Предисловие редактора

> Настоящая научная проблема отличается от олимпиадной задачи только тем, что над первой можно думать в тысячу раз больше.
>
> *Б. Н. Делоне*

Одной из главных задач современной системы образования является выявление и воспитание одаренных школьников. Так, в материалах Национальной образовательной инициативы «Наша новая школа», утвержденной Президентом Российской Федерации Д. А. Медведевым 4 февраля 2010 г., сказано[1]:

«Модернизация и инновационное развитие — единственный путь, который позволит России стать конкурентным обществом в мире 21-го века, обеспечить достойную жизнь всем нашим гражданам. В условиях решения этих стратегических задач важнейшими качествами личности становятся инициативность, способность творчески мыслить и находить нестандартные решения, умение выбирать профессиональный путь, готовность обучаться в течение всей жизни. Все эти навыки формируются с детства.

Школа является критически важным элементом в этом процессе. Главные задачи современной школы — раскрытие способностей каждого ученика, воспитание порядочного и патриотичного человека, личности, готовой к жизни в высокотехнологичном, конкурентном мире. Школьное обучение должно быть построено так, чтобы выпускники могли самостоятельно ставить и достигать серьезных целей, умело реагировать на разные жизненные ситуации».

Развитие системы поддержки талантливых детей является одним из основных направлений развития общего образования, заявленных в рамках Национальной образовательной инициативы. И одной из главнейших составляющих этого направления является *олимпиадное движение*.

Олимпиады для одаренных учащихся сегодня организуются во всех городах и регионах нашей страны как школьные, городские, районные, областные, региональные и, наконец, всерос-

[1] См.: Сайт Департамента образования г. Москвы, Национальная образовательная инициатива «НАША НОВАЯ ШКОЛА» (http://www.educom.ru/ru/nasha_novaya_shkola/school.php).

сийские[1]. Кроме того, организуются международные олимпиады по различным учебным предметам, в которых ежегодно принимают участие, наряду с учащимися из других стран мира, также и российские школьники. Например, в XXII Международной олимпиаде по информатике[2] (14—21 августа 2010 г., г. Ватерлоо, Канада) среди делегаций из 81 страны мира участвовали четверо российских одиннадцатиклассников: Михаил Пядеркин (ГОУ школа-интернат среднего (полного) общего образования «Интеллектуал», Москва) — золотая медаль; Сергей Федоров (МОУ «Физико-технический лицей № 1», г. Саратов) — золотая медаль; Роман Андреев и Глеб Евстропов (ГОУ «Специализированный учебно-научный центр — факультет МГУ им. М. В. Ломоносова, школа им. А. Н. Колмогорова», г. Москва) — серебряные медали.

Однако чтобы достигнуть такого высокого уровня, будущие участники — «олимпийцы» — должны приложить немалые усилия, постоянно совершенствуя свои знания и умения по учебному предмету, не ограничиваясь достаточно тесными рамками школьной программы. Оттачивать свое мастерство будущим победителям необходимо начинать еще с 5—6 классов, участвуя в олимпиадах сначала школьного уровня, на которых, как правило, предлагаются сравнительно простые задания, затем — городского уровня и т. д.; соответственно на каждом таком этапе возрастает сложность решаемых задач.

Именно так, по «многоступенчатому» принципу организована Всероссийская олимпиада школьников, которая включает в себя четыре этапа: школьный (организаторами которого являются различные образовательные учреждения), муниципальный (организуемый органами местного самоуправления муниципальных районов и городских округов в сфере образования), региональный (под руководством органов исполнительной власти субъектов Российской Федерации, осуществляющих управление в сфере образования) и, наконец, заключительный всероссийский этап (организатором которого является Федеральное агентство по образованию). При этом в качестве основных целей

[1] Более подробную информацию о Всероссийской олимпиаде школьников по различным учебным дисциплинам можно найти на интернет-портале http://rosolymp.ru (прежняя версия портала доступна по адресу http://old.rosolymp.ru).

[2] См.: Сайт XXII Международной олимпиады по информатике: http://www.ioi2010org.

и задач Всероссийской олимпиады названы «выявление и развитие у обучающихся творческих способностей и интереса к научно-исследовательской деятельности, создание необходимых условий для поддержки одаренных детей, пропаганда научных знаний, привлечение ученых и практиков соответствующих областей к работе с одаренными детьми, отбор наиболее талантливых обучающихся в состав сборных команд Российской Федерации для участия в международных олимпиадах по общеобразовательным предметам»[1].

Следует также отметить, что победители и призеры заключительного этапа Всероссийской олимпиады школьников — выпускников 2011 г. признаются образовательными учреждениями как результаты государственной итоговой аттестации по соответствующим предметам[2], при этом:

- в аттестат по общеобразовательному предмету, соответствующему профилю олимпиады, выставляется отметка «отлично»;
- ЕГЭ по предмету олимпиады сдавать не требуется;
- победители и призеры получают диплом государственного образца, в котором указано название олимпиады по предмету и который нужно предъявить в вуз при подаче документов для подтверждения своих льгот;
- при поступлении в вуз на специальность, соответствующую профилю олимпиады, победители и призеры принимаются без вступительных экзаменов при предъявлении аттестата о полном образовании;
- при поступлении в вуз на специальность не по профилю олимпиады ее результат засчитывается как 100 баллов по предмету, соответствующему профилю олимпиады.

Еще одна возможность льготного поступления в высшее учебное заведение — участие в олимпиадах, проводимых соответствующим вузом, среди которых Московский государственный университет им. М. В. Ломоносова, Московский государственный технический университет им. Н. Э. Баумана, Московский государственный институт международных отношений

---

[1] «Положение о Всероссийской олимпиаде школьников», утверждено Приказом Министерства образования и науки Российской Федерации от 02.12.2009 № 695; http://www.edu.ru/db-mon/mo/Data/d_09/prm695-1.htm.

[2] См. информацию на сайте Официального информационного портала единого государственного экзамена: http://www1.ege.edu.ru/olympics.

(университет) МИД РФ, Московский государственный университет экономики, статистики и информатики и др. Такие олимпиады, как правило, также включают несколько этапов и проводятся в течение всего года.

Ярким примером является ежегодная Всероссийская олимпиада «Шаг в будущее» (организатор — Московский государственный технический университет им. Н. Э. Баумана (МГТУ)), которая является составной частью одноименной с ней российской научно-социальной программы для молодежи и школьников[1]. Ее участники должны не только суметь решить достаточно сложные задачи по математике, информатике, физике, химии, биологии, литературе, истории, обществознанию, а также по комплексам предметов — естественным наукам, технике и технологии, гуманитарным и социальным наукам, но и продемонстрировать собственные научные проекты.

Победителям и призерам олимпиады «Шаг в будущее» при поступлении в МГТУ им. Н. Э. Баумана, в государственные и муниципальные образовательные учреждения среднего профессионального и высшего профессионального образования в течение одного года предоставляется возможность:

- зачисления в образовательное учреждение без экзамена на специальности, соответствующие профилю олимпиады;
- засчитывания результата олимпиады как максимального количества баллов ЕГЭ по предмету, соответствующему профилю олимпиады.

Аналогичные льготы при поступлении в вуз получают победители и призеры открытых олимпиад. Их особенностью является возможность прямого участия в олимпиаде всех желающих школьников (т. е. без предварительной победы в олимпиадах школьного, городского и т. д. уровней, хотя многие открытые олимпиады имеют всероссийский статус), а также дистанционный характер участия по крайней мере в отборочном туре. Так, например, Открытая олимпиада школьников «Информационные технологии»[2], проводимая Санкт-Петербургским государственным университетом информационных технологий, механики и оптики и Национальным исследовательским университетом

[1] См.: Положение об олимпиаде школьников «Шаг в будущее»: http://www.step-into-the-future.ru/stepol_1.php.

[2] См. информацию по адресу http://olymp.info.ru/10-11/inf.

«Высшая школа экономики» для учащихся 7—11 классов, состоит из двух этапов, где для участия в отборочном этапе достаточно зарегистрироваться на сайте, получать доступ к заданиям и отправлять их решения через Интернет, а заключительный этап, в котором участвуют победители отборочного этапа, является очным.

Таким образом, участие в олимпийском движении — это не только возможность проявить себя, доказать высокий уровень своей компетентности в соответствующей предметной области, но и вполне реальный шанс продолжить обучение по окончании школы в престижном вузе и на той специальности, которую школьник хотел бы выбрать в качестве основного направления своей профессиональной деятельности.

Данная книга предназначена для школьников, которые хотели бы принять участие в российском, а возможно, и общемировом олимпийском движении, а также для тех учителей и руководителей образовательных учреждений, которые считают своей основной задачей не просто дать детям необходимый минимум знаний, умений и навыков, но и воспитать новую плеяду российских «Платонов и быстрых разумом Невтонов», способных достойно представлять нашу страну в мировом научном сообществе XXI в.

Цель книги — познакомить читателей с типовыми правилами проведения олимпиад различного уровня, с правилами участия в них и с примерами заданий, предлагаемых участникам олимпиад, что позволит оценить степень их сложности, а также попробовать свои силы в решении этих задач.

Соответственно построена и структура книги: каждый ее раздел посвящен одному из вышеперечисленных олимпийских этапов. Каждый такой раздел предваряется организационной информацией, а далее приведены условия задач (с разбором их решений в обособленном разделе книги). В том числе для школьного уровня приведен достаточно обширный набор задач с решениями (автор задач А. С. Пономаренко), который позволит учителю информатики осуществить предварительную подготовку школьников к участию в школьных и городских олимпиадах по программированию и станет для них первым шагом на пути к олимпийским высотам.

ШКОЛЬНЫЕ ОЛИМПИАДЫ

■ ОРГАНИЗАЦИОННАЯ ИНФОРМАЦИЯ

Олимпиады по программированию школьного уровня проводятся силами конкретных учебных заведений прежде всего для выявления талантливых учащихся, способных далее принимать участие в олимпиадах окружного, городского, всероссийского и т. д. уровней.

Предлагаемые на школьных олимпиадах задания могут быть практически любыми по содержанию (в том числе представлять собой «традиционные» задания из задачников по программированию), однако более предпочтительно, чтобы условия таких задач имели занимательную форму. Желательно также, чтобы олимпиадные задачи были приближены к реальным ситуациям, с которыми учащиеся могут сталкиваться в повседневной жизни и/или в учебной деятельности, а также по степени сложности несколько превосходили задачи, предлагаемые на обычных занятиях по теме «Программирование» (но вместе с тем были по силам учащимся, особенно на первых порах предолимпиадной подготовки).

Что же касается правил проведения школьных олимпиад, то они могут устанавливаться оргкомитетом (из числа педагогического состава образовательного учреждения, прежде всего учителей информатики), исходя из конкретных особенностей или потребностей данной школы, лицея и т. п. Однако рекомендуется, чтобы такие правила соответствовали общепринятым правилам организации городских, всероссийских и т. д. олимпиад, чтобы познакомить школьников с особенностями проведения таких мероприятий.

■ УСЛОВИЯ ЗАДАЧ

Предлагаемые здесь задачи[1] по уровню сложности соответствуют заданиям окружных олимпиад по программированию, проводившихся в Центральном округе г. Москвы. Условно эти задачи можно разбить на следующие группы:

1) реализация вычислительного процесса (в основном циклического);

[1] Автор задач А. С. Пономаренко.

2) формирование и обработка массивов (как числовых, так и символьных);

3) моделирование;

4) обработка символьной информации;

5) динамическая и статичная графика;

6) реализация заданного алгоритма или формулы.

Программы, являющиеся решениями этих задач, в основном написаны на языке программирования QBasic, но для некоторых задач даны листинги на языке Pascal.

Задачи на вычислительный процесс

1. Разность радикалов

Вычислите разность $\sqrt{1999} - \sqrt{1998}$. Ответ необходимо выдать в обычной десятичной записи, при этом *все* цифры ответа должны быть верными.

2. Экстремальные значения

Найдите экстремальные значения функции $y = \sin(n)$ на отрезке от 1 до 1999, где n — натуральное число. В ответе требуется указать не только значения функции, но и аргументы, при которых они вычислены (значения функции надо вычислять с двойной точностью).

3. Синус 2000 раз

Вычислите (с одинарной точностью) значение выражения: $A = \sin(\sin(\sin(...(\sin 2000°))...)$, в котором знак функции $y = \sin(x)$ встречается 2000 раз. Считать $\pi = 3{,}141593$.

4. Шалтаи и болтаи

175 шалтаев стоят дороже, чем 125 болтаев, но дешевле, чем 126 болтаев. Каждый из них дешевле рубля и стоит четное число копеек. Нетрудно доказать, что за трех шалтаев и одного болтая придется заплатить больше рубля (для решения данной задачи доказывать это не надо). Сколько стоит каждый из них? Выдайте на экран все решения данной задачи.

5. Два корня

Уравнение $\pi^2 \cdot (2\sin x - 5) = 54x^2 - 33\pi x$ имеет два корня на интервале от 0 до π. Вычислите эти корни с точностью до 10^{-6}.

6. Тригонометрическая система

Составьте программу нахождения первого положительного решения системы уравнений:

$$\begin{cases} \sin x + \sin y = -\sqrt{2}, \\ \cos x + \cos y = -1 \end{cases}$$

с точностью до одной секунды, если известно, что значения углов x и y находятся в третьей четверти.

7. «Хитрое» квадратное уравнение

Напишите программу для решения уравнения:

$$x^2 + 1{,}99999999999991x = 0{,}00000000000002.$$

Ответ необходимо выдать не в стандартной, а в обыкновенной десятичной форме числа.

8. «Сила» и «Мощь»

В спортивном зале есть тренажеры «Сила» и «Мощь». Масса тренажера «Сила» равна 12 кг, а тренажера «Мощь» — 15 кг. Масса всех тренажеров составляет 321 кг. Тренажер «Сила» стоит 8000 р., а тренажер «Мощь» — 12 000 р. Напишите программу, которая выдает на экран все возможные решения этой задачи, по образцу.

Тренажеры	«Сила»	«Мощь»	Суммарная стоимость
Количество	1	1	20 000

9. Точки с аппликатой 1

Известно, что точка $A(1; 1; 1)$ лежит в плоскости β, уравнение которой имеет вид:

$$7x - 5y + z - 3 = 0.$$

Напишите программу поиска всех точек, лежащих в этой плоскости, у которых аппликата $z = 1$, а все остальные *натуральные* координаты не превышают 30. На экран нужно вывести количество ответов и координаты искомых точек.

