16}
$101\ 1000_2$	$1\ 00\ 21_3$	$11\ 20_4$	130_8	88_{10}	58_{16}
$101\ 1001_2$	$1\ 00\ 22_3$	$11\ 21_4$	131_8	89_{10}	59_{16}
$101\ 1010_2$	$1\ 01\ 00_3$	$11\ 22_4$	132_8	90_{10}	$5A$
$101\ 1011_2$	$1\ 01\ 01_3$	$11\ 23_4$	133_8	91_{10}	$5B$
$101\ 1100_2$	$1\ 01\ 02_3$	$11\ 30_4$	134_8	92_{10}	$5C$
$101\ 1101_2$	$1\ 01\ 10_3$	$11\ 31_4$	135_8	93_{10}	$5D$
$101\ 1110_2$	$1\ 01\ 11_3$	$11\ 32_4$	136_8	94_{10}	$5E$
$101\ 1111_2$	$1\ 01\ 12_3$	$11\ 33_4$	137_8	95_{10}	$5F$
$1100\ 0000_2$	$1\ 01\ 20_3$	$12\ 00_4$	140_8	96_{10}	60_{16}

Числа					
2-чные	3-чные	4-ричные	8-ричные	10-ричные	16-ричные
$110\ 0001_2$	$1\ 01\ 21_3$	$12\ 01_4$	141_8	97_{10}	61_{16}
$110\ 0010_2$	$1\ 01\ 22_3$	$12\ 02_4$	142_8	98_{10}	62_{16}
$110\ 0011_2$	$1\ 02\ 00_3$	$12\ 03_4$	143_8	99_{10}	63_{16}
$110\ 0100_2$	$1\ 02\ 01_3$	$12\ 10_4$	144_8	100_{10}	64_{16}
$110\ 0101_2$	$1\ 02\ 02_3$	$12\ 11_4$	145_8	101_{10}	65_{16}
$110\ 0110_2$	$1\ 02\ 10_3$	$12\ 12_4$	146_8	102_{10}	66_{16}
$110\ 0111_2$	$1\ 02\ 11_3$	$12\ 13_4$	147_8	103_{10}	67_{16}
$110\ 1000_2$	$1\ 02\ 12_3$	$12\ 20_4$	150_8	104_{10}	68_{16}
$110\ 1001_2$	$1\ 02\ 20_3$	$12\ 21_4$	151_8	105_{10}	69_{16}
$110\ 1010_2$	$1\ 02\ 21_3$	$12\ 22_4$	152_8	106_{10}	$6A_{16}$
$110\ 1011_2$	$1\ 02\ 22_3$	$12\ 23_4$	153_8	107_{10}	$6B_{16}$
$110\ 1100_2$	$1\ 10\ 00_3$	$12\ 30_4$	154_8	108_{10}	$6C_{16}$
$110\ 1101_2$	$1\ 10\ 01_3$	$12\ 31_4$	155_8	109_{10}	$6D_{16}$
$110\ 1110_2$	$1\ 10\ 02_3$	$12\ 32_4$	156_8	110_{10}	$6E_{16}$
$110\ 1111_2$	$1\ 10\ 10_3$	$12\ 33_4$	157_8	111_{10}	$6F_{16}$
$111\ 0000_2$	$1\ 10\ 11_3$	$13\ 00_4$	160_8	112_{10}	70_{16}
$111\ 0001_2$	$1\ 10\ 12_3$	$13\ 01_4$	161_8	113_{10}	71_{16}
$111\ 0010_2$	$1\ 10\ 20_3$	$13\ 02_4$	162_8	114_{10}	72_{16}
$111\ 0011_2$	$1\ 10\ 21_3$	$13\ 03_4$	163_8	115_{10}	73_{16}
$111\ 0100_2$	$1\ 10\ 22_3$	$13\ 10_4$	164_8	116_{10}	74_{16}
$111\ 0101_2$	$1\ 11\ 00_3$	$13\ 11_4$	165_8	117_{10}	75_{16}
$111\ 0110_2$	$1\ 11\ 01_3$	$13\ 12_4$	166_8	118_{10}	76_{16}
$111\ 0111_2$	$1\ 11\ 02_3$	$13\ 13_4$	167_8	119_{10}	77_{16}
$111\ 1000_2$	$1\ 11\ 10_3$	$13\ 20_4$	170_8	120_{10}	78_{16}
$111\ 1001_2$	$1\ 11\ 11_3$	$13\ 21_4$	171_8	121_{10}	79_{16}
$111\ 1010_2$	$1\ 11\ 12_3$	$13\ 22_4$	172_8	122_{10}	$7A_{16}$
$111\ 1011_2$	$1\ 11\ 20_3$	$13\ 23_4$	173_8	123_{10}	$7B_{16}$
$111\ 1100_2$	$1\ 11\ 21_3$	$13\ 30_4$	174_8	124_{10}	$7C_{16}$
$111\ 1101_2$	$1\ 11\ 22_3$	$13\ 31_4$	175_8	125_{10}	$7D_{16}$
$111\ 1110_2$	$1\ 12\ 00_3$	$13\ 32_4$	176_8	126_{10}	$7E_{16}$
$111\ 1111_2$	$1\ 12\ 01_3$	$13\ 33_4$	177_8	127_{10}	$7F_{16}$
$1000\ 0000_2$	$1\ 12\ 02_3$	$20\ 00_4$	200_8	128_{10}	80_{16}

Числа					
2-чные	3-чные	4-ричные	8-ричные	10-ричные	16-ричные
$1000\ 0001_2$	$1\ 12\ 10_3$	$20\ 01_4$	201_8	129_{10}	81_{16}
$1000\ 0010_2$	$1\ 12\ 11_3$	$20\ 02_4$	202_8	130_{10}	82_{16}
$1000\ 0011_2$	$1\ 12\ 12_3$	$20\ 03_4$	203_8	131_{10}	83_{16}
$1000\ 0100_2$	$1\ 12\ 20_3$	$20\ 10_4$	204_8	132_{10}	84_{16}
$1000\ 0101_2$	$1\ 12\ 21_3$	$20\ 11_4$	205_8	133_{10}	85_{16}
$1000\ 0110_2$	$1\ 12\ 22_3$	$20\ 12_4$	206_8	134_{10}	86_{16}
$1000\ 0111_2$	$1\ 20\ 00_3$	$20\ 13_4$	207_8	135_{10}	87_{16}
$1000\ 1000_2$	$1\ 20\ 01_3$	$20\ 20_4$	210_8	136_{10}	88_{16}
$1000\ 1001_2$	$1\ 20\ 02_3$	$20\ 21_4$	211_8	137_{10}	89_{16}
$1000\ 1010_2$	$1\ 20\ 10_3$	$20\ 22_4$	212_8	138_{10}	8A
$1000\ 1011_2$	$1\ 20\ 11_3$	$20\ 23_4$	213_8	139_{10}	8B
$1000\ 1100_2$	$1\ 20\ 12_3$	$20\ 30_4$	214_8	140_{10}	8C
$1000\ 1101_2$	$1\ 20\ 20_3$	$20\ 31_4$	215_8	141_{10}	8D
$1000\ 1110_2$	$1\ 20\ 21_3$	$20\ 32_4$	216_8	142_{10}	8E
$1000\ 1111_2$	$1\ 20\ 22_3$	$20\ 33_4$	217_8	143_{10}	8F
$1001\ 0000_2$	$1\ 21\ 00_3$	$21\ 00_4$	220_8	144_{10}	90_{16}
$1001\ 0001_2$	$1\ 21\ 01_3$	$21\ 01_4$	221_8	145_{10}	91_{16}
$1001\ 0010_2$	$1\ 21\ 02_3$	$21\ 02_4$	222_8	146_{10}	92_{16}
$1001\ 0011_2$	$1\ 21\ 10_3$	$21\ 03_4$	223_8	147_{10}	93_{16}
$1001\ 0100_2$	$1\ 21\ 11_3$	$21\ 10_4$	224_8	148_{10}	94_{16}
$1001\ 0101_2$	$1\ 21\ 12_3$	$21\ 11_4$	225_8	149_{10}	95_{16}
$1001\ 0110_2$	$1\ 21\ 20_3$	$21\ 12_4$	226_8	150_{10}	96_{16}
$1001\ 0111_2$	$1\ 21\ 21_3$	$21\ 13_4$	227_8	151_{10}	97_{16}
$1001\ 1000_2$	$1\ 21\ 22_3$	$21\ 20_4$	230_8	152_{10}	98_{16}
$1001\ 1001_2$	$1\ 22\ 00_3$	$21\ 21_4$	231_8	153_{10}	99_{16}
$1001\ 1010_2$	$1\ 22\ 01_3$	$21\ 22_4$	232_8	154_{10}	9A
$1001\ 1011_2$	$1\ 22\ 02_3$	$21\ 23_4$	233_8	155_{10}	9B
$1001\ 1100_2$	$1\ 22\ 10_3$	$21\ 30_4$	234_8	156_{10}	9C
$1001\ 1101_2$	$1\ 22\ 11_3$	$21\ 31_4$	235_8	157_{10}	9D
$1001\ 1110_2$	$1\ 22\ 12_3$	$21\ 32_4$	236_8	158_{10}	9E
$1001\ 1111_2$	$1\ 22\ 20_3$	$21\ 33_4$	237_8	159_{10}	9F
$1010\ 0000_2$	$1\ 22\ 21_3$	$22\ 00_4$	240_8	160_{10}	$A0_{16}$