10. Метод Монте-Карло

Один из способов вычисления значений некоторых величин с помощью статистического вычислительного эксперимента получил название *метода Монте-Карло*. Примените этот метод для подсчета площади эллипса, уравнение которого имеет вид:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \Leftrightarrow \begin{cases} x = a\cos t, \\ y = b\sin t, \\ t \in [0;\ 2\pi], \end{cases}$$

где a — длина большой полуоси, а b — длина малой полуоси.

Обычно (в натурном эксперименте) этот процесс реализуется так: на листе бумаги рисуется эллипс с осями a и b, а также квадрат единичной площади. Этот лист бумаги равномерно посыпается песком в один слой. Подсчитывается количество песчинок, попавших в эллипс (m), и количество песчинок, попавших в единичный квадрат (n). Тогда приближенное значение площади эллипса равно m/n.

Ваша задача — заменить этот натурный эксперимент виртуальным (вычислительным). На экране нужно нарисовать эллипс размерами $a = 200$ пикселей, $b = 100$ пикселей, а также квадрат со стороной 100 пикселей. Песчинки заменяются случайно выводимыми точками. Требуется вести раздельный подсчет точек, которые попадают в эллипс и в квадрат. Кроме этой информации (как графической, так и символьной), на экран нужно выдать вычисленную площадь эллипса, считая площадь квадрата равной 1. (Теоретически площадь эллипса равна πab.)

Задачи на формирование и обработку массивов

11. Ряд из 100 натуральных чисел

Для первой сотни натуральных чисел, введенных без пробелов, определите цифру, стоящую на k-м месте. (Число k задается с клавиатуры.)

12. Поиск элемента по заданному признаку

Напишите программу поиска элемента (элементов) массива — названий всех месяцев года по заданному признаку (буквосочетанию), введенному с клавиатуры. Первоначальный и конечный массивы нужно вывести на экран (конечный массив может быть пустым). Длина «слова-признака» может быть любой, в том числе больше длины любого элемента заданного массива.

13. Буквенное уравнение

Найдите все решения уравнения $\sqrt{xxxyyyyz} = vvvz$, где x, y, z, v — некоторые цифры, отличные от нуля.

14. «Черные пятницы»

Напишите программу для выявления всех пятниц на конкретный год (как простой, так и високосный), заданный днем недели, приходящимся на 1 января, а также для выявления всех «черных пятниц», выпадающих на 13-е число любого месяца этого года.

15. Числа Фибоначчи

Последовательность (ряд) Фибоначчи задается следующим образом: первые два числа в ней равны единицам, а каждое последующее число представляет собой сумму двух предыдущих. Вот несколько первых членов ряда Фибоначчи: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... Если в этой последовательности убрать пробелы и запятые, то получится некое число A, цифры которого представляют собой числа Фибоначчи, «склеенные» друг с другом по порядку их следования. Напишите программу, позволяющую выяснить, какая цифра стоит на k-м месте в числе A ($1 \leqslant k \leqslant 234$).

Задачи на моделирование

16. Иллюзия вращения

Создайте иллюзию вращения в пространстве плоской кривой — *трехлепестковой розы*, уравнение которой, заданное параметрически, имеет вид:

$$\begin{cases} x = a \cdot \cos 3\varphi \cdot \cos \varphi, \\ y = a \cdot \cos 3\varphi \cdot \sin \varphi, \end{cases}$$

где $\varphi \in [0, 2\pi]$, а a — некоторое число, от которого зависят размеры кривой на экране.

17. Лифт

В подъезде 14-этажного дома есть два работающих лифта, причем на каждом этаже имеется всего одна кнопка вызова для них. Смоделируйте на экране процесс перемещения этих лифтов, в том числе открывание и закрывание дверей, если:

- оба лифта свободны;
- человек, которому нужен лифт, и сами эти лифты находятся на разных (произвольных) этажах;

• по вызову приезжает лифт, ближайший к данному этажу, а если оба лифта удалены от места вызова на одинаковое количество этажей, то приезжает любой из них;

• вызванный лифт отправляется на указанный этаж.

18. Реостат

Смоделируйте опыт по изучению работы реостата (рис. 1). Если бегунок находится в левом положении, то вольтметр должен показывать максимальное напряжение в 4 В. При перемещении бегунка вправо стрелка вольтметра должна отклоняться влево (напряжение в цепи падает вплоть до 0 В при крайнем правом положении бегунка).

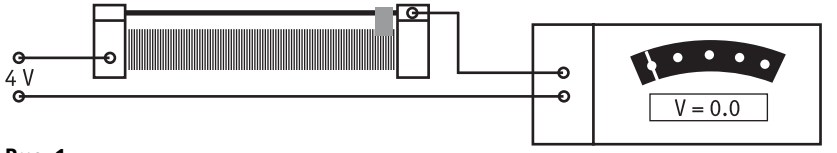


Рис. 1

19. Опыт Бюффона

Французский естествоиспытатель Бюффон когда-то проделал следующий опыт. Он подбрасывал монету 4040 раз и подсчитывал количество выпавших «решек» и «орлов». Классическая вероятность выпадения как «решки», так и «орла» равна 50%. Напишите программу, повторяющую опыт Бюффона 10 раз, причем необходимо каждый раз фиксировать на экране вероятность выпадения «орла», а в конце работы программы выдать на экран:

• усредненный результат десяти опытов по выпадению «орла» в процентах с точностью до 10^{-6};

• абсолютное отклонение усредненного результата от классической вероятности этого процесса в процентах.

В программе используйте генератор псевдослучайных чисел (RANDOM), встроенный в язык программирования.

20. Картинги

Два картинга (гоночных автомобиля) одновременно, с одной линии старта начинают движение по кольцевым трассам с постоянными скоростями, но в противоположных направлениях. Радиус трассы первого картинга R_1 = 105 м, а его скорость V_1 = 54 км/ч. Для второго картинга эти значения: R_2 = 120 м, V_2 = 72 км/ч. Напишите программу, демонстрирующую эту гон-

ку. Определите время, когда оба картинга снова окажутся на линии старта, и количество кругов, которые они сделают до этого.

21. Планета

Напишите программу, демонстрирующую движение планеты вокруг звезды на фоне звездного неба. При этом звездный фон, по которому происходит движение планеты, не должен стираться в процессе работы программы.

Задачи на использование символьных переменных

22. Анаграммы

Напишите программу, которая печатает все анаграммы 4-буквенного слова, введенного с клавиатуры (пример анаграмм слова «лист»: «тлис», «слит» и т. д.).

23. Сложение многозначных чисел

Случайным образом сгенерированы два 36-значных числа. Напишите программу, выполняющую поразрядное («в столбик») сложение этих чисел.

24. Кочерга

Напишите программу для согласования числительного *N* со словом «кочерга». Число *N* вводится с клавиатуры, после чего на экране должна выводиться фраза, соответствующая правилам орфографии и имеющая примерно такой вид: «1 кочерга», «3 кочерги», «5 кочерёг» и т. д.

25. Палиндромы

Существуют фразы-перевертыши (палиндромы), например: «А роза упала на лапу Азора». Точнее, такие фразы являются палиндромами на слух. Фразу можно считать настоящим палиндромом только тогда, когда в ней исключены все пробелы и вся фраза написана либо строчными, либо прописными буквами, например: «АРОЗАУПАЛАНАЛАПУАЗОРА». Другими словами, палиндромом является слово, которое можно совершенно одинаково читать как справа налево, так и наоборот.

Напишите программу, которая вычисляет палиндромы с точностью до пробелов (т. е. когда вводимая фраза может содержать пробелы, но все слова в ней должны быть написаны либо строчными, либо прописными буквами).

26. Кроссворд

Напишите программу, составляющую заполненный кроссворд из шести 5-буквенных слов, заданных с клавиатуры. Кроссворд должен иметь примерно такой вид:

У	К	С	У	С
К		О		А
Р	Ы	В	О	К
О		О		Л
П	А	К	Л	Я

Определения к словам и нумерацию слов по горизонтали и вертикали в данном случае выводить на экран не требуется. Предполагается, что в кроссворде есть *только два* слова, начинающихся с одной и той же буквы (верхней левой ячейки таблицы).

27. Пароль доступа

Программист защитил свою программу паролем в виде осмысленного русского слова, состоящего из строчных букв, и некоторого числа. Чтобы разгадать этот пароль было труднее, часть букв вводилась в латинском регистре (т. е. символьная часть пароля состояла из букв одинакового начертания как в кириллице, так и в латинице). Злоумышленник подсмотрел символьную часть пароля — слово «ВЕСНА». Кроме того, он заметил, что две буквы этого слова были введены именно на кириллице. И наконец, он узнал, что число, стоящее в конце пароля, — это сумма ASCII-кодов букв, записанных в символьной части пароля.

Напишите программу, которая выдает все возможные пароли, соответствующие описанным правилам, причем латинские буквы должны быть выведены другим цветом.

28. Перевод «16 → 10»

Напишите программу для перевода положительного шестнадцатеричного числа, содержащего не более четырех значащих цифр (от 0,001 до FFFF), в десятичное число. Ответ нужно выводить в виде таблицы, например:

Система счисления	
16	10
0.0EF	0.058349609375

29. Неправильное сокращение дробей

Среди дробей, числитель и знаменатель которых — двузначные числа, есть и такие, что при неправильном их сокращении все равно получается правильный ответ. Например:

$\frac{19}{95}$ — если зачеркнуть девятки в числителе и знаменателе, то получается правильный ответ $\frac{1}{5}$;

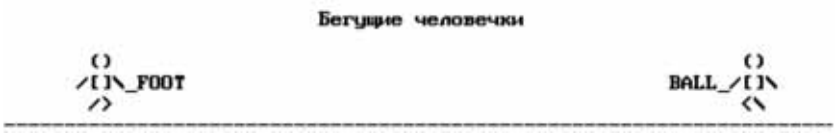
$\frac{26}{65}$ — если зачеркнуть шестерки, получается правильный ответ $\frac{2}{5}$.

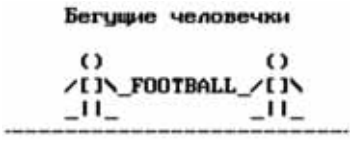
Напишите программу для поиска таких математических курьезов для трехзначных чисел в числителе и знаменателе дроби (для определенности будем считать, что числитель всегда больше знаменателя и что в ответе могут фигурировать только несократимые дроби, например, дробь $\frac{676}{169} = \frac{76}{19} = 4$ не удовлетворяет условию задачи, так как она сократима). Кроме того, для упрощения программы будем брать только случаи, когда в исходной дроби зачеркивается только первая цифра числителя и вторая цифра знаменателя.

30. Бегущие человечки

Когда-то мониторы не поддерживали графический режим, и программистам приходилось изощряться, чтобы создать «псевдографику» для изображения некоторых процессов (например, для создания заставок к играм).

Напишите программу, изображающую бегущих навстречу друг другу человечков с плакатами «FOOT» и «BALL». Для изображения человечков можно использовать символы: `)`, `(`, `/`, `[`, `]`, `\`, `>`, `<`, `_`, `|` и пробел (а также любые другие символы, позволяющие создать иллюзию бегущих человечков). Вот примерный вид середины процесса и его окончания:





31. Признак делимости на 11

Известен следующий признак делимости числа n на 11: «Натуральное число n кратно 11 тогда и только тогда, когда сумма его цифр, стоящих на нечетных местах, равна сумме цифр, стоящих на четных местах, или когда разность этих сумм (по модулю) кратна 11». Напишите программу, проверяющую согласно этому признаку делимости введенное число N, содержащее не более 9 цифр.

Задачи на динамическую и статичную графику

32. Вовочка на катере

Ровно в 12 ч 00 мин Вовочка выезжает на катере от пристани A до пристани B, которая находится на расстоянии 50 км вниз по течению реки. Первые 45 мин каждого часа он плывет по течению, а затем разворачивается и 15 мин плывет против течения (зачем он так делает, никому не понятно, но для задачи это и неважно ☺). Известно, что собственная скорость катера — 15 км/ч, а скорость течения реки — 5 км/ч. Изобразите на экране процесс движения катера от A до B с одновременной выдачей информации о текущем времени, расстоянии катера от пристани A и скорости катера в данный момент, а также выведите в конце работы программы точное время прибытия на пристань B (с точностью до минуты).

33. Зеркальное отображение текста

Напишите программу, зеркально отображающую заданное пользователем слово или фразу относительно горизонтальной прямой. (Текст вводится с клавиатуры.)

34. Шарикоподшипник

Напишите программу, изображающую шарикоподшипник с n шариками. Натуральное число n (где $n \in [6, 30]$) вводится с клавиатуры.

35. Летящее копье

Напишите программу, изображающую полет копья, центр тяжести которого перемещается по параболе, а само копье всегда занимает положение касательной к текущей точке своей траектории.

36. Зеркало

На столе стоит зеркало, а перед ним — свеча, которая отражается в зеркале. Пламя свечи время от времени колеблется. Напишите программу, демонстрирующую это на экране.

37. Летящий конверт

Напишите программу, которая изображает на экране летящий по ветру почтовый конверт (который в полете вращается и увеличивается в размерах).

38. Корабельный винт

Напишите программу, изображающую вращение корабельного винта (он может вращаться как по часовой стрелке — движение корабля вперед, так и против часовой стрелки — задний ход корабля). Направление вращения задается с клавиатуры. Нарисовать лопасти винта можно в виде трехлепестковой розы:

$$\begin{cases} x = r \cdot \cos 3\varphi \cdot \cos \varphi, \\ y = r \cdot \cos 3\varphi \cdot \sin \varphi, \end{cases}$$

где $\varphi \in [0, 2\pi]$, а r — длина лепестка.

39. Закрашенная таблица

Напишите программу, которая рисует таблицу размерами $M \times N$ клеток (числа M и N задаются с клавиатуры, ограничения на их значения формирует сам участник олимпиады). Случайным образом часть клеток таблицы окрасьте в синий цвет. В процессе окрашивания запрещается подсчитывать количество окрашенных клеток или запоминать их координаты. Уже после окончания рисования отдельно просканируйте изображение таблицы и подсчитайте количество клеток, окрашенных в синий цвет.

40. Путешествие по глобусу

Широко известна задача об указании таких точек на модели Земли (шар с твердой поверхностью), что, пройдя из одной из них 100 км строго на юг (по меридиану), затем 100 км строго на восток (по параллели) и 100 км строго на север, вы оказываетесь в той же самой точке, откуда начали свое движение. Как правило, одну такую точку знают все: это Северный полюс. Однако эта задача имеет множество решений. Например, вблизи Южного полюса тоже есть параллель, все точки которой отвечают условиям задачи: эта параллель находится на 100 км севернее паралле-

ли, которая имеет длину 100 км. Напишите программу, которая выводила бы красным цветом подобные точки, а движущаяся зеленая точка оставляла бы след, по которому происходит описанное выше движение путешественника.

41. Парашют

Напишите программу, демонстрирующую спуск парашютиста, который сначала опускался вертикально вниз, затем подул ветер, и парашютист сместился к краю монитора, а затем ветер подул в обратную сторону, и парашютист начал смещаться к другому краю монитора.

Реализация заданного алгоритма (формулы)

42. Уравнение Пелля

Напишите программу, позволяющую найти пять первых натуральных решений уравнения Пелля:

$$x^2 - 5y^2 = 1.$$

43. Числа, кратные трем

Напишите программу, которая:

- вводит с клавиатуры первый член последовательности — натуральное число, кратное трем и не превышающее 100;
- вычисляет следующие члены последовательности, каждый из которых равен сумме кубов цифр предыдущего.

Например, если введено исходное число 33 (первый член последовательности), то:

второй член — $3^3 + 3^3 = 27 + 27 = 54$;
третий член — $5^3 + 4^3 = 125 + 64 = 189$;
четвертый член — $1^3 + 8^3 + 9^3 = 1 + 512 + 729 = 1242$;
пятый член — $1^3 + 2^3 + 4^3 + 2^3 = 1 + 8 + 64 + 8 = 81$;
шестой член — $8^3 + 1^3 = 512 + 1 = 513$;
седьмой член — $5^3 + 1^3 + 3^3 = 125 + 1 + 27 = 153$

(на этом вычисления надо прекратить — известно, что *любая* последовательность, формируемая по этому закону, рано или поздно приводит к числу 153).

Таким образом, если введено исходное число, равное 33, то на экран должна быть выведена запись: 33 54 189 1242 81 513 153. Для другого исходного числа запись будет другой, но заканчиваться она будет тоже числом 153.

44. Задача о коровах

Есть прекрасная и довольно сложная (с точки зрения математики) задача Л. Н. Толстого о коровах, которые паслись на одном лугу в течение лета: «Трава на всем лугу растет одинаково густо и быстро. 70 коров съедают всю траву за 24 дня, а 30 коров — за 60 дней. За сколько дней съедят всю траву n коров?»

Решение этой задачи в общем виде определяется формулой:

$$t = \frac{4800}{3n - 10},$$

где n — количество коров ($n > 17$), которые паслись на этом лугу в течение t дней.

Напишите программу, находящую все целочисленные решения этой задачи при условии, что в нашем регионе трава растет не более четырех месяцев (весна — лето).

45. Температурные шкалы

Широко известны четыре температурные шкалы: Цельсия (t °C), Реомюра (t °R), Фаренгейта (t °F) и Кельвина (t K). Существуют формулы перехода от одной температурной шкалы в другую:

$$t\ °\mathrm{R} = 0{,}8 \cdot t\ °\mathrm{C};$$

$$t\ °\mathrm{F} = 1{,}8 \cdot t\ °\mathrm{C} + 32;$$

$$t\ \mathrm{K} = t\ °\mathrm{C} + 273.$$

Напишите программу для пересчета из одной температурной шкалы во все другие (с точностью до целого числа градусов) с учетом того, что $t\ \mathrm{K} \geqslant 0$ (0 K — абсолютный температурный нуль). Ответы оформите в виде таблицы.

ОКРУЖНЫЕ (ГОРОДСКИЕ) ОЛИМПИАДЫ. МОСКОВСКАЯ ГОРОДСКАЯ ОЛИМПИАДА ПО ПРОГРАММИРОВАНИЮ

■ ОРГАНИЗАЦИОННАЯ ИНФОРМАЦИЯ

Сайт: http://www.olympiads.ru/moscow/index.shtml
Сроки проведения: февраль.

Московская олимпиада по программированию проводится в два этапа: первый этап включает регистрацию участников и квалификационный отбор, а второй (заключительный) этап является собственно очным олимпиадным соревнованием. Принять участие в этой олимпиаде могут все желающие школьники (в том числе не из Москвы).

В предыдущие годы Московская олимпиада по программированию проводилась для школьников 7—9 классов, однако с 2010/2011 учебного года в рамках этого мероприятия организованы соревнования для учащихся 10—11 классов и отдельно — для учащихся 6—9 классов. Олимпиада ориентирована в первую очередь на школьников, начавших изучать программирование относительно недавно; организаторы не рекомендуют принимать участие в этой олимпиаде школьникам, уже имеющим значительные успехи в олимпиадах по программированию высокого уровня (например, на заключительном этапе Всероссийской олимпиады школьников или на Открытой олимпиаде по программированию).

Квалификационный отбор (заочный) заключается в решении и сдаче в автоматическую проверяющую систему трех (для учащихся 10—11 классов) или двух (для учащихся 6—9 классов) задач одной из московских олимпиад по программированию для 7—9 классов за предыдущие годы, причем все решенные задачи обязательно должны быть из одной олимпиады (нельзя сдать по одной задаче из трех разных олимпиад). Решения задач сдаются в проверяющую систему на сайте informatics.mccme.ru в разделе «Квалификационный отбор на Московскую олимпиаду по программированию 2011 года» (http: // informatics.mccme.ru/moodle/course/view.php?id=127) под личным логином участника.

После прохождения квалификационного отбора тем, кто планирует принимать участие в заключительном этапе Московской олимпиады по программированию, необходимо также заполнить регистрационную форму.

После окончания регистрации для соответствующих классов публикуются списки зарегистрированных участников, а затем на сайте публикуется информация о времени и месте (местах) проведения олимпиады.

Правила проведения Московской олимпиады по программированию

Олимпиада (очный этап) проводится в один тур продолжительностью 4 часа, в ходе которого участникам выдается комплект из 4—6 задач.

Решением каждой задачи должна являться программа на одном из допустимых языков программирования. Решение проверяется на заранее заготовленном жюри наборе тестов. В зависимости от результата работы программы тот или иной тест либо засчитывается (с начислением участнику баллов за этот тест), либо не засчитывается. Полное решение каждой задачи оценивается 100 баллами.

Участник в любой момент во время тура может отправить решение на проверку в проверяющую систему. Во время тура решение проверяется на части тестов (с общей суммой 60 баллов), а результат прохождения тестов сразу сообщается участнику. Основная проверка решений на полном наборе тестов осуществляется после окончания тура. При этом для каждой задачи проверяется последнее присланное на проверку решение и полученные участником баллы заносятся в итоговый протокол.

Решение должно выдавать одинаковые ответы на одинаковые тесты вне зависимости от времени запуска и программного окружения. Жюри вправе произвести неограниченное количество запусков программы участника и выбрать наихудший результат по каждому из тестов. Кроме того, в условии задач может быть указано максимальное время работы программы на одном тесте и максимальный объем памяти, который может использовать программа (включая всю память, доступную процессу решения: код самой программы, область переменных программы, выделяемую программой память, а также память под системные нужды — стек и т. д.).

Во время тура участникам олимпиады запрещается пользоваться Интернетом, любыми электронными устройствами, в том числе личными компьютерами, калькуляторами, электронными записными книжками, средствами связи (пейджерами, мобильными телефонами и т. п.), плеерами, электронными носителями информации (дискетами, CD- и DVD-дисками, модулями флеш-памяти и т. п.). Однако во время тура участники могут использовать принесенную с собой учебную литературу и личные записи (не электронные версии).

В течение тура участник может задавать членам жюри вопросы по условиям задач и получать на них ответы. При этом вопросы задаются через проверяющую систему и должны формулироваться так, чтобы ответ был в форме «да» или «нет».

За сохранность своих данных во время тура ответственность каждый участник несет самостоятельно. Чтобы минимизировать возмож-

ные потери данных, необходимо своевременно сохранять свои файлы и данные на компьютере.

Для учащихся 6—9 классов разбор задач и награждение победителей и призеров проводятся в день олимпиады. Результаты проверки решений учащихся 10—11 классов публикуются в день проведения олимпиады вечером, апелляции по результатам проверки принимаются по e-mail в течение 1—2 дней. После рассмотрения апелляций и принятия жюри решения о победителях и призерах олимпиады списки победителей и призеров публикуются на сайте олимпиады.

■ УСЛОВИЯ ЗАДАЧ

Ниже приведены тексты задач с критериями оценки решений X Московской олимпиады по программированию[1]; в разделе решений, указаний и ответов приведены решения этих задач на двух языках программирования (QBasic и Pascal), а также тесты, на которых проверялась правильность работы программ, и критерии оценки этих задач.

1. Дороги

Имеются четыре города: *A*, *B*, *C*, *D* и дорожная сеть между ними (рис. 2). Напишите программу, выводящую на экран длину кратчайшего пути и сам этот путь между городами *A* и *B*.

Формат входных данных

Вводится пять целых чисел l_j, где $j \in \{1, 2, 3, 4, 5\}$ — длины соответствующих дорог ($10 \leqslant l_j \leqslant 100$).

Формат выходных данных

Первой строкой выводится целое число — длина кратчайшего пути. Второй строкой выводится последовательность городов, из которых состоит кратчайший путь, начиная с города *A*. Если решений несколько, то достаточно вывести любое из них.

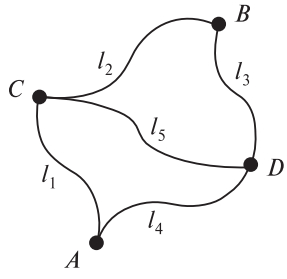


Рис. 2

Пример

Входные данные	Выходные данные
10 40 20 50 15	45 ACDB

Оценка за полное решение задачи — 5 баллов.

[1] Материал предоставлен А. С. Пономаренко (см. [4]).

2. «Ширли» и «Мырли»

Известно, что 6 моделей автомобиля «Ширли» стоят дороже, чем 7 моделей автомобиля «Мырли», а 5 моделей «Ширли» — дешевле 6 моделей «Мырли». Каждая модель стоит нечетное число рублей и ее цена не превышает 100 р. Стоимость трех моделей автомобиля «Ширли» и одной модели «Мырли» больше 341 р. Определить и вывести на экран количество решений данной задачи и цену каждой модели в каждом возможном случае.

Формат входных данных

Вводимых данных нет.

Формат выходных данных

Первой строкой выводится количество решений задачи (N). В каждой из последующих N строк выводятся два целых числа: цена модели «Ширли» и цена модели «Мырли». Порядок следования этих N строк значения не имеет.

Оценка за полное решение задачи — 7 баллов.

3. Кузнечик

Кузнечик прыгает по числовой оси. Он может прыгать только на 6 делений вперед или на 7 делений назад. Напишите программу, выводящую на экран последовательность прыжков и их количество, если кузнечику надо попасть из нулевой точки в некоторую заданную точку.

Формат входных данных

Вводится одно целое число N ($-10 \leqslant N \leqslant 10$) — координата точки, в которую должен попасть кузнечик.

Формат выходных данных

Программа должна вывести строку из букв F и B, задающую последовательность прыжков: F — прыжок на 6 делений вперед (forward), B — прыжок на 7 делений назад (back). После этого нужно вывести количество таких прыжков.

Если решений возможно несколько, то достаточно вывести любое из них.

Пример

Входные данные	Выходные данные
5	FFB 3

Оценка за полное решение задачи — 9 баллов.

4. Отрезки

Пользователь вводит координаты двух точек $A(a, b)$ и $B(m, n)$ в прямоугольной системе координат XOY, начало которой находится в центре экрана, а единица масштаба по обеим осям составляет 20 пикселей. Требуется найти такую точку C, лежащую на оси OY, чтобы сумма длин отрезков AC и BC была минимальной.

Составьте программу, которая:

- определяла бы и выводила на экран, в каких пределах должны находиться целые координаты a, b, m и n, чтобы обе точки $A(a, b)$ и $B(m, n)$ отображались на экране;
- позволяла бы пользователю ввести целые числа a, b, m и n; при этом уже не требуется проверять, лежат ли введенные числа в найденных промежутках;
- рисовала бы систему координат XOY (начало координат — в центре экрана) с указанием направления и названий осей, а также отмечала на осях и подписывала точки с целочисленными координатами;
- выводила бы на экран координаты такой точки C, лежащей на оси OY, чтобы сумма длин отрезков AC и BC была минимальной; отмечала маленькими кружками и подписывала на экране точки A, B, C и рисовала отрезки AC и BC.

Рекомендуется использовать графический режим с разрешением 640 × 480 точек.

Оценка за полное решение задачи — 15 баллов.

5. Рикошет

Напишите программу, отображающую в графическом режиме прямоугольную область размером 200 × 200 пикселей идвигающуюся внутри нее точку (рис. 3). В начальный момент времени точка находится в центре прямоугольной области и движется в направлении, заданном пользователем. Достигнув края прямоугольной области, точка отскакивает от него, причем угол падения α равен углу отражения β (см. рис. 3).

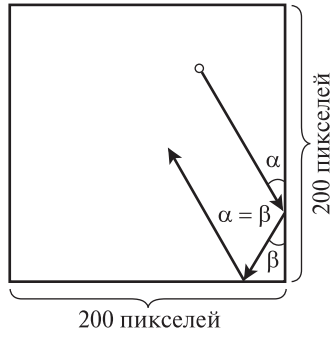


Рис. 3

При этом точка не теряет своей скорости, затем отскакивает опять и т. д.

Формат входных данных

Направление движения точки задается как угол φ (в градусах) от оси OX против часовой стрелки ($0 \leqslant \varphi \leqslant 360$). Например, угол 0° соответствует движению вправо, а 90° — вертикально вверх.