Числа					
2-чные	3-чные	4-ричные	8-ричные	10-ричные	16-ричные
$1010\ 0001_2$	$1\ 22\ 22_3$	$22\ 01_4$	241_8	161_{10}	$A1_{16}$
$1010\ 0010_2$	$2\ 00\ 00_3$	$22\ 02_4$	242_8	162_{10}	$A2_{16}$
$1010\ 0011_2$	$2\ 00\ 01_3$	$22\ 03_4$	243_8	163_{10}	$A3_{16}$
$1010\ 0100_2$	$2\ 00\ 02_3$	$22\ 10_4$	244_8	164_{10}	$A4_{16}$
$1010\ 0101_2$	$2\ 00\ 10_3$	$22\ 11_4$	245_8	165_{10}	$A5_{16}$
$1010\ 0110_2$	$2\ 00\ 11_3$	$22\ 12_4$	246_8	166_{10}	$A6_{16}$
$1010\ 0111_2$	$2\ 00\ 12_3$	$22\ 13_4$	247_8	167_{10}	$A7_{16}$
$1010\ 1000_2$	$2\ 00\ 20_3$	$22\ 20_4$	250_8	168_{10}	$A8_{16}$
$1010\ 1001_2$	$2\ 00\ 21_3$	$22\ 21_4$	251_8	169_{10}	$A9_{16}$
$1010\ 1010_2$	$2\ 00\ 22_3$	$22\ 22_4$	252_8	170_{10}	AA
$1010\ 1011_2$	$2\ 01\ 00_3$	$22\ 23_4$	253_8	171_{10}	AB
$1010\ 1100_2$	$2\ 01\ 01_3$	$22\ 30_4$	254_8	172_{10}	AC
$1010\ 1101_2$	$2\ 01\ 02_3$	$22\ 31_4$	255_8	173_{10}	AD
$1010\ 1110_2$	$2\ 01\ 10_3$	$22\ 32_4$	256_8	174_{10}	AE
$1010\ 1111_2$	$2\ 01\ 11_3$	$22\ 33_4$	257_8	175_{10}	AF
$1011\ 0000_2$	$2\ 01\ 12_3$	$23\ 00_4$	260_8	176_{10}	$B0_{16}$
$1011\ 0001_2$	$2\ 01\ 20_3$	$23\ 01_4$	261_8	177_{10}	$B1_{16}$
$1011\ 0010_2$	$2\ 01\ 21_3$	$23\ 02_4$	262_8	178_{10}	$B2_{16}$
$1011\ 0011_2$	$2\ 01\ 22_3$	$23\ 03_4$	263_8	179_{10}	$B3_{16}$
$1011\ 0100_2$	$2\ 02\ 00_3$	$23\ 10_4$	264_8	180_{10}	$B4_{16}$
$1011\ 0101_2$	$2\ 02\ 01_3$	$23\ 11_4$	265_8	181_{10}	$B5_{16}$
$1011\ 0110_2$	$2\ 02\ 02_3$	$23\ 12_4$	266_8	182_{10}	$B6_{16}$
$1011\ 0111_2$	$2\ 02\ 10_3$	$23\ 13_4$	267_8	183_{10}	$B7_{16}$
$1011\ 1000_2$	$2\ 02\ 11_3$	$23\ 20_4$	270_8	184_{10}	$B8_{16}$
$1011\ 1001_2$	$2\ 02\ 12_3$	$23\ 21_4$	271_8	185_{10}	$B9_{16}$
$1011\ 1010_2$	$2\ 02\ 20_3$	$23\ 22_4$	272_8	186_{10}	BA
$1011\ 1011_2$	$2\ 02\ 21_3$	$23\ 23_4$	273_8	187_{10}	BB
$1011\ 1100_2$	$2\ 02\ 22_3$	$23\ 30_4$	274_8	188_{10}	BC
$1011\ 1101_2$	$2\ 10\ 00_3$	$23\ 31_4$	275_8	189_{10}	BD
$1011\ 1110_2$	$2\ 10\ 01_3$	$23\ 32_4$	276_8	190_{10}	BE
$1011\ 1111_2$	$2\ 10\ 01_3$	$23\ 33_4$	277_8	191_{10}	BF
$1100\ 0000_2$	$2\ 10\ 10_3$	$30\ 00_4$	300_8	192_{10}	$C0_{16}$

Числа					
2-чные	3-чные	4-ричные	8-ричные	10-ричные	16-ричные
$1100\ 0001_2$	$2\ 10\ 11_3$	$30\ 01_4$	301_8	193_{10}	$C1_{16}$
$1100\ 0010_2$	$2\ 10\ 12_3$	$30\ 02_4$	302_8	194_{10}	$C2_{16}$
$1100\ 0011_2$	$2\ 10\ 20_3$	$30\ 03_4$	303_8	195_{10}	$C3_{16}$
$1100\ 0100_2$	$2\ 10\ 21_3$	$30\ 10_4$	304_8	196_{10}	$C4_{16}$
$1100\ 0101_2$	$2\ 10\ 22_3$	$30\ 11_4$	305_8	197_{10}	$C5_{16}$
$1100\ 0110_2$	$2\ 11\ 00_3$	$30\ 12_4$	306_8	198_{10}	$C6_{16}$
$1100\ 0111_2$	$2\ 11\ 01_3$	$30\ 13_4$	307_8	199_{10}	$C7_{16}$
$1100\ 1000_2$	$2\ 11\ 02_3$	$30\ 20_4$	310_8	200_{10}	$C8_{16}$
$1100\ 1001_2$	$2\ 11\ 10_3$	$30\ 21_4$	311_8	201_{10}	$C9_{16}$
$1100\ 1010_2$	$2\ 11\ 11_3$	$30\ 22_4$	312_8	202_{10}	CA_{16}
$1100\ 1011_2$	$2\ 11\ 12_3$	$30\ 23_4$	313_8	203_{10}	CB_{16}
$1100\ 1100_2$	$2\ 11\ 20_3$	$30\ 30_4$	314_8	204_{10}	CC_{16}
$1100\ 1101_2$	$2\ 11\ 21_3$	$30\ 31_4$	315_8	205_{10}	CD_{16}
$1100\ 1110_2$	$2\ 11\ 22_3$	$30\ 32_4$	316_8	206_{10}	CE_{16}
$1100\ 1111_2$	$2\ 12\ 00_3$	$30\ 33_4$	317_8	207_{10}	CF_{16}
$1101\ 0000_2$	$2\ 12\ 01_3$	$31\ 00_4$	320_8	208_{10}	$D0_{16}$
$1101\ 0001_2$	$2\ 12\ 02_3$	$31\ 01_4$	321_8	209_{10}	$D1_{16}$
$1101\ 0010_2$	$2\ 12\ 10_3$	$31\ 02_4$	322_8	210_{10}	$D2_{16}$
$1101\ 0011_2$	$2\ 12\ 11_3$	$31\ 03_4$	323_8	211_{10}	$D3_{16}$
$1101\ 0100_2$	$2\ 12\ 12_3$	$31\ 10_4$	324_8	212_{10}	$D4_{16}$
$1101\ 0101_2$	$2\ 12\ 20_3$	$31\ 11_4$	325_8	213_{10}	$D5_{16}$
$1101\ 0110_2$	$2\ 12\ 21_3$	$31\ 12_4$	326_8	214_{10}	$D6_{16}$
$1101\ 0111_2$	$2\ 12\ 22_3$	$31\ 13_4$	327_8	215_{10}	$D7_{16}$
$1101\ 1000_2$	$2\ 20\ 00_3$	$31\ 20_4$	330_8	216_{10}	$D8_{16}$
$1101\ 1001_2$	$2\ 20\ 01_3$	$31\ 21_4$	331_8	217_{10}	$D9_{16}$
$1101\ 1010_2$	$2\ 20\ 02_3$	$31\ 22_4$	332_8	218_{10}	DA_{16}
$1101\ 1011_2$	$2\ 20\ 10_3$	$31\ 23_4$	333_8	219_{10}	DB_{16}
$1101\ 1100_2$	$2\ 20\ 11_3$	$31\ 30_4$	334_8	220_{10}	DC_{16}
$1101\ 1101_2$	$2\ 20\ 12_3$	$31\ 31_4$	335_8	221_{10}	DD_{16}
$1101\ 1110_2$	$2\ 20\ 20_3$	$31\ 32_4$	336_8	222_{10}	DE_{16}
$1101\ 1111_2$	$2\ 20\ 21_3$	$31\ 33_4$	337_8	223_{10}	DF_{16}
$1110\ 0000_2$	$2\ 20\ 22_3$	$32\ 00_4$	340_8	224_{10}	$E0_{16}$