Формат выходных данных

Графическое изображение движения точки, которое продолжается, пока не будет нажата любая клавиша. Скорость движения можно выбрать произвольно в диапазоне от 70 до 200 пикселей в секунду. Точка должна быть светлой, а фон — темным (чтобы было лучше ее заметно).

Оценка за полное решение задачи — 18 баллов.

6. Степень

Чтобы проверить, как ученики умеют считать, Мария Ивановна каждый год задает им на дом одну и ту же задачу: для заданного натурального числа A найти такое минимальное натуральное N, что N в степени N (т. е. число N, умноженное на себя N раз) делится на A. От года к году и от ученика к ученику меняется только число A.

Вы решили помочь будущим поколениям. ☺ Для этого вам необходимо написать программу, решающую эту задачу.

Формат входных данных

Вводится единственное число A ($1 \leqslant A \leqslant 1\,000\,000\,000$ — на всякий случай, вдруг Мария Ивановна задаст большое число, чтобы «завалить» кого-нибудь... ☺).

Формат выходных данных

Вывести единственное число N.

Максимальное время работы на одном тесте: 3 с.

Примеры

Входные данные	Выходные данные
8	4
13	13

Оценка за полное решение задачи — 30 баллов.

ВСЕРОССИЙСКИЕ ОЛИМПИАДЫ. ВСЕРОССИЙСКАЯ ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

■ ОРГАНИЗАЦИОННАЯ ИНФОРМАЦИЯ

Сайт: http://www.step-into-the-future.ru
Сроки проведения: февраль—май.

Всероссийская олимпиада школьников «Шаг в будущее» была основана в 1991 г. Московским государственным техническим университетом им. Н. Э. Баумана как одно из ведущих направлений реализации Российской научно-социальной программы для молодежи и школьников «Шаг в будущее». Эта программа в соответствии с Распоряжением Правительства Российской Федерации от 20 мая 1998 г. № 573-р является составной частью государственной политики в области кадрового обеспечения российской науки. Председатель оргкомитета олимпиады — ректор МГТУ им. Н. Э. Баумана, профессор Анатолий Александрович Александров. Научный руководитель программы «Шаг в будущее» — президент МГТУ им. Н. Э. Баумана, академик Российской академии наук Игорь Борисович Федоров. Коллектив создателей программы «Шаг в будущее» награжден Премией Президента Российской Федерации в области образования. Победители и призеры Всероссийской олимпиады школьников «Шаг в будущее» по рекомендации Министерства образования и науки РФ ежегодно награждаются стипендиями Президента Российской Федерации. Ежегодно лауреаты научных олимпиад программы «Шаг в будущее» в составе 25 молодых ученых и исследователей со всего мира представляют Российскую Федерацию на церемонии вручения Нобелевских премий.

Основные цели Всероссийской олимпиады школьников «Шаг в будущее»:

- выявление и развитие у учащихся профилированных творческих способностей и интереса к научно-исследовательской деятельности;
- формирование ключевых компетенций, профессионально-значимых качеств личности и мотивации к практическому применению предметных знаний;
- создание необходимых условий для поддержки творчески одаренных детей;
- научное просвещение и целенаправленная профессиональная ориентация учащейся молодежи;

- распространение и популяризация научных знаний;
- привлечение в вузы школьников, наиболее способных и подготовленных к освоению программ высшего профессионального образования.

Всероссийская олимпиада школьников «Шаг в будущее» проводится ежегодно на всей территории России — от Карелии на западе до Якутии на востоке и от Мурманска на севере до Дагестана на юге. Базовыми организациями для проведения олимпиады являются региональные представительства программы «Шаг в будущее». Сегодня в разветвленной сети таких представительств действуют 102 координационных центра и более 300 организаций — ассоциированных участников программы. Каждое региональное представительство — это комплекс, который включает в себя школы, вузы, центры молодежного творчества, научные институты и предприятия. Как правило, в состав организаций-учредителей координационных центров входят региональные органы управления образованием, наукой, молодежной политикой. Координационные центры программы «Шаг в будущее» поддерживают региональные, городские и сельские научные и профессиональные молодежные общества.

Всероссийская олимпиада школьников «Шаг в будущее» включает в себя два вида конкурсных испытаний: **научно-образовательное соревнование** (защита проектов и выполнение заданий по общеобразовательным предметам или комплексам предметов) и **академическое соревнование** (выполнение заданий по общеобразовательным предметам или комплексам предметов). Задания олимпиады формируются по профилям, ориентированным на выявление граждан, наиболее способных и подготовленных к освоению программы высшего профессионального образования. В задания олимпиады могут быть также включены сочинение на темы, связанные с профильной областью предметных знаний, и испытание творческих способностей участников.

Из победителей и призеров Всероссийской олимпиады школьников «Шаг в будущее» в рамках программы «Шаг в будущее» ежегодно формируются и направляются национальные делегации на ведущие международные молодежные мероприятия. Среди них — Стокгольмский международный молодежный научный семинар с участием в церемонии вручения Нобелевских премий (с 1998 г.), Соревнование молодых ученых Европейского союза (с 1997 г.), Лондонский международный молодежный научный форум (с 1996 г.), Международная и Европейская научные выставки «ЭКСПО-НАУКА» (с 1996 г.), Международная научная и инженерная выставка Intel ISEF (США, с 1996 г.), Международный космический лагерь (Норвегия, с 2002 г.) и др. Лауреаты Всероссийской олимпиады школьников «Шаг в будущее» в составе национальных делегаций неоднократно занимали на международном уровне призовые места, награждались спецпризами и в том числе стажировками в ведущих зарубежных научных центрах.

За весь период развития Всероссийской олимпиады школьников «Шаг в будущее» в ней приняли участие школьники из 78 субъектов Российской Федерации, 242 городов и 376 сельских населенных пунктов. Ежегодно научно-профессиональные делегации в составе ученых МГТУ им. Н. Э. Баумана, Российской академии наук, Российской академии образования и ряда других организаций выезжают в регионы России для проведения первого этапа олимпиады. При этом важно учесть, что особое значение в программе «Шаг в будущее» уделяется именно образованию и воспитанию детей, проживающих в отдаленных городах и поселках: в настоящее время в олимпиадных мероприятиях участвует более 10 000 таких участников.

Научно-образовательное соревнование

Первый (отборочный) этап проведения олимпиады проводится в очно-заочной форме на федеральных окружных соревнованиях, конкурсных научных мероприятиях организаций — официальных участников программы «Шаг в будущее», а также в рамках конкурсного отбора, которое проводит жюри олимпиады. Отборочный этап проводится в два тура — это защита научно-исследовательских проектов и конкурсный отбор жюри олимпиады.

Для участия в отборочном этапе научно-образовательного соревнования учащиеся подают заявку на регистрацию и материалы, содержащие результаты выполнения научно-исследовательской работы, в одну из базовых организаций олимпиады в определенные оргкомитетом сроки. По результатам отборочного этапа локальные жюри или жюри олимпиады определяют победителей и призеров первого этапа олимпиады.

Второй (заключительный) этап проведения олимпиады проводится в очной форме на Всероссийской научной конференции молодых исследователей «Шаг в будущее», Российской молодежной научной и инженерной выставке в два тура.

Один из туров заключительного этапа — *научное соревнование* — организуется в форме защиты научно-исследовательских работ. Другой — *академический тур* — включает выполнение профильных заданий по общеобразовательному предмету или комплексу предметов. Тур «Защита научно-исследовательских работ» проводится в течение двух дней (согласно расписанию работы профильных секций конференции) либо в течение трех дней (согласно расписанию работы выставочной экспозиции).

На конференции ориентировочное время для доклада результатов работы — 10 минут. После доклада автор защищает свою работу, отвечая на вопросы экспертов и присутствующих. Для обсуждения доклада может быть выделено дополнительное время. Оценка работы проводится экспертной комиссией секции конференции в составе 5 ученых и специалистов, каждый из которых выставляет свою оценку за защиту работы (не более 20 баллов). Общая оценка (максимум 100 баллов) вычисляется как сумма оценок экспертов.

На выставке ориентировочное время для доклада результатов работы также 10 минут. Автор доклада защищает свой проект перед высокопрофессиональными членами жюри в индивидуальном порядке. Оценка работы проводится членами жюри (от 6 до 15 ученых и специалистов). Каждый из них выставляет свою оценку за защиту работы (не более 100 баллов); общая оценка рассчитывается как среднее арифметическое.

Тур «Выполнение профилированных заданий по общеобразовательному предмету/комплексу предметов» проводится в течение одного дня, который не совпадает с проведением тура «Защита научно-исследовательских работ». Тур олимпиады включает олимпиадные задания по общеобразовательному предмету или комплексу предметов. При этом в билете для каждого задания указывается наибольший балл, начисляемый за его выполнение (максимальная сумма баллов участника тура составляет 100 баллов). На выполнение олимпиадных заданий отводится 240 минут.

Разбор профилированных заданий олимпиады проводится в очной форме (согласно расписанию олимпиады) или через официальный сайт олимпиады.

Победители и призеры заключительного этапа научно-образовательного соревнования определяются по сумме баллов, полученных каждым участником в ходе защиты научно-исследовательской работы на конференции/выставке и выполнения олимпиадных заданий в рамках академического тура. Общая сумма баллов — не более 200.

Академическое соревнование

Первый (отборочный) этап проведения олимпиады проводится организаторами олимпиады в очной или заочной форме и (или) с применением дистанционных образовательных технологий. Для участия в отборочном этапе академического соревнования учащиеся подают заявку на регистрацию в одну из базовых организаций олимпиады. Для заочной формы отборочного этапа олимпиады вместе с заявкой подается выполненный вариант олимпиадного задания по общеобразовательному предмету или комплексу предметов. Очная форма отборочного этапа проводится в общеобразовательных учебных заведениях в виде выполнения профилированных заданий по общеобразовательному предмету/комплексу предметов.

По результатам отборочного этапа академического соревнования жюри олимпиады определяет победителей и призеров первого этапа.

Второй (заключительный) этап проведения олимпиады проводится в очной форме в Москве, а также в других городах Российской Федерации с участием базовых организаций и вузов-партнеров олимпиады в виде выполнения заданий по общеобразовательному предмету или комплексу предметов. Конкретные сроки проведения мероприятий академического соревнования ежегодно утверждаются председателем

оргкомитета олимпиады и председателем центрального совета программы «Шаг в будущее».

Выполнение профилированных заданий по общеобразовательному предмету/комплексу предметов проводится в течение одного или двух дней. В билете для каждого задания указывается наибольший балл, начисляемый за его выполнение. Максимальная сумма баллов составляет 100 баллов. На выполнение олимпиадных заданий отводится 240 минут.

Разбор профилированных заданий олимпиады проводится в очной форме (согласно расписанию олимпиады) или через официальный сайт олимпиады.

Олимпиада школьников «Шаг в будущее» (предмет «Информатика»)

Олимпиада школьников «Шаг в будущее» по предмету «Информатика» проводится с 2004 г. Московским государственным техническим университетом им. Н. Э. Баумана при участии Московского государственного института радиотехники, электроники и автоматики (технического университета) и Московского государственного технического университета им. А. Н. Косыгина. В последние годы эта олимпиада входит в Перечень олимпиад, утвержденный Министерством образования и науки РФ, что позволяет победителям и призерам заключительного этапа в течение одного года пользоваться одной из льгот — поступлением в вуз без экзаменов или высшим баллом (100) по предмету соответствующего вступительного испытания. При этом каждый вуз самостоятельно устанавливает конкретный вид льготы в зависимости от уровня олимпиады и степени диплома участника заключительного этапа олимпиады.

Олимпиада школьников «Шаг в будущее» по предмету «Информатика» проводится в виде академического и научно-образовательного соревнований, каждое из которых проходит в два этапа — отборочный и заключительный. Отборочный этап, как правило, проводится в заочной форме, а заключительный — только в очной. В заключительном этапе принимают участие только победители и призеры отборочного этапа.

Научно-образовательное соревнование олимпиады школьников «Шаг в будущее» по предмету «Информатика» проводится в виде двух туров: «Защита научно-исследовательских работ» и «Решение профилированных заданий по предмету «Информатика».

Тур «Защита научно-исследовательских работ». Каждый участник научно-образовательного соревнования самостоятельно выполняет научно-исследовательскую работу в соответствии с общими требованиями, размещенными на сайте программы «Шаг в будущее» (http://www.step-into-the-future.ru), под руководством школьных или вузовских преподавателей. Авторы этих работ должны продемонстрировать свои способности и творческие возможности решения достаточно сложных для среднестатистического школьника задач.

Участники заключительного этапа проходят предварительный отбор в регионах Российской Федерации и последующее рецензирование научно-исследовательских работ, рекомендованных региональными представителями программы «Шаг в будущее» для участия в финале отборочного этапа. Рецензирование работ проводится силами преподавателей МИРЭА, работающих на различных кафедрах, связанных с информатикой и информационными технологиями. Большинство рецензентов имеют ученые степени кандидатов и докторов наук и работают по профилю, соответствующему тематике рецензируемых работ.

Защита работ проводится в аудитории, оснащенной компьютерной техникой и проектором. Каждый участник выступает со своей презентацией. Ориентировочное время для доклада — 10 минут. После доклада автор защищает свою работу, отвечая на вопросы экспертов и присутствующих. При необходимости руководитель секции может выделить дополнительное время для завершения и обсуждения доклада.

Оценка работы проводится экспертной комиссией секции в составе 5 ученых и специалистов, каждый из которых выставляет оценку за защиту работы (не более 20 баллов). Общая оценка (максимум 100 баллов) вычисляется как сумма оценок экспертов.

Тур «Решение профилированных заданий по предмету «Информатика». Задания по информатике охватывают следующие разделы школьного курса «Информатика и информационные технологии»: алгебра логики (понятия и основные законы, логические элементы), алгоритмизация и типы переменных, моделирование, формализация, оптимизация, использование случайных чисел, использование знаний по кодированию символов (кодовые страницы). Участники олимпиады также должны иметь знания из области школьных разделов геометрии (свойства треугольников, прямые в декартовой системе координат) и физики (электрические цепи), а также иметь представление о переменных, уметь разбивать решение задачи на отдельные этапы и составлять последовательность выполнения операций (алгоритмизация), уметь представлять выполнение алгоритма в виде блок-схемы или текста программы на одном из языков программирования (Pascal, Basic, C++). Для школьников 8—9 и 10—11 классов предлагаются разные варианты заданий.