Числа					
2-чные	3-чные	4-ричные	8-ричные	10-ричные	16-ричные
$1110\ 0001_2$	$2\ 21\ 00_3$	$32\ 01_4$	341_8	225_{10}	$E1_{16}$
$1110\ 0010_2$	$2\ 21\ 01_3$	$32\ 02_4$	342_8	226_{10}	$E2_{16}$
$1110\ 0011_2$	$2\ 21\ 02_3$	$32\ 03_4$	343_8	227_{10}	$E3_{16}$
$1110\ 0100_2$	$2\ 21\ 10_3$	$32\ 10_4$	344_8	228_{10}	$E4_{16}$
$1110\ 0101_2$	$2\ 21\ 11_3$	$32\ 11_4$	345_8	229_{10}	$E5_{16}$
$1110\ 0110_2$	$2\ 21\ 12_3$	$32\ 12_4$	346_8	230_{10}	$E6_{16}$
$1110\ 0111_2$	$2\ 21\ 20_3$	$32\ 13_4$	347_8	231_{10}	$E7_{16}$
$1110\ 1000_2$	$2\ 21\ 21_3$	$32\ 20_4$	350_8	232_{10}	$E8_{16}$
$1110\ 1001_2$	$2\ 21\ 22_3$	$32\ 21_4$	351_8	233_{10}	$E9_{16}$
$1110\ 1010_2$	$2\ 22\ 00_3$	$32\ 22_4$	352_8	234_{10}	EA
$1110\ 1011_2$	$2\ 22\ 01_3$	$32\ 23_4$	353_8	235_{10}	EB
$1110\ 1100_2$	$2\ 22\ 02_3$	$32\ 30_4$	354_8	236_{10}	EC
$1110\ 1101_2$	$2\ 22\ 10_3$	$32\ 31_4$	355_8	237_{10}	ED
$1110\ 1110_2$	$2\ 22\ 11_3$	$32\ 32_4$	356_8	238_{10}	EE
$1110\ 1111_2$	$2\ 22\ 12_3$	$32\ 33_4$	357_8	239_{10}	EF
$1111\ 0000_2$	$2\ 22\ 20_3$	$33\ 00_4$	360_8	240_{10}	$F0_{16}$
$1111\ 0001_2$	$2\ 22\ 21_3$	$33\ 01_4$	361_8	241_{10}	$F1_{16}$
$1111\ 0010_2$	$2\ 22\ 22_3$	$33\ 02_4$	362_8	242_{10}	$F2_{16}$
$1111\ 0011_2$	$10\ 00\ 00_3$	$33\ 03_4$	363_8	243_{10}	$F3_{16}$
$1111\ 0100_2$	$10\ 00\ 01_3$	$33\ 10_4$	364_8	244_{10}	$F4_{16}$
$1111\ 0101_2$	$10\ 00\ 02_3$	$33\ 11_4$	365_8	245_{10}	$F5_{16}$
$1111\ 0110_2$	$10\ 00\ 10_3$	$33\ 12_4$	366_8	246_{10}	$F6_{16}$
$1111\ 0111_2$	$10\ 00\ 11_3$	$33\ 13_4$	367_8	247_{10}	$F7_{16}$
$1111\ 1000_2$	$10\ 00\ 12_3$	$33\ 20_4$	370_8	248_{10}	$F8_{16}$
$1111\ 1001_2$	$10\ 00\ 20_3$	$33\ 21_4$	371_8	249_{10}	$F9_{16}$
$1111\ 1010_2$	$10\ 00\ 21_3$	$33\ 22_4$	372_8	250_{10}	FA
$1111\ 1011_2$	$10\ 00\ 22_3$	$33\ 23_4$	373_8	251_{10}	FB
$1111\ 1100_2$	$10\ 01\ 00_3$	$33\ 30_4$	374_8	252_{10}	FC
$1111\ 1101_2$	$10\ 01\ 01_3$	$33\ 31_4$	375_8	253_{10}	FD
$1111\ 1110_2$	$10\ 01\ 02_3$	$33\ 32_4$	376_8	254_{10}	FE
$1111\ 1111_2$	$10\ 01\ 10_3$	$33\ 33_4$	377_8	255_{10}	FF
$1\ 0000\ 0000_2$	$10\ 01\ 11_3$	$1\ 00\ 00_4$	400_8	256_{10}	100_{16}

2. Таблица сложения шестнадцатеричных чисел

Все числа в таблице записаны в шестнадцатеричной системе счисления.

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
10	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20

3. Таблица умножения шестнадцатеричных чисел

Все числа в таблице записаны в шестнадцатеричной системе счисления.

×	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E	20
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D	30
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C	40
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B	50
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A	60
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69	70
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78	80
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87	90
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	56	A0
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5	B0
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4	C0
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3	D0
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2	E0
F	0	F	1E	2D	3C	4B	5A	69	78	87	56	A5	B4	C3	D2	E1	F0
10	0	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	100

§ 2. Символы

1. Русский алфавит и внеалфавитные буквы

№	Буква	Название	Транслитерация	№	Буква	Название	Транслитерация
1	А а	а	a	18	Р р	эр	r
2	Б б	бэ	b	19	С с	эс	s
3	В в	вэ	v, w	20	Т т	тэ	t
4	Г г	гэ	g	21	У у	у	u
5	Д д	дэ	d	22	Ф ф	эф	f
6	Е е	е	e, je, ye	23	Х х	ха	h, kh
7	Ё ё	ё (=е в сортировке)	e, jo, yo	24	Ц ц	цэ	c, ts
8	Ж ж	жэ	g, zh	25	Ч ч	че	ch
9	З з	зэ	z	26	Ш ш	ша	sh
10	И и	и	i	27	Щ щ	ща	shch, sch, shh
11	Й й	и краткое	j, y				
12	К к	ка	k	28	Ъ ъ	твердый знак (ст. ер)	', нет
13	Л л	эль	l	29	Ы ы	ы	y
14	М м	эм	m	30	Ь ь	мягкий знак (ст. ерь)	нет, ', 'h
15	Н н	эн	n	31	Э э	э	e
16	О о	о	o	32	Ю ю	ю	ju, yu
17	П п	пэ	p	33	Я я	я	ja, ya

Выделенная транслитерация однозначно передает русские слова.

В текстах могут использоваться буквы, не входящие в алфавит — все десять гласных с ударением.

№	Буква	Название	Альтернативный набор	№	Буква	Название	Альтернативный набор
1	А́ а́	а ударная	**А**, А, А', а', *А*, *а*	18	У́ у́	у ударная	**У**, У, У', у', *У*, *у*
2	Е́ е́	е ударная	**Е**, Е, Е', е', *Е*, *е*	19	Ы́ ы́	ы ударная	**Ы**, Ы, Ы', ы', *Ы*, *ы*
3	Ё́ ё́	ё ударная	**Ё**, Ё, Ё', ё', *Ё*, *ё*	20	Э́ э́	э ударная	**Э**, Э, Э', э', *Э*, *э*
4	И́ и́	и ударная	**И**, И, И', и', *И*, *и*	21	Ю́ ю́	ю ударная	**Ю**, Ю, Ю', ю', *Ю*, *ю*
5	О́ о́	о ударная	**О**, О, О', о', *О*, *о*	22	Я́ я́	я ударная	**Я**, Я, Я', я', *Я*, *я*

2. Современный латинский и английский алфавиты

Приведем русские названия букв современного латинского (не путать с «древнелатинским»!) и английского алфавитов.

Современный латинский алфавит используется в математике и информатике.