На выполнение заданий по информатике отводится 4 часа (240 минут). Поскольку решение заданий по информатике не сводится только к программированию, все олимпиадные задания выполняются в письменной форме на листах бумаги формата А4. Это позволяет проверить не только конечный результат, но и логику мышления участника олимпиады, его общие знания курса информатики, а также имеющиеся навыки и творческий потенциал.

Особенностью решения практически любой задачи по информатике и программированию является то, что у них, как правило, не бывает однозначных и единственных путей к получению верного решения.

Каждый школьник может решать задачу по-своему, используя различные средства.

Победители и призеры научно-образовательного соревнования олимпиады школьников «Шаг в будущее» определяются по сумме результатов защиты работ и выполнения заданий по информатике (максимально возможный балл — 200).

Академическое соревнование по предмету «Информатика» проводится в виде олимпиады, на которой участникам предлагается выполнить ряд заданий по информатике разного характера и сложности.

Олимпиада проводится в два этапа: отборочный и заключительный (очный). Задания по информатике основаны на тех же темах и разделах школьного курса «Информатика и ИКТ», что и для тура «Решение профилированных заданий по предмету «Информатика» научно-образовательного соревнования.

В ходе подготовки к олимпиаде школьники могут ознакомиться с макетными вариантами заданий олимпиады и списком рекомендуемой литературы на сайтах вузов, проводящих олимпиаду, чтобы иметь представление о характере будущих задач. В состав заданий входят задачи разного уровня сложности, чтобы участники олимпиады смогли попробовать свои силы, знания и способности творчески подходить к их решению. Решив относительно легкую задачу, можно переходить и к более сложным заданиям, требующим размышлений.

Количество заданий академического соревнования, уровень их сложности, а также критерии оценивания утверждаются председателем методической комиссии головного вуза по предмету. На выполнение заданий по информатике отводится 4 часа (240 минут). Максимально возможная суммарная оценка за полностью выполненные задания составляет 100 баллов. Победители и призеры академического соревнования определяются по итогам проверки работ очного этапа с учетом апелляций.

■ УСЛОВИЯ ЗАДАЧ

В предлагаемом вниманию читателей кратком обзоре приведены условия задач по информатике, предлагавшихся на олимпиадах «Шаг в будущее» в 2010—2011 гг. В отдельном разделе книги показаны и подробно прокомментированы варианты возможных решений нескольких таких задач.

Следует отметить, что в решении большинства задач по программированию важную роль играет выбор модели решения. Все, кто хотя бы немного знаком с программированием, знают, что любую задачу можно реализовать в виде программы различными способами. Если задача решена верно, то ответы в различных реализациях программы будут одинаковыми, но одни вари-

анты программы будут меньше по объему программного кода, а другие могут выполняться быстрее. Жюри всегда приветствуется более короткое и «красивое» решение.

1. Последовательность

Дана последовательность, состоящая из записанных подряд натуральных чисел:

12345678910111213141516171819202 1... и т. д.

Вывести на печать N-ю цифру данной последовательности.

Формат входных данных

Вводится натуральное число N.

Формат выходных данных

Выведите N-ю цифру последовательности.

Пример

Входные данные	Выходные данные
21	5

2. Схема

Упростить схему, состоящую из логических элементов (рис. 4), используя понятия и законы алгебры логики. X, Y, Z — логические переменные (входные двоичные сигналы).

Входные данные

Логическая схема, изображенная на рисунке.

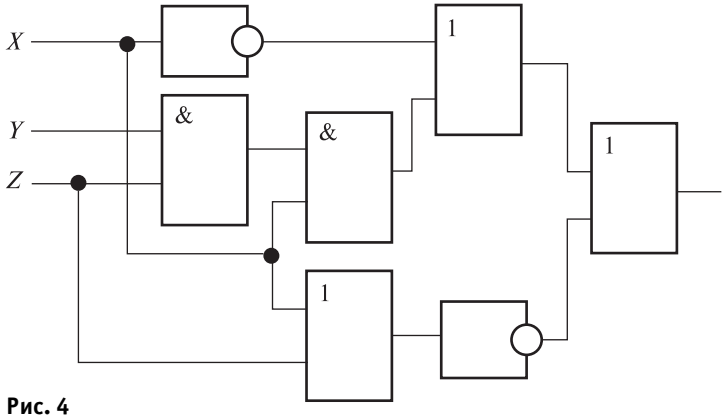


Рис. 4

Выходные данные

Упрощенный вариант логической схемы, реализующей ту же самую логическую функцию.

Примечание. Элементы со значком «&» реализуют логическую функцию «И», элементы со значком «1» реализуют логическую функцию «ИЛИ», оставшиеся элементы без обозначений реализуют логическую функцию «НЕ».

3. Электрическая цепь

Электрическая цепь состоит из N одинаковых звеньев с однотипными конденсаторами одинакового номинала C мкФ (C — целое число) (рис. 5).

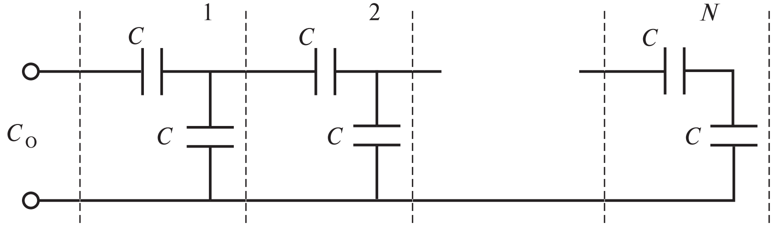


Рис. 5

Разработать программу (блок-схему алгоритма) расчета общей (входной) емкости C_O данной электрической цепи с точностью до трех знаков после запятой.

Формат входных данных

Количество звеньев N и номинал конденсаторов C (мкФ).

Формат выходных данных

Величина общей емкости цепи C_O (мкФ).

4. Радуга

На шахматной доске в произвольном порядке располагаются цветные кубики с размером грани, совпадающим с размером клетки доски. При этом на линиях, обозначаемых на доске буквами, располагаются кубики одного цвета в таком порядке: на линии a — белые, а на линиях b—h — окрашенные в семь основных цветов радуги от красного до фиолетового. На каждой линии может быть от 1 до 3 кубиков.

Разработать программу (блок-схему алгоритма) расстановки кубиков на доске и определения отсутствующих цветов радуги на левой проекции доски с кубиками.

Формат выходных данных

При каждом запуске программы должно происходить произвольное автоматическое заполнение шахматного поля цветными кубиками в соответствии с условиями задачи.

На экран должно быть выведено положение кубиков на шахматной доске по образцу, указанному в примере (буквами Б, К, О, Ж, З, Г, С, Ф обозначены позиции кубиков соответствующих цветов, символом «*» — пустые клетки), а также текстовое сообщение: «На левой проекции отсутствуют цвета: <*перечислить через запятую цвета кубиков, отсутствующие на левой проекции*>».

Пример

Входные данные	**Выходные данные**								
—		*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*
	1	*	*	*	Ж	*	Г	*	Ф
	2	Б	К	*	*	З	*	С	*
	3	*	*	О	*	*	*	*	Ф
	4	*	*	*	*	*	Г	*	*
	5	*	К	*	*	З	*	С	*
	6	*	*	*	*	*	*	*	Ф
	7	*	К	*	*	*	*	*	*
	8	Б	*	О	*	*	*	С	*
	На левой проекции отсутствуют цвета: зеленый, синий								

5. «Бильярд Льюиса Кэрролла»

На прямоугольном бильярдном столе (рис. 6) размером $H \times L$ (где H и L — целые числа), имеющем только угловые лузы, из левой нижней лузы под углом 45° к боковым стенкам вылетает шар. Он ударяется в борт, отскакивает, ударяется еще раз и т. д. до тех пор, пока не попадет в одну из боковых луз. Угол падения шара к борту и угол отскока считать равными.

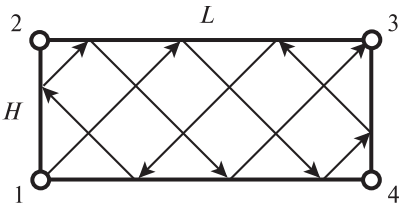


Рис. 6

Составьте программу (блок-схему алгоритма), позволяющую определить:

1) из скольких отрезков будет состоять ломаная траектория движения шара до попадания в лузу;

2) в какую лузу попадет шар.

Формат входных данных

Размеры стола H и L.

Формат выходных данных

Два числа — количество отрезков траектории шара и номер лузы, в которую он попадет.

МЕЖДУНАРОДНЫЕ ОЛИМПИАДЫ. ТУРНИР АРХИМЕДА

■ ОРГАНИЗАЦИОННАЯ ИНФОРМАЦИЯ

Сайт: http://informatics.mccme.ru/arhimed

Сроки проведения: ежегодно в конце апреля — начале мая в Москве и других регионах РФ, а также в странах СНГ.

Турнир Архимеда — это цикл соревнований по математике и информатике, организуемых группой учителей г. Москвы совместно с преподавателями и студентами московских вузов. Общая особенность всех таких соревнований — их открытость как для школьников, так и для преподавателей. В личных соревнованиях может участвовать любой школьник, а в командных — любая школа, подавшая заявку в установленный срок. Кроме того, все желающие учителя могут участвовать как в подборе задач, так и в проверке работ учащихся.

Турнир Архимеда по информатике — это командная олимпиада, возникшая в 2007 г., преследующая те же цели и ставящая те же задачи, что и математические соревнования Турнира Архимеда. Она предназначена для школьников, еще только начинающих изучение программирования, причем, в отличие от математических соревнований, возраст участников никак не ограничивается: ведь кто-то из них начал изучать программирование в 7 классе или еще раньше, а кто-то — только в 10 классе.

Олимпиада проводится очно, в течение одного дня. Программа мероприятия включает:

- знакомство участников с правилами турнира;
- знакомство с компьютером и тестирующей системой (пробный тур);
- трехчасовой основной (компьютерный) тур;
- разбор задач;
- награждение победителей.

При этом организаторы олимпиады стараются рассказать школьникам и о других соревнованиях по программированию, в которых они могут принять участие, а также о доступных им кружках по программированию.

Олимпиада проводится в конце учебного года, в конце апреля — начале мая. Выбор этих сроков не случаен: к этому времени даже школьники, изучающие программирование первый год, уже могут продемонстрировать полученные навыки и умения.

Задачи для турнира подбираются так, чтобы, с одной стороны, для их решения не требовались какие-то специальные знания (в частности, по математике), недоступные семи-восьмиклассникам, но, с другой стороны, они демонстрировали все многообразие приемов и ситуаций, встречающихся на более продвинутых соревнованиях. Поэтому разбор задач является очень важной частью турнира: даже если школьники решили какую-то задачу самостоятельно, в ходе ее разбора они могут узнать немало нового и интересного. Кроме того, организаторы турнира стараются подобрать задачи так, чтобы даже школьникам, незнакомым с отдельными базовыми темами (например, с обработкой строк или двумерных массивов), все же было доступно большинство задач. Некоторые задачи почти не требуют программирования, их решение может быть записано буквально в одну строку, но при этом требуют грамотного математического или логического анализа ситуации.

В Москве турнир ежегодно проводится на базе Физико-математической школы № 2007 (где турнир и был придуман в 2007 г.). С 2010 г. к проведению турнира подключились и другие регионы: так, третий турнир (2010 г.) прошел уже в 44 регионах России, Беларуси и Украины и собрал 184 команды (более 500 участников с 4 по 11 класс). При этом центральный оргкомитет готовит все материалы для проведения турнира (условия задач, тесты и проверяющие программы, решения и разборы задач, презентации для проведения разборов), а местные организаторы собирают участников, обеспечивают проведение олимпиады в своем регионе (городе, школе и т. д.) и награждение победителей. Как уже было сказано, турнир проводится в конце апреля — начале мая, и каждый регион выбирает удобные для местных условий конкретные даты проведения мероприятия. После окончания турнира во всех регионах на сайте публикуется сводная таблица результатов.

Автоматическая проверка решений на олимпиадах по программированию

В последние годы стандартом де-факто стал следующий формат проведения олимпиад по программированию. Школьнику (или команде) предоставляется компьютер и набор задач (обычно от 3 до 6 на личных олимпиадах и от 8 до 12 — на командных). При этом в каждой задаче участник должен написать программу на одном из допустимых язы-

ков программирования. Как правило, организаторы стараются, чтобы перечень допустимых языков учитывал запросы всех участников, хотя по техническим причинам не всегда удается обеспечить проверку решений на всех таких языках.

Программа, являющаяся решением задачи, должна читать входные данные из текстового файла или из стандартного потока ввода (с клавиатуры), выполнять требуемые действия и выводить ответ в текстовый файл или в стандартный поток вывода (на экран). При этом входные данные подготовлены в точном соответствии с форматом, описанным в условии задачи, а вывод результатов также должен осуществляться в строгом соответствии с описанным форматом (в частности, не должно выводиться ничего лишнего). Программа, предложенная участником в качестве решения, должна работать для любых входных данных, удовлетворяющих ограничениям, описанным в условии, причем время ее работы не должно превышать указанного в условии (как правило, 1—2 с, реже — 3—5 с), а объем всей используемой программой памяти не должно превосходить заданный предел (как правило, 64, 128 или 256 Мб).

Написанная программа сдается участником для проверки в *тестирующую систему* — комплекс программ, специально разработанных для проведения олимпиад по программированию. Сдача решения происходит через локальную сеть или Интернет, при этом в качестве клиента используется браузер или специальное приложение. В тестирующую систему пересылается *исходный код* программы, а не исполняемый файл. Тестирующая система компилирует (при необходимости) исходный код и запускает программу, проверяя ее на наборе заранее подготовленных тестовых примеров (одних и тех же для всех участников). Одновременно тестирующая система контролирует объем используемой памяти и прерывает выполнение программы по истечении указанного времени. Участнику олимпиады сами эти тестовые примеры недоступны, но демонстрируется протокол проверки (информация о результате прохождения каждого теста). Далее, в зависимости от правил конкретной олимпиады, по результатам прохождения тестов начисляются баллы и определяется победитель.