№	Буква	Латинское название	Английское название	№	Буква	Латинское название	Английское название
1	A a	а	эй	14	N n	эн	эн
2	B b	бэ	би	15	O o	о	оу
3	C c	цэ	си	16	P p	пэ	пи
4	D d	дэ	ди	17	Q q	ку	кью
5	E e	е	и	18	R r	эр	а
6	F f	эф	эф	19	S s	эс	эс
7	G g	жэ, гэ	джи	20	T t	тэ	ти
8	H h	аш, хэ	эйч	21	U u	у	ю
9	I i	и	ай	22	V v	вэ	ви
10	J j	джи	джей	23	W w	дубль вэ	дабл ю
11	K k	ка	кей	24	X x	икс	экс
12	L l	эль	эл	25	Y y	игрек	уай
13	M m	эм	эм	26	Z z	зет	зед

3. Современный греческий алфавит

Приведем русские названия букв современного греческого (не путать с древнегреческим) алфавита, а также латинские эквиваленты этих греческих букв. Современный греческий алфавит используется в математике и информатике.

Обратите внимание, что три строчные греческие буквы имеют по два начертания.

№	Буква	Русское название	Латинский эквивалент	№	Буква	Русское название	Латинский эквивалент
1	Α α	а́льфа	a	13	Ν ν	ню	n
2	Β β	бе́та	b	14	Ξ ξ	кси	x
3	Γ γ	га́мма	g	15	Ο ο	о́микрон	o
4	Δ δ	де́льта	d	16	Π π	пи	p
5	Ε ε	э́псилон	e	17	Ρ ρ	ро	r
6	Ζ ζ	дзе́та	z	18	Σ σ, ς	си́гма	s
7	Η η	э́та	e	19	Τ τ	та́у	t
8	Θ θ, ϑ	те́та	th	20	Υ υ	и́псилон	y
9	Ι ι	ио́та	i	21	Φ φ, ϕ	фи	ph
10	Κ κ	ка́ппа	c	22	Χ χ	хи	ch
11	Λ λ	ла́мбда	l	23	Ψ ψ	пси	ps
12	Μ μ	мю	м	24	Ω ω	о́мега	o

4. **Все русские знаки препинания из аски-кодов**

№	Код	Знак препинания	Название
1	32		Пробел, интервал
2	33	!	Восклицательный знак
3	34	"	Прямые, или машинописные, кавычки
4	39	'	Апостро́ф
5	40	(	Левая открывающаяся скобка
6	41	)	Правая закрывающаяся скобка
7	44	,	Запятая
8	45	-	Де́фис, знак переноса, «тире»
9	46	.	Точка
10	58	:	Двоеточие
11	59	;	Точка с запятой
12	63	?	Вопросительный знак

5. **Все русские знаки препинания из второй половины кириллической кодовой таблицы** Windows

Символ ударения отсутствует в кириллической кодовой таблице. Его можно найти в западноевропейской таблице с кодом 0180.

№	Код	Знак препинания	Название
1	0130	‚	Запятая, открывающаяся одинарная кавычка
2	0132	„	Левые открывающиеся кавычки «лапки»
3	0133	…	Многоточие
4	0145	‘	Фигурный, или типографский, обратный апостро́ф, закрывающаяся одинарная кавычка
5	0146	’	Фигурный, или типографский, апостро́ф
6	0147	“	Правые закрывающиеся кавычки «лапки»
7	0148	”	Фигурные, или типографские, прямые кавычки
8	0151	—	Тире, диапазон, диалог
9	0171	«	Левые открывающиеся кавычки «елочки»
10	0187	»	Правые закрывающиеся кавычки «елочки»

6. **Все специальные знаки из аски-кодов**

№	Код	Символ	Название
1	35	#	Номер, решетка, хэш-символ, фунт, диез, шарп
2	36	$	Знак денежной единицы, «доллар»
3	37	%	Проценты
4	38	&	Амперса́нд, «и» английское и логическое, коммерческое «et»
5	42	*	Звездочка, умножить
6	43	+	Плюс
7	47	/	Слеш, поделить, наклонная черта вправо, капиталистическая палочка
8	60	<	Меньше чем, меньше, левая открывающаяся угловая скобка
9	61	=	Равно, знак равенства
10	62	>	Больше чем, больше, правая закрывающаяся угловая скобка
11	64	@	«А» коммерческое, коммерческое «at», собака, знак цены, кваква
12	91	[	Левая, открывающаяся квадратная скобка
13	92	\	Обратный слеш, бэкслеш, наклонная черта влево, коммунистическая палочка
14	93	]	Правая, закрывающаяся квадратная скобка
15	94	^	Крыша, шляпка, облегченное ударение, сиркомфле́кс, циркумфле́кс, вставка, возведение в степень, символ вставки
16	95	_	Подчеркивание, горизонтальная черта
17	96	`	Обратный апостро́ф, грав акцент, гравис, тупое или обратное ударение
18	123	{	Левая, открывающаяся фигурная скобка
19	124	|	Вертикальная палочка, вертикальная черта, логическое «или»
20	125	}	Правая, закрывающаяся фигурная скобка
21	126	~	Волна, тильда

7. **Все специальные знаки из второй половины кириллической кодовой таблицы** Windows

№	Код	Символ	Название
1	0134	†	Крестик (используется как знак сноски)
2	0135	‡	Двойной крестик (используется как знак сноски)
3	0136	€	Евро
4	0137	‰	Промилле, тысячные доли, десятые доли процента
5	0139	‹	Фигурная, или типографская, левая открывающаяся угловая скобка
6	0149	•	Круг
7	0150	–	Американская черточка, минус (склеивающий перенос при просмотре непечатаемых символов в Windows)
8	0153	™	Знак торговой марки
9	0155	›	Фигурная, или типографская, правая закрывающаяся угловая скобка
10	0160		Неразрывный пробел
11	0164	¤	Солнышко, общий знак денежной единицы
12	0166	¦	Вертикальная палочка с разрывом
13	0167	§	Параграф
14	0169	©	Знак копирайта
15	0172	¬	Уголок, знак отрицания (мягкий перенос при просмотре непечатаемых символов в Windows)
16	0173	-	Неразрывный дефис
17	0174	®	Знак регистрации
18	0176	°	Градус
19	0177	±	Плюс-минус
20	0181	µ	Мю, знак микро
21	0182	¶	Абзац (абзац при просмотре непечатаемых символов в Windows)
22	0183	·	Точка, средняя точка, умножить
23	0185	№	Номер

§ 3. Экран монитора

1. Гигиенические требования к величине символов на мониторе

Рассмотрим только один, но для современного состояния персональных компьютеров ключевой, аспект гигиенических требований к видеодисплейным терминалам (далее — ВДТ).

Воспользуемся следующим изданием:

> 1. Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы: Санитарные правила и нормы.— М.: Информационно-издательский центр Госкомсанэпиднадзора России, 1996.— 64 с.— ISBN 5-7508-0049-0.

На с. 34 этого издания установлено ограничение на размер пикселя экрана монитора:

> «6. Размер минимального элемента отображения (пикселя) для монохромного ВДТ, мм — 0,3.»

Итак, размер пикселя не должен быть меньше, чем 0,3 мм. Причем это ограничение, по-видимому, не устарело с времени издания «Гигиенических требований…»: символы текста на экране обычно выводятся одним цветом на фоне другого цвета, поэтому текст программных меню и рабочих текстовых полей можно считать монохромным.

Величина буквы на экране зависит, конечно, также и от ее размера в пикселях. На той же странице того же издания говорится, что размер прописных букв и цифр не должен быть меньше, чем 5×7 пикселей.

Однако набор букв алфавита меньшего размера практически невозможно разработать, поэтому это требование для современных прикладных программ выполняется, и даже с маленьким запасом. Символы текста программного меню и текстовых полей удовлетворяют поставленным требованиям.

2. Визуальный размер пикселя на мониторе

Вычислим визуальный размер пикселя на мониторе. Очевидно, что эта величина зависит от следующих параметров:

1) разрешения, которое выставлено на мониторе: чем больше размеры пиксельной матрицы монитора, тем меньше размер пикселя;

2) размера монитора по диагонали: чем больше монитор, тем больше и пиксель.

Расположение на экране пикселей, которые состоят из трех точек трех цветов, напоминает расположение сот. Поэтому сделаем естественное предположение, что разрешение экрана не зависит от направления, вдоль которого оно измеряется.

Воспользуемся для измерения разрешения диагональю матрицы пикселей. Размер диагонали легко посчитать по теореме Пифагора, зная обе смежные стороны. Поскольку стороны матрицы пикселей обычно относятся как 4 : 3, то диагональ и эти стороны будут в пропорции 5 : 4 : 3. Отметим, что физические размеры экрана монитора также соотносятся как 4 : 3.

Получаем следующую таблицу, в которой приведены разрешение матрицы пикселей экрана в dpi и размер пикселя в мм.