Поскольку в задаче может быть несколько правильных ответов, для каждой программы пишется отдельная проверяющая программа (*checker*), которая по входным данным тестового примера, правильному ответу (или другой дополнительной информации) и ответу, выдаваемому программой участника, определяет, успешно ли пройден тест.

Особенности языков программирования при использовании на командных олимпиадах

Традиционными языками программирования на школьных олимпиадах считаются Паскаль и Си. Именно в расчете на них готовятся задачи олимпиады. Поэтому при решении задач на других языках могут возникать проблемы (например, решение может не укладываться в ограничения по времени или по объему памяти), но вместе с тем участ-

ники могут получить и определенные преимущества благодаря более широким возможностям другого языка и его стандартных библиотек. Традиционно на олимпиадах по информатике разрешается использование всех библиотек, входящих в стандартную поставку соответствующего компилятора.

При решении задач на Паскале чаще всего используются компиляторы Free Pascal, Delphi и Borland (Turbo) Pascal. В некоторых олимпиадах в последнее время также поддерживается учебный пакет PascalABC. Язык Паскаль с точки зрения его применения к олимпиадным задачам отличается простотой изучения, удобством при отладке задач, но обладает небольшим набором стандартных функций и несколько избыточным синтаксисом (например, по сравнению с языком Си).

При решении задач на языке Си/Си++ используются компиляторы GNU C/C++ и Microsoft Visual C++ (реже — Borland C++ 3.1). Для работы участникам обычно предлагаются среды MinGW, Dev-C++, Code::blocks, Visual Studio, реже — Eclipse. Язык Си отличается высокой скоростью работы программ, кратким синтаксисом и обширной стандартной библиотекой (STL), активно используемой для решения сложных олимпиадных задач. Однако отладка программ на этом языке представляет собой несколько более трудный процесс, чем, например, на языке Паскаль.

Некоторых школьников до сих пор учат программировать на языке QBasic. Для проверки задач на этом языке в автоматической тестирующей системе написанная участником программа должна считывать данные из файла и записывать их также в файл, а исходный код для сдачи в тестирующую систему должен быть сохранен как текстовый файл. Кроме того, следует помнить, что не все задачи олимпиады можно решить, используя QBasic.

При использовании Visual Basic или Visual C# (если эти языки поддерживаются на олимпиаде) необходимо создавать *консольное приложение*.

Программы на языке Java обычно работают несколько медленнее, чем на языке Паскаль или Си, и иногда, если это не было особо предусмотрено организаторами олимпиады, могут не уложиться в отведенный лимит времени. Преимуществом же Java является обширный набор функций в его стандартных библиотеках.

В последнее время среди школьников набирает популярность язык программирования Python. Тем, кто хочет представить решение на этом языке, следует обратить внимание на то, какая именно его версия (2 или 3) поддерживается на данной олимпиаде, поскольку синтаксис языка в этих версиях различен. Кроме того, нужно учитывать, что программы на Python в отдельных случаях могут работать в десятки раз медленнее программ на других языках, хотя по краткости и удобству написания кода этому языку, пожалуй, нет равных среди вышеперечисленных. Python также обладает обширными стандартными библиотеками с большим числом предоставляемых ими возможностей.

Иногда на олимпиадах также поддерживаются другие языки программирования, такие как Perl, Ruby, PHP и т. д.

Официальные правила командной олимпиады

Олимпиада проводится по так называемой *системе ACM* — официальным правилам проведения студенческих командных чемпионатов мира, принятым на большинстве командных олимпиад (включая школьные). Согласно этим правилам, проверка сданных решений производится онлайн. При этом по ходу тура можно посылать решение на проверку неограниченное количество раз, оцениваются только полностью решенные задачи (успешно проходящие все тесты). Основным параметром при оценивании является количество решенных задач, а при равном их количестве учитывается так называемое «штрафное время», зависящее от времени, затраченного на решение каждой такой задачи и количества сделанных попыток.

Общие положения. Турнир Архимеда по программированию проводится среди школьных команд (без ограничения возраста учащихся).

Команды регистрируются заранее представителем из числа участников команды или другим контактным лицом (например, учителем). Каждая команда состоит из трех участников, однако по решению местных организаторов к соревнованию может быть допущена и команда, состоящая из меньшего числа участников.

Проведение соревнования. Каждая команда получает в свое распоряжение один компьютер. Составление исходных текстов решений и их отладка осуществляется участниками с использованием сред программирования, определяемых местными организаторами.

Накануне соревнования проводится пробный тур, в ходе которого участники могут ознакомиться с рабочими местами и тестирующей системой. Результаты пробного тура при подведении итогов соревнований не учитываются.

Участники могут приносить с собой и использовать любые источники информации на бумажном носителе — книги, справочники, листинги программ и пр. Запрещается использование любых электронных средств хранения или передачи информации — портативных компьютеров, калькуляторов, сотовых телефонов и пр.

Во время проведения соревнования взаимодействие команд и жюри осуществляется с помощью специального интерфейса тестирующей системы или в письменном виде.

Решение задач. Командам предлагается от 7 до 12 задач. Продолжительность тура составляет 3 часа, однако жюри имеет право продлить соревнования в случае каких-либо непредвиденных обстоятельств.

Во время соревнования команды решают предложенные задачи. Решением является программа (файл с исходным текстом), написанная на одном из разрешенных языков программирования. Команда может решать задачи на различных языках программирования.

Входные данные подаются программе в стандартном потоке ввода (считываются с клавиатуры). Программа должна выводить ответ в стандартный поток вывода (на экран). Вывод в стандартный поток ошибок запрещен. Формат ввода в точности соответствует описанному в условии задачи. Проверять корректность входных данных не требуется.

Размер исходного текста одной программы с решением не может превышать 64 Кб.

Команда может обратиться к жюри с вопросом по условию какой-либо задачи. Вопрос должен быть сформулирован на русском или английском языке и предполагать ответ «Да» или «Нет». Жюри может ответить на поставленный вопрос «Да», «Нет», «См. условие», если считает, что ответ на поставленный вопрос содержится в условии задачи, или «Без комментариев». Жюри также может разослать ответ на поставленный одной из команд вопрос всем участникам соревнования.

Проверка решений. Проверка представленных командами решений проводится во время соревнований. Участники посылают решения в тестирующую систему с помощью предоставленного им программного обеспечения. При отправке решения команда выбирает компилятор, который будет использован жюри (все компиляторы работают под управлением ОС Linux).

Решения участников запускаются под ОС Linux. Решение проверяется путем запуска программы на наборе тестов, который недоступен участникам и является одинаковым для всех команд. Решение засчитывается, если программа выдает верные ответы на все тесты.

Первыми тестами в наборе всегда являются тесты из условия задачи; эти тесты идут в том же порядке, в котором они приведены в условии задачи.

В условии каждой задачи указывается максимальное время ее выполнения для одного теста. Если на одном из тестов программа превысит это время, то решение считается неверным.

По мере готовности команда посылает свои решения в жюри для проверки и далее может продолжать работу над другими задачами. Результаты проверки отправленного решения доступны команде немедленно по завершении его проверки. При этом команде сообщается, зачтено ли решение, и если оно не зачтено, то сообщается тип ошибки и номер теста, на котором произошла ошибка.

При возникновении ошибки на этапе компиляции программа уже не проверяется ни на одном тесте.

Решения одной и той же команды с идентичным исходным текстом, сделанные подряд, игнорируются. Если решение было признано неправильным из-за ошибки компиляции, то такое решение не учитывается при последующем вычислении штрафного времени для успешно сданных задач.

Определение победителей. Полную и окончательную ответственность за проверку правильности представляемых решений несет жюри. Все решения жюри окончательны и обжалованию не подлежат.

Команды ранжируются по количеству решенных (прошедших все тесты) задач. Команды, решившие одинаковое число задач, ранжируются по суммарному времени их решения.

Суммарное время решения определяется как сумма времени решения каждой успешно сданной задачи. Время решения задачи при этом определяется как время от начала соревнования до момента отправки решения, признанного правильным, плюс 20 штрафных минут за каждое забракованное решение. Задачи, не признанные решенными к мо-

менту окончания соревнования, никакого вклада в суммарное время не дают (в том числе и в виде штрафов за забракованные решения).

Все промежуточные результаты, объявленные в ходе соревнования, являются неофициальными. Официальные результаты объявляются, как правило, в тот же день по окончании соревнования.

Итоги подводятся отдельно в каждом месте (регионе) проведения. По окончании турнира во всех регионах на сайте турнира публикуется сводная таблица результатов для всех участников.

■ УСЛОВИЯ ЗАДАЧ

Ниже приведены условия задач из первых трех турниров (всех, проведенных к моменту написания данной книги), причем задачи упорядочены не по номеру турнира, а по сложности (хотя эта оценка, конечно, достаточно субъективна).

Отдельно во второй части книги приведены разборы решений этих задач, сопровождаемые фрагментами исходного кода программ. При этом в разборах задач используется язык Python. Это сделано по нескольким причинам.

Во-первых, код на языке Python максимально лаконичен (похож на псевдокод) и понятен: такие решения смогут понять школьники, пишущие программы на любых других языках. В этом им помогут минимальные комментарии к текстам программ, в том числе добавленные в них в качестве «врезок».

Во-вторых, язык Python приобретает все большую популярность, в том числе при обучении школьников.

Наконец, в-третьих, школьники, в том числе владеющие другими языками программирования, прочитав лишь разборы приведенных ниже задач, смогут решать на языке Python многие другие олимпиадные задачи (т. е. разборы этих задач составляют своего рода «мини-учебник» по языку Python).

1. «У моего папы больше денег» (турнир III, задача C)

Двое играют в такую игру. Первый игрок называет число, а затем второй также называет свое число. Если число второго игрока больше, то он выиграл, в противном случае (даже если числа равны) выигравшим считается первый игрок. Помогите второму игроку — напишите программу, которая будет успешно играть в эту игру.

Формат входных данных

Натуральное число A, которое назвал первый игрок (в числе A не больше 100 цифр).

Формат выходных данных

Одно натуральное число — какой-нибудь (любой) выигрышный ход второго игрока.

Примеры

Входные данные	Выходные данные
1	5
1000000000000000	1000000000000001

2. Карточки (турнир I, задача I)

Вася выложил в ряд слева направо 100 карточек, на которых написаны числа 1, 2, 3, ..., 100 соответственно (числами вниз). После этого он поменял местами карточки, на которых написаны числа i и j. Петя открывает карточки по очереди слева направо. Какое минимальное количество карточек ему придется открыть, чтобы точно выяснить, какие карточки поменял местами Вася?

Формат входных данных

Два числа — i и j во входном файле. Числа записаны через пробел.

Формат выходных данных

Одно число — минимальное количество карточек, которое достаточно открыть Пете.

Ограничения

Во всех тестовых примерах натуральные числа i и j различны и лежат в пределах от 1 до 100.

3. Кольцевая (турнир III, задача F)

Два автомобиля движутся по кольцевой дороге длины L в противоположных направлениях. Они начинают движение из одной точки и едут с постоянными скоростями v_1 и v_2 соответственно. Требуется определить, на каком расстоянии друг от друга они окажутся в момент времени T.

Формат входных данных

Четыре натуральных числа L, v_1, v_2, T, разделенных пробелами. Все числа не превосходят 100.

Формат выходных данных

Расстояние между автомобилями в момент времени T — длина кратчайшей из двух дуг дороги между автомобилями.

Примеры

Входные данные	Выходные данные
10 1 2 1	3
10 2 3 2	0

4. Сумма цифр (турнир II, задача G)

Подсчитайте количество натуральных чисел на отрезке от a до b, сумма цифр которых четна.

Формат входных данных

Два натуральных числа a и b, не превосходящие миллиарда ($a \leqslant b$).

Формат выходных данных

Одно число — количество чисел с четной суммой цифр, больших либо равных a и меньших либо равных b.

Примеры

Входные данные	Выходные данные
1 5	2
10 10	0

5. Бегуны (турнир I, задача E)

Два бегуна тренируются на кольцевой дорожке легкоатлетического стадиона длиной 400 м. Они начинают бег из одной точки и бегут по заданиям тренера указанное число минут с указанной скоростью (в метрах в минуту). Требуется определить расстояние, на котором бегуны окажутся друг от друга в конце тренировки (длину более короткой части дуги дорожки между спортсменами).

Формат входных данных

В первой строке вводится одно число — количество заданий, которые каждый спортсмен получил от тренера (оба спортсмена получили одинаковое количество заданий). В каждой из следующих строк записаны задания для спортсменов в следующем формате:

$$v_1\ t_1\ v_2\ t_2,$$

где v_1 — скорость для первого игрока (м/мин), t_1 — время, на протяжении которого спортсмен должен бежать со скоростью v_1; v_2 и t_2 — соответствующие величины для второго бегуна. Скорости — положительные числа, если требуется бежать в направлении по часовой стрелке, и отрицательные, если требуется бежать против часовой стрелки.

Формат выходных данных

Одно число — расстояние между бегунами в конце тренировки.

Примеры

Входные данные	Выходные данные	Комментарий
2 8000 1 8000 1 8000 10 8001 10	10	Тренер выдал каждому спортсмену по два задания. Сначала они оба бегут 1 мин со скоростью 8000 м/мин по часовой стрелке, а затем 10 мин — один со скоростью 8000 м/мин, а второй со скоростью 8001 м/мин. Поэтому в конце тренировки расстояние между ними будет равно 10 м (так как во второй части тренировки второй спортсмен каждую минуту пробегает на один метр больше)
1 50 4 –100 1	100	Первый бегун пробегает по 50 м за минуту в течение 4 мин, т. е. всего 200 м. Второй бегун пробегает за минуту 100 м в противоположном направлении (знак «–»). Поскольку длина круга равна 400 м, то расстояние между бегунами в конце тренировки составит 100 м
4 1000 4 1000 4 2000 1 2000 1 1223 7 1223 7 1 1 1 1	0	Поскольку бегуны *всегда* получали одинаковые задания от тренера, они окажутся на финише одновременно. Расстояние между ними будет равно нулю

Ограничения

Все входные данные — целые и по модулю не превосходят 30 000. Количество заданий — не меньше 1.