Размер в пикселях		Размер монитора по диагонали, дюймы, мм						
матрицы	диагонали	13" 330,2мм	14" 355,6мм	15" 381,0мм	16" 406,4мм	17" 431,8мм	18" 457,2мм	19" 482,6мм
640 × 480	800	62 dpi 0,41 мм	57 dpi 0,44 мм	53 dpi 0,48 мм	50 dpi 0,51 мм	47 dpi 0,54 мм	44 dpi 0,57 мм	42 dpi 0,60 мм
800 × 600	1000	77 dpi 0,33 мм	71 dpi 0,36 мм	67 dpi 0,38 мм	63 dpi 0,41 мм	59 dpi 0,43 мм	56 dpi 0,46 мм	53 dpi 0,48 мм
1024 × 768	1280	98 dpi 0,26 мм	91 dpi 0,28 мм	85 dpi 0,30 мм	80 dpi 0,32 мм	75 dpi 0,34 мм	71 dpi 0,36 мм	67 dpi 0,38 мм
1280 × 960	1600	123 dpi 0,21 мм	114 dpi 0,22 мм	107 dpi 0,24 мм	100 dpi 0,25 мм	94 dpi 0,27 мм	89 dpi 0,29 мм	84 dpi 0,30 мм
1600 × 1200	2000	154 dpi 0,17 мм	143 dpi 0,18 мм	133 dpi 0,19 мм	125 dpi 0,20 мм	118 dpi 0,22 мм	111 dpi 0,23 мм	105 dpi 0,24 мм

3. Рекомендации по разрешению мониторов

Предыдущая таблица позволяет выработать рекомендации по оптимальному разрешению экрана как ЭЛТ-мониторов, так и ЖК-мониторов.

Для ЭЛТ-мониторов следует учесть тот факт, что визуальный, действительный размер по диагонали матрицы пикселей примерно на дюйм меньше паспортного размера монитора по диагонали.

Следующие две таблицы получены из предыдущей выбором минимального размера пикселя, большего 0,3 мм.

Таблица рекомендаций по разрешению матрицы ЭЛТ-монитора.

Размер монитора	Оптимальное разрешение экрана	Размер пикселя
14"	800 × 600	0,33 мм
15"	800 × 600	0,36 мм
17"	1024 × 768	0,32 мм
19"	1024 × 768	0,36 мм

Следует отметить, что с увеличением размера монитора размер пикселя должен немного возрастать, поскольку с увеличением экрана приходится от него отодвигаться, чтобы сохранить угол обзора.

Для ЖК-мониторов визуальный размер по диагонали матрицы пикселей совпадает с паспортным размером монитора по диагонали. Поскольку ЖК-мониторы менее ярки и контрастны, чем ЭЛТ-мониторы, то размер пикселя для первых должен быть больше, чем для вторых.

Таблица рекомендаций по разрешению матрицы ЖК-монитора.

Размер монитора	Оптимальное разрешение экрана	Размер пикселя
14"	800 × 600	0,36 мм
15"	800 × 600	0,38 мм
17"	1024 × 768	0,34 мм
19"	1024 × 768	0,38 мм

4. Критика других рекомендаций по разрешению мониторов

Рассмотрим рекомендации по разрешению мониторов, которые приведены в следующих учебных пособиях для вузов.

1. Информатика: Базовый курс / С. В. Симонович и др.— СПб.: Питер, 2002.— 640 с.— ISBN 5-8046-0134-2.

2. Щипин Ю. К., Телепин А. М., Колков С. В. Информатика для гуманитарных вузов.— М.: Московский гуманитарный университет. Ростов-на-Дону: Изд-во Феникс, 2004.— 224 с.— ISBN 5-222-04753-9.

В этих пособиях приведена одна и та же таблица [2, с. 75; 3, с. 37], которая совпадает со следующей.

Размер монитора	Оптимальное разрешение экрана
14"	640×480
15"	800×600
17"	1024×768
19"	1280×1024

Разберем особенности этой таблицы.

1. В последней строчке не выдержано отношение сторон матрицы экрана 4 : 3. Причина такого решения нигде не объясняется. Возможно, это опечатка.

2. Нет разделения ЭДТ- и ЖК-мониторов. Возможно, неявно предполагается, что эта таблица предназначена сразу для обоих типов мониторов. Для данного диапазона диагоналей, как показывает анализ выше, это действительно так.

3. Из предыдущего анализа следует, что для монитора 14" размеры матрицы экрана занижены, а с диагональю 19" — завышены. Возможно, эта таблица составлена чисто механически: были взяты последовательные значения размеров монитора, которым были поставлены в соответствие последовательные значения разрешения экрана.

Таким образом, данная таблица удовлетворительна, только необходимо заменить разрешение для монитора 19" на 1024×768.

5. **Установка разрешения экрана в** Windows

В Windows имеется, конечно, несколько способов добраться до того окна, где можно изменить размеры матрицы пикселей экрана. Опишем здесь один из них, возможно, самый удобный и понятный.

Поскольку существует несколько действующих версий операционной системы Windows и, кроме того, некоторые пользователи могут существенно поменять оформление окон Windows на своем компьютере, скриншоты (копии экрана) не будут приведены. Алгоритм просмотра и изменения разрешения экрана будет приведен в обычном для этого издания описательном стиле.

Итак, для просмотра и возможного изменения разрешения экрана монитора можно воспользоваться следующим алгоритмом, взятом из русского Windows 98.

1. Щелкаем по свободному месту Рабочего стола Windows правой кнопкой мыши. Появляется контекстное меню.

2. В этом контекстном меню выбираем опцию **Свойства**, появляется окно **Свойства**: **Экран**.

3. В окне **Свойства**: **Экран** щелкаем по закладке **Настройка**.

4. В закладке **Настройка** в области **Область экрана** показано действующее разрешение экрана. Если разрешение изменять не нужно, выходим щелчком по кнопке **Отмена**.

5. Для изменения разрешения передвигаем мышью движок, определяющий разрешение, в нужное положение. Щелчок по кнопке **Отмена** отменит сделанное изменение, а по кнопке **OK** компьютер установит выбранное движком разрешение.

6. Также нужно проследить за глубиной цвета, значение которой показано на той же закладке **Настройка** в области **Цветовая палитра**. Желательно установить значение True Color (24 бита). При отсутствии этого значения можно установить True Color (32 бита).

7. После изменения параметров экрана изучите расположение матрицы изображения относительно рамок экрана. Возможно, придется восстановить правильный размер и положение матрицы изображения. При программном изменении параметров экрана аппаратные параметры расположения матрицы изображения могут измениться.

8. Расположение пиксельной матрицы изображения осуществляется путем изменения аппаратных параметров монитора обычно клавишами управления, расположенными на самом мониторе, и здесь не описываются ввиду разнообразия моделей мониторов.

6. Установка частоты обновления экрана

Важная и гораздо менее известная характеристика монитора, существенно отвечающая за утомляемость глаз пользователя,— это частота обновления экрана монитора (частота кадровой развертки).

Категорически рекомендуется, чтобы частота обновления монитора была не менее 85 Гц.

Установить частоту обновления экрана можно кнопкой **Дополнительно** в закладке **Настройка** в области **Область экрана** (см. Прил. 1.15). После щелчка по кнопке **Дополнительно** необходимо выбрать закладку **Адаптер**.

7. **Названия и sRGB-значения стандартных цветов**

24-битный цвет кодируется 6-значным 16-ричным числом. Первые два разряда слева (1-й байт) задают яркость красного цвета (red), средние два разряда (2-й байт) задают яркость зеленого цвета (green) и последние два разряда (3-й байт) яркость синего цвета (blue).