6. Тапочки (турнир I, задача H)

У меня в прихожей стоят в ряд 20 тапочек — 10 левых и 10 правых. Приходя домой, я переобуваюсь и выбираю два тапочка — левый и правый, в которые мне удобнее всего засунуть ноги. Естественно, что левый тапочек должен стоять левее правого и расстояние (количество других тапочек) между ними должно быть как можно меньше. Напишите программу, которая вычисляет, сколько тапочек стоит между теми, которые мне удобнее всего надеть.

Формат входных данных

Во входном файле записано 10 нулей и 10 единиц в некотором порядке. Единица соответствует левому тапочку, 0 — правому. Числа разделены пробелами.

Формат выходных данных

Количество тапочек между самыми удобными тапочками или –1, если таких нет.

Пример

Входные данные	Выходные данные	Комментарий
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	0	Можно, например, надеть первый (левый) и второй (правый) тапочек, между которыми нет других тапочек

7. Робот (турнир II, задача A)

В левом нижнем углу доски размером $N \times M$ клеток стоит Робот. Он может ходить на одну клетку по горизонтали, вертикали или диагонали. Требуется переместить Робота в правый верхний угол за наименьшее количество ходов.

Формат входных данных

Два натуральных числа: N (высота доски) и M (ширина доски), не превышающие 100.

Формат выходных данных

Последовательность ходов *в одном из возможных* кратчайших путей. Каждый ход обозначается заглавной латинской буквой:

U — вверх,

R — вправо,

D — вверх и вправо.

Буквы выводятся без пробелов в одной строке.

Пример

Входные данные	Выходные данные
3 2	DU

8. Квартиры (турнир II, задача B)

В доме несколько подъездов. В каждом подъезде одинаковое количество квартир. Квартиры нумеруются подряд, начиная с единицы. Может ли в некотором подъезде первая квартира иметь номер x, а последняя — номер y?

Формат входных данных

Два натуральных числа x и y ($x \leqslant y$), не превышающие 10 000.

Формат выходных данных

Слово YES (заглавными латинскими буквами), если такое возможно, и NO — в противном случае.

Примеры

Входные данные	Выходные данные
11 15	YES
2 10	NO

9. Турнир (турнир II, задача F)

В однокруговом турнире участвовало N команд (каждая сыграла с каждой по одному матчу). Победителями считаются все команды, которые выиграли *не меньше* партий, чем каждая из остальных. Какое наибольшее количество победителей может быть в таком турнире?

Формат входных данных

Одно натуральное число, не превосходящее 1000, — количество команд.

Формат выходных данных

Одно число — максимальное количество победителей.

Пример

Входные данные	Выходные данные
2	1

10. Автомат по продаже билетов (турнир I, задача J)

В Московском метрополитене вновь появляются автоматы для продажи билетов. Вас просят написать программу, которая будет рассчитывать, какую сдачу и какими купюрами и монетами требуется выдать пассажиру.

Формат входных данных

Во входном файле записана сначала стоимость билета, который хочет приобрести пассажир, затем общее количество купюр и монет, которые он опустил в автомат, а затем достоинства каждой из этих купюр и монет. Входные данные записаны в одной строке и разделены пробелами. Известно, что сумма всех купюр больше, чем стоимость билета.

Формат выходных данных

Программа должна вычислить, какими купюрами и монетами можно выдать сдачу, и вывести достоинство каждой из этих купюр или монет в произвольном порядке. Автомат может выдавать сдачу купюрами в 10, 50, 100 и 500 р., а также монетами в 1, 2 и 5 р. Если решений несколько, то требуется выдать одно любое из них. Если решений нет, то требуется выдать текст:

Sorry! Our monetary system is not perfect!
Please, choose another way to pay!
Thank you!

Примеры

Входные данные	Выходные данные	Комментарий
100 1 500	100 100 100 100	Пассажир хочет купить билет стоимостью 100 р. Для этого он опустил в автомат одну купюру достоинством 500 р. и получил в качестве сдачи 4 купюры по 100 р. Возможны и другие варианты ответа, например: 100 100 100 50 10 10 10 10 10
100 5 1 10 1 10 100	1 10 1 10	Пассажир хочет купить билет стоимостью 100 р. Но почему-то он опустил в автомат не только купюру в 100 р., но и еще две монеты по рублю и две купюры по 10 р. Этот излишек и будет возвращен ему в качестве сдачи. Возможны и другие варианты ответа

Ограничения

Во всех тестовых примерах стоимость билета — натуральное число, не превосходящее 1000 р., количество купюр и монет не более 50, достоинство каждой не превосходит 500 р. Общая сумма денег, опущенная в автомат покупателем, превосходит стоимость билета.

11. Конь (турнир II, задача D)

На доске размером $K \times N$ клеток (K строк, N столбцов) в i-й строке и j-м столбце стоит шахматный конь. Может ли он за один или несколько ходов попасть в клетку в s-й строке и m-м столбце?

Формат входных данных

Шесть натуральных чисел: K, N, i, j, s, m ($1 \leqslant K \leqslant N \leqslant 100$). Клетки ($i$, j) и (s, m) не совпадают.

Формат выходных данных

Слово YES, если такое возможно, и NO — в противном случае.

Пример

Входные данные	Выходные данные
8 8 1 2 7 8	YES

12. Уравнение (турнир I, задача B)

Решите в целых числах уравнение $\sqrt{ax + b} = c$, где a, b, c — заданные целые числа: найдите все решения или сообщите, что решений в целых числах нет.

Формат входных данных

Три числа a, b и c, разделенные пробелами.

Формат выходных данных

Все решения уравнения *в порядке возрастания* либо текст NO SOLUTION (заглавными буквами), если решений нет. Если решений бесконечно много, нужно вывести текст MANY SOLUTIONS.

Примеры

Входные данные	Выходные данные	Комментарий
1 0 0	0	Уравнение $\sqrt{x} = 0$ имеет единственный корень $x = 0$
1 2 3	7	Уравнение $\sqrt{x + 2} = 3$ имеет единственное решение $x = 7$
1 2 –3	NO SOLUTION	Уравнение $\sqrt{x + 2} = -3$ решений не имеет

Ограничения

Во всех тестовых примерах все входные данные — целые числа, не превосходящие 100.

13. Бесконечная таблица (турнир I, задача C)

Натуральные числа записаны в (бесконечную) таблицу, как показано на рис. 7. Требуется по заданному числу вывести всех его соседей (числа, записанные в клетках сверху, справа, слева и снизу, если таковые имеются).

Формат входных данных

Одно натуральное число.

Формат выходных данных

Все числа, записанные в соседних клетках с данным, *в порядке возрастания*. Числа должны разделяться пробелом.

					17	...
				10	18	...
			5	11	19	...
	2		6	12	20	...
1	3		7	13	21	...
	4		8	14	22	...
			9	15	23	...
				16	24	...
					25	...
						...

Рис. 7

Примеры

Входные данные	Выходные данные	Комментарий
1	3	Снизу, сверху и справа от 1 чисел нет, а справа стоит число 3
7	3 6 8 13	См. рис.
10	11 18	См. рис.

N=10

–	–	17
–	**10**	18
5	11	19

N=7

2	6	12
3	**7**	13
4	8	14

Ограничения

Во всех тестовых примерах заданное число не превосходит 1 000 000 000. (В Borland Pascal для работы с большими целыми числами вместо типа **integer** используйте тип **longint**.)

14. Теннис (турнир II, задача H)

Вася, Петя и Коля играли в теннис на вылет (проигравший пропускал следующую партию, уступая свое место третьему). Вася утверждает, что сыграл x партий, Петя — что сыграл y партий, Коля — z партий. Определите, могло ли такое быть.

Формат входных данных

Три целых неотрицательных числа x, y, z, не превосходящих 1000.

Формат выходных данных

Слово YES (заглавными буквами), если такое могло быть, и NO — в противном случае.

Примеры

Входные данные	Выходные данные
3 1 2	YES
1 1 1	NO

15. Хорошие стихи (турнир III, задача G)

Вы когда-нибудь задумывались над тем, как отличить хорошие стихи от посредственных? Нет? А вот редактор литературного журнала занимается этим каждый день, получая тонны корреспонденции от молодых авторов, желающих стать известными поэтами. В последнее время большая часть стихов присылается по электронной почте, поэтому у редактора возникла мысль автоматизировать процесс. Он твердо уверен, что стихи тем лучше, чем точнее в них рифма. Он считает две строки зарифмованными, если у них совпадает несколько последних букв. И чем больше букв совпадает, тем лучше зарифмованы строки. Например, у строк «палка» и «веревка» совпадают только пары последних букв «ка», а у строк «олимпиада» и «рая и ада» совпадают четыре буквы (пробелы пропускаются). Поэтому вторая рифма лучше. Редактор считает, что в четверостишии (четыре строки) первая строка должна рифмоваться с третьей, а вторая — с четвертой. Для каждой из этих двух пар строк он подсчитывает количество совпадающих последних символов и из этих двух чисел выбирает наибольшее. Полученное число он называет *коэффициентом качества стихотворения* — чем он выше, тем больше шансов у стихотворения быть опубликованным. Помогите редактору — напишите программу, которая определяет качество стихотворения. И кто знает, может быть, благодаря вашим усилиям мир познакомится с гениальными стихами (см. первый пример ☺).

Формат входных данных

Четыре непустые строки, каждая из которых состоит из не более 100 строчных латинских букв (стихотворение уже подверглось предварительной обработке — из него удалили все пробелы и знаки препинания, а заглавные буквы сделали строчными).

Формат выходных данных

Одно число — коэффициент качества стихотворения.

Примеры

Входные данные	Выходные данные
yapomnyuchudnoemgnovenje peredomnojyavilasty kakmimoletnoevidenje kakgenijchistoykrasoty	4
eto vovse ne stihi	0
etootlichnyestihi etootlichnyestihi etootlichnyestihi etootlichnyestihi	17

16. Число (турнир II, задача C)

Вводится натуральное число. Требуется разделить запятыми тройки его цифр (считая справа).

Формат входных данных

Одно натуральное число, не превышающее 10^{100}.

Формат выходных данных

То же самое число с разделением троек цифр запятыми.

Примеры

Входные данные	Выходные данные
1000	1,000
12345678	12,345,678
999	999

17. Трудная задача из ЕГЭ (турнир III, задача A)

Петя долго готовился к сдаче ЕГЭ по информатике. Он научился решать все задачи, и лишь задачу A12 ему научиться решать не удалось. Но он надеется тайно пронести на экзамен ноутбук и просит вас написать программу, которая ему поможет. Вот как выглядит эта трудная задача в демоверсии варианта ЕГЭ 2010 г.:

A12. Витя пригласил своего друга Сергея в гости, но не сказал ему код от цифрового замка своего подъезда, а послал следующее SMS-сообщение:

«В последовательности цифр 3182 все цифры больше 5 разделить на 2 (при необходимости отбросив остаток), а затем удалить из полученной последовательности все четные цифры».

Какой код получил Сергей, выполнив указанные в сообщении действия?

Формат входных данных

Четыре цифры в одной строке без пробелов — последовательность, содержащаяся в SMS-сообщении.

Формат выходных данных

Код цифрового замка без пробелов.

Пример

Входные данные	Выходные данные
0586	53

18. Считалочка (турнир III, задача E)

Дети решили поиграть в догонялки и, чтобы выбрать водящего, встали в круг и стали считаться. Для этого они использовали считалочку. Показывая пальцем по очереди на каждого стоящего в кругу, считающий произносит одно слово, и тот, на кого придется последнее слово, будет водить. Требуется по заданной считалочке определить, кто будет водить.

Формат входных данных

В первой строке вводится считалочка. Она состоит из слов, записанных латинскими буквами. Слова разделены одним пробелом. Знаков препинания нет, строка начинается и заканчивается буквой. В считалочке не менее двух слов, а длина строки не превосходит 100.

Во второй строке в том же формате вводится список имен школьников в том порядке, в котором они стоят по кругу. Считать начинают с первого школьника. Детей не менее двух, а длина строки не превосходит 100.

Формат выходных данных

Имя школьника, которому предстоит водить.

Примеры

Входные данные	Выходные данные
To be or not to be John Mary Ann Kate	Mary
Na zolotom kryltse sideli Vasya Vasya Vasya	Vasya

19. Негласный палиндром (турнир I, задача G)

Возьмем произвольное слово и проделаем с ним следующую операцию: поменяем местами его первую согласную букву с последней согласной, вторую согласную букву — с предпоследней согласной и т. д. Если после этой операции мы вновь получим исходное слово, то будем называть такое слово *негласным палиндромом*. Например, слова sos, rare, rotor, gong, karaoke являются негласными палиндромами.

Требуется написать программу, которая по заданному слову определяет, является ли оно негласным палиндромом.

Формат входных данных

Одно слово.

Формат выходных данных

Слово YES, если введенное слово является негласным палиндромом, и NO — в противном случае.

Примеры

Входные данные	Выходные данные	Комментарий
tennete	YES	Первая согласная буква — t меняется местами с последней, тоже t; вторая согласная буква — n меняется местами с предпоследней, тоже n
karaoke	YES	Первая согласная буква — k меняется местами с последней, тоже k; вторая согласная буква является одновременно и предпоследней, поэтому остается на месте
disk	NO	Это слово не является негласным палиндромом, поскольку при указанных перестановках букв получается слово kisd

Ограничения

Во всех тестовых примерах слово записано строчными латинскими буквами и состоит не более чем из 20 символов.

20. Прыжки с трамплина (турнир III, задача H)

На соревнованиях по прыжкам на лыжах с трамплина техника прыжка оценивается пятью судьями. Каждый судья ставит оценку от 1 до 20, после чего одна наименьшая и одна наибольшая оценки отбрасываются.

Вам нужно написать программу, которая будет демонстрировать результаты прыжка для телетрансляции. Она должна выводить пять оценок, которые поставили судьи, не меняя их порядка, а затем их сумму, и при этом брать в скобки те оценки, которые не учитываются при расчете суммы.

Формат входных данных

Пять натуральных чисел от 1 до 20, разделенных пробелом.