№	Название цвета		Шестнадцатеричная кодировка R-G-B
	русское	английское	
1	Черный	Black	00 00 00
2	Темно-синий	Navy	00 00 80
3	Синий	Blue	00 00 FF
4	Зеленый	Green	00 80 00
5	Темно-голубой	Teal	00 80 80
6	Салатовый	Lime	00 FF 00
7	Голубой	Aqua	00 FF FF
8	Темно-красный	Maroon	80 00 00
9	Фиолетовый	Purple	80 00 80
10	Оливковый	Olive	80 80 00
11	Серый	Gray	80 80 80
12	Светло-серый	Silver	C0 C0 C0
13	Красный	Red	FF 00 00
14	Светло-фиолетовый	Fuchsia	FF 00 FF
15	Желтый	Yellow	FF FF 00
16	Белый	White	FF FF FF

§ 4. **Операционная система**

1. **Стандартные расширения имен файлов**

№	Расширение	Стандартный тип файла с таким расширением
1	.arj	Сжатый файл в формате ARJ
2	.asm	Листинг на ассемблере (текст без форматирования)
3	.bas	Листинг на Бейсике (текст без форматирования)
4	.bat	Командный файл (текст без форматирования)
5	.bmp	Графический файл
6	.c	Листинг на C (текст без форматирования)
7	.com	Маленький выполняемый файл
8	.cpp	Листинг на C++ (текст без форматирования)
9	.dll	Подгружаемый выполняемый файл
10	.doc	Текст формата Word до Word 2003
11	.docx	Текст формата Word 2007
12	.exe	Запускаемый выполняемый файл
13	.for	Листинг на Фортране (текст без форматирования)
14	.gif	Графический файл (формат с анимацией)
15	.htm	Веб-текст в разметке HTML (текст без форматирования)
16	.jpg	Графический файл (сжатый с потерями формат)
17	.mid	Сгенерированный звук в формате MIDI
18	.mp3	Сжатый звук в формате MPEG 3-го уровня
19	.pas	Листинг на Паскале (текст без форматирования)
20	.pdf	Текст в разметке PDF (текст без форматирования)
21	.ps	Текст в разметке PostScript (текст без форматирования)
22	.rar	Сжатый файл в формате RAR
23	.sys	Файл операционной системы (текст без форматирования)
24	.tex	Текст в разметке TeX (текст без форматирования)
25	.tif	Графический файл (профессиональный формат)
26	.txt	Текст без форматирования
27	.zip	Сжатый файл в формате ZIP

Литература

Основная

Приведенные здесь книги переиздаются, разумеется, каждый год. Приведены ссылки и описания тех из них, которые попали в руки автора.

Это не означает, что книги, изданные в другие годы, хуже или лучше. Пользуйтесь теми изданиями, которые Вам удалось достать.

1. *ЕГЭ 2008. И*нформатика. Федеральный банк экзаменационных материалов / авт.-сост. П. А. Якушкин, С. С. Крылов.— М.: Эксмо, 2008.— 124 с.: ил.— ISBN 978-5-699-23651-0.

Аннотация.

В издании представлены более 100 экзаменационных заданий частей А, В и С. Задания подготовлены официальным разработчиком контрольных измерительных материалов — Федеральным институтом педагогических измерений — и сгруппированы по экзаменационным темам, соответствующим кодификатору ЕГЭ по информатике. По каждой теме предложены рекомендации и комментарии разработчиков заданий ЕГЭ, ко всем заданиям приведены ответы и критерии оценивания.

Книга адресована *выпускникам* средней школы и *абитуриентам* для подготовки к единому государственному экзамену по информатике. *И*здание поможет *учителям* и *репетиторам* организовать эффективную подготовку учащихся к ЕГЭ.

Оглавление.

Предисловие.

Введение.

ЭКЗАМЕНАЦИОННЫЕ ЗАДАНИЯ.

Раздел 1. **Информационные процессы и системы**. Информация и ее кодирование. Алгоритмизация и программирование. Основы логики. Моделирование.

Раздел 2. **Информационные и коммуникационные технологии**. Программные средства информационных и коммуникационных технологий. Обработка графической информации. Обработка информации в электронных таблицах. Хранение, поиск и сортировка информации в базах данных. Телекоммуникационные технологии. Технологии программирования.

КОММЕНТАРИИ.

Раздел 1. **Информационные процессы и системы**. Информация и ее кодирование. Алгоритмизация и программирование. Основы логики. Моделирование.

Раздел 2. **Информационные и коммуникационные технологии**. Общее замечание. Программные средства информационных и коммуникационных технологий. Обработка графической информации. Обработка информации в электронных таблицах. Хранение, поиск и сортировка информации в базах данных. Телекоммуникационные технологии. Технологии программирования.

Ответы.

Приложения. Приложение 1. Статистика выполнения заданий экзаменационной работы по информатике. Приложение 2. Сравнение результатов ЕГЭ по различным предметам в 2005 г., 2006 г., 2007 г.

2. *Единый государственный экзамен 2009*. Информатика. Универсальные материалы для полготовки учащихся / авт.-сост. С. С. Крылов [и др.]; под ред. В. Р. Лещинера; ФИПИ.— М.: Интеллект-Центр, 2009.— 136 с.: ил.— ISBN 978-5-89790-537-9.

Аннотация.

Пособие предназначено для подготовки к сдаче единого государственного экзамена по информатике 2009 г. Содержит общие рекомендации по подготовке к экзамену, тренировочные примеры по каждому типу заданий, тренировочные варианты экзамена с решениями и ответами. Рассмотрены наиболее важные, с точки зрения подготовки к экзамену, теоретические вопросы и даны практические рекомендации по выполнению заданий.

Оглавление.

Предисловие.

Анализ результатов экзамена 2008 **года по информатике**. Информация и ее кодирование. Алгоритмизация и программирование. Основы логики. Моделирование. Информационные технологии. Программирование.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПОДГОТОВКЕ К ЕГЭ 2009.

Общие замечания.

1. **Информационные процессы и системы**. 1.1. Информация и ее кодирование. *Единицы измерения и методы измерения количества информации. Скорость передачи информации. Представление числовой информации. Системы счисления.* 1.2. Алгоритмизация и программирование. *Алгоритмы, виды алгоритмов, описания алгоритмов. Формальное исполнение алгоритма. Основные алгоритмические конструкции: следование, ветвление, цикл. Блок-схемы. Переменные. Работа с массивами.* 1.3. Основы логики. 1.4. Моделирование.

2. **Информационные и коммуникационные технологии**. 2.1. Файловые системы. 2.2. Обработка графической информации. 2.3. Обработка информации в электронных таблицах. *Работа с формулами, абсолютные и относительные ссылки. Диаграммы и графики.* 2.4. Базы данных. 2.5. Телекоммуникационные технологии. *Адресация в сети Интернет. Поиск информации в сети Интернет.* 2.6. Технология программирования.

ТРЕНИРОВОЧНЫЕ ЗАДАНИЯ.

1. **Информационные процессы и системы**. 1.1. Информация и ее кодирование. *Единицы измерения и методы измерения количества информации. Скорость передачи информации. Представление числовой информации. Системы счисления.* 1.2. Алгоритмизация и программирование. *Алгоритмы, виды алгоритмов, описания алгоритмов. Формальное исполнение алгоритма. Основные алгоритмические конструкции: следование, ветвление, цикл. Блок-схемы. Переменные. Работа с массивами.* 1.3. Основы логики. 1.4. Моделирование.

2. **Информационные и коммуникационные технологии**. 2.1. Файловые системы. 2.2. Обработка графической информации. 2.3. Обработка информации в электронных таблицах. *Работа с формулами, абсолютные и относительные ссылки. Диаграммы и графики.* 2.4. Базы данных. 2.5. Телекоммуникационные технологии. *Адресация в сети Интернет. Поиск информации в сети Интернет.* 2.6. Технология программирования.

ТРЕНИРОВОЧНЫЕ ВАРИАНТЫ 2009 Г.

Вариант 1. Часть 1. Часть 2. Часть 3.

Вариант 2. Часть 1. Часть 2. Часть 3.

ОТВЕТЫ И РЕШЕНИЯ.

Ответы к тренировочным заданиям.

Ответы к тренировочным вариантам-2009. Ответы к варианту 1. *Часть 1. Часть 2. Часть 3.* Ответы к варианту 2. *Часть 1. Часть 2. Часть 3.*

3. *Сафронов И. К.* Готовимся к ЕГЭ. Информатика.— СПб.: БХВ-Петербург, 2007.— 255 с.: ил.— ISBN 978-5-94157-966-2.

Аннотация.

В пособии рассматриваются варианты ЕГЭ по информатике за последние два учебных года (2005/2006, 2006/2007) с подробным разбором всех заданий. Для самостоятельной работы предлагаются задания, подобные официальным, и приводятся их решения. Даны требования к знаниям выпускника по информатике и краткие теоретические пояснения к основным разделам учебного курса. Большое внимание уделено алгебре логики, системам счисления, единицам измерения информации, организации информации, алгоритмизации.

Оглавление.

Предисловие.

Часть 1. **Задания с выбором ответа**. 1.1. Единицы измерения информации. Кодирование информации. 1.2. Системы счисления. 1.3. Определение значений переменных после выполнения фрагментов алгоритмов и программ. 1.4. Алгебра логики или булева алгебра. 1.5. Комбинаторика. 1.6. Файловая система. 1.7. Разное.

Часть 2. **Ответ как набор символов**. 2.1. Системы счисления и логическая алгебра. 2.2. Задачи с исполнителями. 2.3. Логические задачи. 2.4. Объем информации. 2.5. Комбинаторика и закономерности. 2.6. Файловая структура в Интернете. 2.7. Поиск в Интернете.

Часть 3. **Самостоятельные задания**. 3.1. Доработка алгоритмов. 3.2. Разработка алгоритма. 3.3. Разработка правильной стратегии. 3.4. Программирование.

Часть 4. **Разбор заданий ЕГЭ**. 4.1. Разбор заданий «ЕГЭ-2006». *Часть А. Задания с выбором ответа. Часть В. Ответ как набор символов. Часть С. Самостоятельные задания.* 4.2. Демо-версия ЕГЭ-2007. *Часть А. Задания с выбором ответа. Часть В. Ответ как набор символов. Часть С. Самостоятельные задания.*

Часть 5. **Решение задач части** 1 **и** 2. 5.1. Единицы измерения информации. Кодирование информации. 5.2. Системы счисления. 5.3. Определение значений переменных после выполнения фрагментов алгоритмов и программ. 5.4. Алгебра логики или булева алгебра.

Заключение.

Приложения. Приложения 1. Пояснения к демонстрационному варианту. Приложение 2. Инструкция по выполнению работы. Приложение 3. Соглашения об обозначениях на экзамене. Приложение 4. Критерии оценки задач части С (задания с развернутым ответом). Приложение 5. Язык поисковых запросов. Приложение 6. Рекомендуемые Интернет-ресурсы по ЕГЭ.

Литература.

4. *ЕГЭ. Официальный информационный портал Единого государственного экзамена* [Электрон. ресурс]. URL: http: / / www.ege.edu.ru.

Оглавление.

Общая информация о ЕГЭ. Основные сведения о ЕГЭ. Краткий словарь ЕГЭ. Расписание ЕГЭ. Правила и процедура проведения ЕГЭ. ЕГЭ и школа. ЕГЭ и вуз. ЕГЭ в субъектах РФ. Вопрос—Ответ.

Экзаменационные материалы ЕГЭ. Типы заданий. Демонстрационные варианты ЕГЭ. Спецификации по предметам. Федеральный банк тестовых заданий. Шкалирование результатов ЕГЭ.

Нормативные документы. Основные нормативные правовые акты по проведению ЕГЭ. Нормативные правовые и организационно-распорядительные акты по проведению ЕГЭ. в 2009 г. Инструктивно-методические документы по проведению ЕГЭ в 2009 г. Архив документов.

Статистика ЕГЭ.

Мероприятия.

5. *Федеральный институт педагогических измерений* [Электрон. ресурс]. URL: http://www.fipi.ru.

Оглавление.

О нас. Направления деятельности. Структура. Публикации. Сотрудничество. Контакты.

Единый государственный экзамен. Контрольные измерительные материалы (КИМ). Открытый сегмент ФБТЗ. Методические письма.

9 **класс**. **Экзамен в новой форме**. Контрольные измерительные материалы. Технология обработки результатов. Методические письма.

Пособия для подготовки. Издания, рекомендованные ФИПИ к использованию в учебном процессе.

Научно-исследовательская работа. Научная деятельность в области педагогических измерений. Отчеты ФИПИ.

Повышение квалификации. Эксперты предметных комиссий регионов.

Пресс-центр. Новости. Пресс-релизы.

Конференции. Конференции. Семинары.

6. *Сайт информационной поддержки Единого государственного экзамена в компьютерной форме* [Электрон. ресурс]. URL: http://www.ege.ru.

Оглавление.

Портал ЕГЭ.

Карта сайта.

Для организаторов.

Для учащихся.

Нормативные документы.

Демоверсии.

Вопросы и ответы.

Контакты.

Дополнительная

1. Любые учебники по информатике.

2. Любые учебники по программированию.

3. *Мациевский С. В., Ишанов С. А.* Теоретическая информатика: учебное пособие.— Калининград: Изд-во РГУ им. И. Канта, 2007.— 528 с.: ил.— ISBN 978-5-88874-778-0.

Аннотация.

Материал книги принципиально ограничен основами теоретической информатики, как ее понимают сегодня большинство авторов.

Книга предназначена для обучения информатике школьников и студентов гуманитарных направлений. Материал полностью покрывает теоретическую часть государственной программы по информатике.

Замечания и предложения направляйте по электронному адресу одного из авторов matsievsky@newmail.ru.

Оглавление.

Часть II. Вычислительная техника.

Глава 1. Персональный компьютер.

§ 1. **Компьютеры и аппаратура**. 1. Классификация компьютеров. *1°. Компьютер, универсальный компьютер. Классификация компьютеров по мощности. 2°. Классификации персональных компьютеров. 3°. Упражнения.* 2. Классификация аппаратуры. *1°. Системный блок и периферия. 2°. Кабели и питание. 3°. Упражнения.*

§ 2. **Управление компьютером**. 1. Клавиатура. *1°. Алфавитно-цифровая клавиатура. 2°. Вспомогательная клавиатура. Набор символов. 3°. Упражнения.* 2. Манипуляторы. *1°. Мышь. Указатель мыши. 2°. Виды мышей и другие манипуляторы. Сенсорное управление. 3°. Упражнения.*

§ 3. **Матричные устройства**. 1. Монитор. *1°. Виды мониторов. 2°. Настройка монитора. 3°. Упражнения.* 2. Сканер и принтер. *1°. Виды сканеров. 2°. Виды принтеров. 3°. Упражнения.*

Глава 2. Память.

§ 1. **Классификация памяти**. 1. Виды памяти и ее расположение. *1°. Виды памяти. 2°. Системная плата и расположение памяти. 3°. Упражнения.* 2. Виды оперативной памяти. *1°. Центральный процессор, его команды и память. 2°. Взаимодействие памяти компьютера. 3°. Упражнения.*

§ 2. **Структура магнитных дисков**. 1. Дорожка и сектор. *1°. Дорожка. 2°. Сектор. 3°. Упражнения.* 2. Кластер и форматирование. *1°. Кластер. 2°. Форматирование. 3°. Упражнения.*

§ 3. **Структура оптических дисков**. 1. CD. *1°. Структура компакт-дисков. 2°. Форматы и виды компакт-дисков. 3°. Упражнения.* 2. DVD, HD-DVD, Blue Ray. FMD. *1°. DVD. 2°. HD-DVD, Blue Ray. FMD. 3°. Упражнения.*

Глава 3. Локальная сеть.

§ 1. **Определение компьютерной сети**. 1. Виды компьютерных сетей. *1°. Передача информации и компьютерные сети. 2°. Локальная и глобальная сети. Он- и оффлайн. 3°. Упражнения.* 2. Функции компьютерной сети. *1°. Специальные функции сети. Обмен данными. 2°. Доступ к ресурсам. 3°. Упражнения.*

§ 2. **Организация локальной сети**. 1. Архитектура и оборудование локальной сети. *1°. Линейная архитектура локальной сети. 2°. Звездообразная архитектура локальной сети. 3°. Упражнения.* 2. Специальные сетевые устройства. *1°. Файл-сервер и локальная станция. 2°. Принт-сервер. 3°. Упражнения.*

§ 3. **Защита данных и аппаратуры**. 1. Логин и пароль. Администратор. *1°. Логин и пароль. 2°. Необходимость логина. Администратор. 3°. Упражнения.* 2. Доступ к данным и аппаратуре. *1°. Доступ к данным. 2°. Доступ к аппаратуре. 3°. Упражнения.*

Глава 4. Глобальная сеть и Интернет.

§ 1. **Глобальная сеть и Интернет**. 1. Организация глобальной сети. *1°. Два уровня глобальной сети. 2°. Адресное пространство. 3°. Упражнения.* 2. Организация Интернета. *1°. Интернет. Доступ. Устойчивость. 2°. IP- и доменные адреса, DNS-сервер. 3°. Упражнения.*

§ 2. **Электронная почта**. 1. Общие сведения об электронной почте. *1°. Электронное письмо и его схема доставки. 2°. Алгоритм работы с электронной почтой. 3°. Упражнения.* 2. Структура электронного письма и его адрес. *1°. Структура электронного письма. 2°. Адрес электронного письма в Интернете. 3°. Упражнения.*

ГЛАВА 4. ВЕБ-ПРОСТРАНСТВО.

§ 1. **Архитектура веб-пространства**. 1. Структура текста. Гипертекст. *1°. Расположение и упорядоченность текста. Оглавление. 2°. Ссылки. Индексы. Гипертекст. Меню. 3°. Упражнения.* 2. Электронная страница, веб-пространство и сайт. *1°. Электронная страница и ее отличие от печатной. 2°. Веб-страница. Гиперссылка. Веб-пространство. Сайт. 3°. Упражнения.*

§ 2. **Поиск и просмотр сайтов**. 1. Браузер. *1°. Открытие веб-страницы. Журнал и избранные страницы. 2°. Искажение веб-страниц: разные браузеры и кодировки. 3°. Упражнения.* 2. Поиск в веб-пространстве. *1°. Локальный и глобальный поиск. 2°. Каталожный и гибридный поиски. 3°. Упражнения.*

§ 3. **Обмен и создание информации**. 1. Глобальная электронная почта. *1°. Необходимость, объем и время жизни почтового ящика. 2°. Почтовая программа. Архив писем. Локальная и веб-почта. 3°. Упражнения.* 2. Создание сайтов. *1°. Файлы, директории и дерево директорий сайта. 2°. Шаги публикации и редактирования сайта в Интернете. 3°. Упражнения.*

ЧАСТЬ IV. СПРАВОЧНАЯ ИНФОРМАЦИЯ.

1. **Числа**. 1.1. Двоичные, 8-ричные и 16-ричные числа от 0 до 256. 1.2. Таблицы сложения и умножения шестнадцатеричных чисел. *1°. Таблица сложения шестнадцатеричных чисел. 2°. Таблица умножения шестнадцатеричных чисел.*

2. **Символы**. 2.1. Основные алфавиты. *1°. Русский алфавит и внеалфавитные буквы. 2°. Современный латинский и английский алфавиты. 3°. Современный греческий алфавит.* 2.2. Искусственные алфавиты. *1°. Алфавит азбуки Морзе. 2°. Алфавит Брайля.* 2.3. Некоторые иностранные алфавиты. *1°. Украинский алфавит – український алфавіт. 2°. Болгарский алфавит. 3°. Сербский алфавит – азбуке Срба. 4°. Польский алфавит – alfabet polski. 5°. Чешский алфавит – česká abeceda. 6°. Хорватский алфавит. 7°. Венгерский алфавит – Magyar ábécé. 8°. Немецкий алфавит. 9°. Французский алфавит. 10°. Шведский алфавит. 11°. Испанский алфавит. 12°. Итальянский алфавит. 13°. Литовский алфавит. 14°. Латышский алфавит.* 2.4. Знаки препинания и специальные знаки. *1°. Все русские знаки препинания из аски-кодов. 2°. Все русские знаки препинания из второй половины кириллической кодовой таблицы Windows. 3°. Все специальные знаки из аски-кодов. 4°. Все специальные знаки из второй половины кириллической кодовой таблицы Windows.* 2.5. Элементы национальных наборов. *1°. Элементы стандартного русского набора. 2°. Элементы иностранного набора.* 2.6. Нестандартный набор. *0°. Причины нестандартного набора. 1°. Набор символов, отсутствующих на раскладке клавиатуры. 2°. Набор одних символов с помощью других. 3°. Набор математических символов.* 2.7. Римские числа. *1°. Сто первых римских чисел. 2°. Единицы, десятки, сотни и тысячи римских чисел.* 2.8. Аски-коды. 2.9. Кириллические кодовые таблицы. *1°. Кодовая таблица win (кодировка Windows 1251). 2°. Кодовая таблица koi8 (кодировка KOI8-R). 3°. Кодовая таблица dos (альтернативная кодировка DOS 866). 4°. Коды псевдографики в кодировках koi8 и dos.* 2.10. Кодовые таблицы Windows (кроме кириллической). *1°. Западноевропейская кодовая таблица. 2°. Центральноевропейская кодовая таблица. 3°. Балтийская кодовая таблица. 4°. Греческая кодовая таблица. 5°. Турецкая кодовая таблица. 6°. Кодовая таблица иврит. 7°. Арабская кодовая таблица.* 2.11. Полная двухбайтовая кодовая таблицы Windows. *0°. Уникод и Windows: однобайтные таблицы и шрифты. 1°. ASCII –*

основная латиница (коды 0000 – 007F). 2°. Latin 1 – латиница-1 (коды 0080 – 00FF). 3°. European Latin – расширенная латиница-A (коды 0100 – 017F). 4°. Extended Latin – расширенная латиница-B (коды 0180 – 024F). 5°. Standard Phonetic – фонетические знаки (коды 0250 – 02AF). 6°. Modifier Letters – символы изменения пробела (коды 02B0 – 02FF). 7°. General Diacritical Marks – диакритические знаки (коды 0300 – 036F). 8°. Greek – греческий основной (коды 0370 – 03CF). 9°. Greek – греческие и коптские символы (коды 03D0 – 03FF). 10°. Cyrillic – кириллица (коды 0400 – 04FF). 11°. Hebrew – иврит расширенный (коды 0590 – 05CF). 12°. Hebrew – иврит основной (коды 05D0 – 05FF). 13°. Arabic – арабский основной (коды 0600 – 065F). 14°. Arabic – арабский расширенный (коды 0660 – 06FF). 15°. Unknown – дополнительная латиница (коды 1E00 – 1EFF). 16°. General Punctuation – знаки пунктуации (коды 2000 – 206F). 17°. Sups & Subs – верхние и нижние индексы (коды 2070 – 209F). 18°. Currency – денежные символы (коды 20A0 – 20CF). 19°. Letterlike Symbols – буквенные символы (коды 2100 – 214F). 20°. Number Forms – числовые символы (коды 2150 – 218F). 21°. Arrows – стрелки (коды 2190 – 21FF). 22°. Mathematical Operators – математические операторы (коды 2200 – 22FF). 23°. Miscellaneous Technical – технические символы (коды 2300 – 23FF). 24°. Control Pictures – значки управляющих кодов (коды 2400 – 243F). 25°. Form and Chart Components – символы рамок (коды 2500 – 257F). 26°. Blocks – символы заполнения (коды 2580 – 259F). 27°. Geometric Shapes – геометрические фигуры (коды 25A0 – 25FF). 28°. Miscellaneous Dingbats – различные значки Dingbats (коды 2600 – 26FF). 29°. User Zone – область личных символов (коды E000 – F8FF). 30°. User Zone – декоративные варианты букв (коды FB00 – FB4F). 31°. User Zone – арабский декоративный A (коды FB50 – FDFF). 32°. User Zone – арабский декоративный B (коды FE70 – FEFF). 33°. User Zone – специальные символы (коды FFF0 – FFFF).

3. **Мультимедиа**. 3.1. Названия и sRGB-значения стандартных цветов. 3.2. Уровень типичных звуков.

4. **Персональный компьютер**. 4.1. Установка разрешения экрана. *1°. Гигиенические требования к величине символов на мониторе. 2°. Визуальный размер пикселя на мониторе. 3°. Рекомендации по разрешению мониторов. 4°. Критика других рекомендаций по разрешению мониторов. 5°. Установка разрешения экрана в Windows.* 4.2. Установка частоты обновления экрана.

5. **Операционная система**. 5.1. Стандартные расширения имен файлов. 5.2. Стандарт имен файлов 8.3.

Литература. Краткий список. Аннотации и оглавления.

4. *Ишанов С. А., Клевцур С. В., Мациевский С. В.* Информатика: учебное пособие.— Калининград: Изд-во РГУ им. И. Канта, 2006.— 500 с.: ил.

Аннотация.

Учебное пособие является записью лекций по информатике, прочитанных поступающим в Российский государственный университет им. И. Канта. Оно достаточно сбалансировано, а его отдельные разделы увязаны между собой.

Учебный материал рассчитан на тех, кто не знаком ни с компьютером, ни с программированием и, безусловно, будет полезен всем школьникам. При работе с этой книгой компьютер не необходим, но желателен.

Две первые части — «Базовые сведения» и «Приложения» — могут быть использованы для получения базовых знаний перед компьютерной практикой, третья часть — «Программирование» — перед программированием на Паскале или любом другом процедурном языке.

Курс может быть использован не только школьниками, готовящимися к вступительному экзамену по информатике. Эти лекции будут полезны как при обучении информатике в школе, так и при преподавании информатики в вузах. Не исключается работа с материалом при самообразовании.

Содержание соответствует «Стандарту среднего (полного) общего образования по информатике и ИКТ» Министерства образования и науки РФ (см. http://www.ed.gov.ru, Общее образование: начальное, основное, среднее).

Замечания и пожелания принимаются по адресу math@kaliningrad.org.

Оглавление.

Учебное издание

Сергей Валентинович Мациевский

ИНФОРМАТИКА КАК РЕШЕНИЕ ЗАДАЧ ЕГЭ

Учебное пособие

Редакция С. В. Мациевского.
Набор и верстка С. В. Мациевского.

Подписано в печать 31.03.2009. Формат $60\times90^{1}/_{16}$.
Бумага для множительных аппаратов. Усл. печ. л. 26.
Уч.-изд. л. 20. Тираж 250 экз. Заказ .

Издательство Российского государственного университета им. Иммануила Канта
236041, г. Калининград, ул. им. Александра Невского, 14.

ДЛЯ ЗАМЕТОК