Формат выходных данных

Те же числа выведите в том же порядке, взяв в скобки минимальное (а если их несколько — самое левое из них) и максимальное (а если их несколько — самое правое из них) число, а также сумму всех чисел, не взятых в скобки. Все числа (включая сумму) должны быть напечатаны в одной строке и разделены одним пробелом (внутри скобок пробелов быть не должно). Перед суммой должен стоять знак равенства, отделенный слева и справа одним пробелом. Порядок оценок должен быть такой же, как и во входных данных.

Примеры

Входные данные	Выходные данные
1 2 3 4 5	(1) 2 3 4 (5) = 9
10 11 10 11 10	(10) 11 10 (11) 10 = 31

21. Будильники (турнир II, задача E)

Будильник в сотовом телефоне можно настроить так, чтобы он звонил каждый день в одно и то же время либо в указанное время в определенный день недели. Независимо можно настроить несколько будильников. По информации о будильниках, текущему времени и дню недели определите, когда прозвонит очередной будильник.

Формат входных данных

В первой строке вводятся три числа, задающие текущее время: день недели (от 1 до 7), часы и минуты.

Во второй строке вводится одно натуральное число N, не превосходящее 100, — количество будильников.

В следующих N строках вводятся описания N будильников. Описание каждого будильника состоит из трех чисел: дня недели (число от 1 до 7 для понедельника... воскресенья соответственно или 0, если будильник должен звонить каждый день), часов (от 0 до 23) и минут (от 0 до 59).

Формат выходных данных

Три числа через пробел, задающие день недели, час и минуту, когда прозвонит ближайший будильник.

Примеры

Входные данные	Выходные данные
2 10 20 2 1 23 15 0 10 10	3 10 10
7 1 1 3 7 0 59 7 23 59 7 1 1	7 1 1

Комментарий. Во втором примере третий будильник прозвенит в заданный начальный момент времени.

22. Магический квадрат (турнир III, задача I)

Магическим квадратом называют таблицу, в которой записаны числа 1, 2, 3, ... по одному разу, так что суммы чисел в каждой строке и в каждом столбце равны. Мы расскажем вам об одном из методов построения магических квадратов (его называют *сиамским*). Он годится только для построения квадратов с нечетной стороной (3×3, 5×5, ...).

Поставим число 1 в верхнюю клетку центрального столбца. Далее будем двигаться по диагонали вправо вверх, расставляя в клетки последовательно числа 2, 3, 4, ... Если мы вышли за пределы таблицы вверх, то нужно перейти к нижней клетке того же столбца и продолжить с нее. Если мы вышли за правую границу, то нужно перейти к левой клетке той строки, куда мы должны

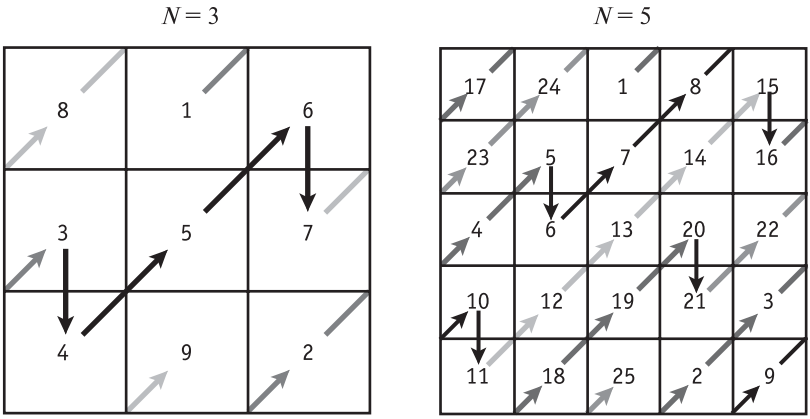


Рис. 8

были попасть. Если же мы одновременно вышли и вверх, и вправо, то нужно перейти в левую нижнюю клетку квадрата.

Если в следующей клетке на нашем пути уже стоит число, то вместо хода вправо вверх нужно сделать ход вниз (опять же, если мы при этом выйдем за границы квадрата, то нужно перейти к верхней клетке того же столбца). Примеры для квадратов 3×3 и 5×5 показаны на рис. 8.

Формат входных данных

Одно натуральное *нечетное* число N, не превосходящее 30, — размер квадрата.

Формат выходных данных

Числа, записанные в квадрате. Выравнивать числа по столбцам не обязательно. Обратите внимание: требуется вывести именно магический квадрат, полученный применением указанного метода.

Примеры

Входные данные	Выходные данные
5	17 24 1 8 15 23 5 7 14 16 4 6 13 20 22 10 12 19 21 3 11 18 25 2 9
1	1

23. Шахматное домино (турнир I, задача F)

Комплект шахматного домино состоит из 32 костяшек 2×1, каждый из квадратов которой окрашен в черный или белый цвет (часть костяшек состоит из двух белых квадратов, часть — из двух черных, а часть — из одного белого и одного черного). Комплект такого домино выложен на шахматную доску. Разрешается поворачивать костяшки домино на 180 градусов (менять местами их квадраты), оставляя каждую костяшку на своем месте. Требуется выяснить, можно ли так повернуть часть костяшек домино, чтобы в каждом горизонтальном ряду были квадраты только одного цвета.

Формат входных данных

Вводится 8 строк по 8 чисел. Каждое число соответствует номеру доминошки, которая покрывает данную клетку. Число положительное, если квадрат доминошки белый, и отрицательное, если он черный.

Формат выходных данных

Одно слово — YES или NO (заглавными буквами).

Примеры (для простоты примеры приводятся для доски 4×4, программа же должна работать только для доски 8×8)

Входные данные	Выходные данные	Комментарий
1 –1 2 2 4 4 5 5 6 6 7 7 8 8 3 3	NO	В левом углу лежит горизонтальная доминошка, квадраты которой разного цвета. 1 — квадрат белого цвета, –1 — квадрат черного цвета. Поэтому первую горизонталь нельзя сделать одноцветной
1 2 3 –4 –1 –2 –3 4 5 5 8 8 –7 –7 –6 –6	YES	Достаточно перевернуть доминошку 4

Ограничения

Доминошки нумеруются числами от 1 до 32 в произвольном порядке.

24. Деление с остатком (турнир III, задача D)

Вася учится делить с остатком. Он взял некоторое число, разделил его на 2 и отбросил остаток. То, что получилось, разделил на 3 и опять отбросил остаток. Полученное число он разделил на 4, отбросил остаток и получил число K. Какое число мог выбрать Вася изначально?

Формат входных данных

Натуральное число K, не превосходящее 1000.

Формат выходных данных

Все возможные числа, которые мог изначально выбрать Вася, по возрастанию, с разделением их пробелами.

Пример

Входные данные	Выходные данные
1	24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

25. Разложение на простые множители (турнир III, задача B)

Разложение на простые множители числа 12 можно записать тремя способами:

$$12 = 2 \cdot 2 \cdot 3 = 2 \cdot 3 \cdot 2 = 3 \cdot 2 \cdot 2.$$

Сколькими способами можно записать разложение на простые множители числа N?

Формат входных данных

Одно натуральное число N ($2 \leqslant N \leqslant 1000$).

Формат выходных данных

Одно число — количество различных записей разложения.

Примеры

Входные данные	Выходные данные
12	3
13	1

26. Вася (турнир I, задача D)

Вася давно мечтает выиграть олимпиаду по информатике. У него всего три слабых места: циклы, массивы и строки. Перед сегодняшним турниром Вася провел интенсивную подготовку, в ходе которой он решил A задач на циклы, B задач на массивы и C задач на строки. Впоследствии выяснилось, что из решенных задач D были и на циклы, и на массивы, E — на циклы и на строки, F — на строки и на массивы. И даже было G задач, которые включали и циклы, и строки, и массивы. Помогите Васе вычислить, сколько всего различных задач он решил.

Формат входных данных

Числа A, B, C, D, E, F и G, разделенные пробелами.

Формат выходных данных

Одно число — количество задач, решенных Васей.

Примеры

Входные данные	Выходные данные	Комментарий
0 0 0 0 0 0 0	0	
1 1 1 0 0 0 0	3	Вася решил по одной задаче на циклы, массивы и строки. При этом ни одной задачи на две темы сразу он не решил. Следовательно, он решил три разных задачи
1 1 1 1 1 1 1	1	Вася решил всего одну задачу на все три темы сразу

Ограничения

Во всех тестовых примерах все входные данные корректны и не превосходят 1000. Числа могут быть равны нулю.

ОЛИМПИАДЫ, ПРОВОДИМЫЕ ВУЗАМИ. МОСКОВСКИЙ ГОРОДСКОЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ. МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

■ ОРГАНИЗАЦИОННАЯ ИНФОРМАЦИЯ

Олимпиады по информатике проводятся в Московском городском педагогическом университете ежегодно.

В 2005—2008 гг. задание включало от 6 до 8 задач, на решение которых отводилось 3 или 4 часа. Большая часть заданий предполагала бескомпьютерное решение, и только в 2007—2008 гг. к ним было добавлено одно или два задания, выполняемых на компьютере с использованием одной из систем программирования (Visual Basic, Turbo Basic, QBasic, Borland C++, Turbo Pascal, Borland Delphi). При этом предполагается, что ввод (вывод) информации в программе, являющейся решением задачи, производится из файла (в файл) либо с клавиатуры (на экран).

К выполнению заданий по программированию участники допускаются только после сдачи бланка с решениями предыдущих заданий (всех или их части; бланк не должен был оставаться пустым).

Максимальная оценка работы участника составляет 100 баллов, причем половину этой суммы составляют баллы за составление программ (при наличии таких заданий). При равенстве суммы баллов победителем считается учащийся, правильно выполнивший наибольшее число заданий на компьютере.

■ УСЛОВИЯ ЗАДАЧ

Ниже приведены примеры задач (с комментариями к решениям и критериями их оценивания), предлагавшиеся на олимпиадах по информатике на математическом факультете МГПУ с 2005 по 2008 г.

Нетрудно заметить, что многие из этих заданий довольно тесно перекликаются с заданиями единого государственного эк-

замена по информатике, что лишний раз подчеркивает важность школьного олимпиадного движения: в ходе подготовки учащихся к участию в подобных олимпиадах одновременно осуществляется и их подготовка к сдаче ЕГЭ.

2005 год

1. Системы счисления

Число x в системе счисления с основанием p записывается как 4840. Известно, что это число делится нацело на десятичное число 11. Найти основание p. Рассмотреть два возможных случая:

1) $p < 11$;

2) $p < 31$.

Ответ обоснуйте.

Оценка за полное решение задачи — 10 баллов.

2. Терминология

Что такое *цилиндр* в контексте устройства компьютера? Какими параметрами он характеризуется? Дайте развернутый ответ.

Оценка за полное решение задачи — 10 баллов.

3. Логическая задача

Ночью к мосту подошла семья (папа, мама, малыш и бабушка). Папа может перейти мост за 1 минуту, мама — за 2, малыш — за 5, бабушка — за 10 минут. Мост выдерживает только двоих.

Есть один фонарик. Двигаться по мосту без фонарика нельзя. Светить издали нельзя. Нести друг друга на руках нельзя. Бросать фонарик друг другу нельзя. Если мост переходят двое, то они идут с меньшей из их скоростей.

Какое минимальное время требуется этой семье для перехода через мост? Обоснуйте ответ, указав последовательность перехода моста всеми членами семьи. Доказывать, что за меньшее время перейти мост нельзя, не требуется.

Оценка за полное решение задачи — 10 баллов.

4. Программирование

Чему будут равны значения переменных c и *d* после выполнения программы?

Школьный алгоритмический язык	Язык программирования Паскаль	Язык программирования Бейсик

c:=11; a:=24; b:=14;
d:=2*a-3
нц пока d >= b
| c:=c-1; d:=d-b
кц
	c:=11; a:=24; b:=14;
	d:=2*a-3;
	while d >= b do
	begin
	c:=c-1; d:=d-b
	end
		c=11
		a=24
		b=14
		d=2*a-3
		while d >= b
		c:=c-1
		d:=d-b
		wend

Оценка за полное решение задачи — 5 баллов.

5. Последовательности

Дано целое число k ($1 \leqslant k \leqslant 180$). Рассмотрим последовательность цифр 10111213...9899, получаемую выписыванием подряд всех двузначных чисел. Составьте программу для определения k-й цифры последовательности, не используя переменные строкового типа.

Формат входных данных

Вводится номер искомой цифры k.

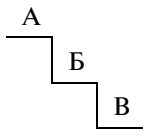
Формат выходных данных

Вывести цифру, стоящую в описанной последовательности на k-м месте.

Оценка за полное решение задачи — 15 баллов.

6. Три приятеля

Три приятеля — Андрей, Борис и Вадим — сидели друг за другом без головных уборов. При этом Борису и Вадиму было запрещено оглядываться назад, Борис видел голову сидящего перед ним Вадима, а Андрей — головы обоих своих приятелей.



Из мешка, содержащего две белые и три черные шапки (о чем все трое были осведомлены), каждому была надета шапка неизвестного для него цвета, а еще две шапки неизвестных для всех цветов остались в мешке.

Андрей заявил, что он не может определить цвет своей шапки. Борис услышал ответ Андрея и сказал, что у него тоже не хватает данных для определения цвета шапки у него на голове.

Мог ли Вадим на основании ответов своих приятелей определить цвет своей шапки? Ответ обоснуйте.

Оценка за полное решение задачи — 20 баллов.

7. Пересадки

В театре N мест, пронумерованных целыми числами от 1 до N. Некоторые зрители опоздали на спектакль, поэтому после третьего звонка зрители, которые имели билеты на неудобные места, пересели на более удобные. Опоздавшие, которые пришли уже после третьего звонка, садились на первое попавшееся свободное место.

В антракте один из опоздавших зрителей решил пересесть на свое место согласно билету. Если его место до этого было занято, тот, кто там сидел, тоже пересаживался на свое место. Если и там кто-то уже сидел, то и этот зритель также вынужден был перейти на свое место по билету, и т. д.

Поскольку в театр попали только зрители, имевшие на руках билеты, то начавшийся в антракте процесс пересаживания зрителей в какой-то момент закончился.

Составьте программу, определяющую, сколько человек в результате такого пересаживания были вынуждены поменять свое место (включая зрителя, инициировавшего весь этот процесс).

Формат входных данных

Считается, что входные данные уже находятся в памяти компьютера:

- переменная N — количество мест в зале;
- массив A содержит N целых чисел: в ячейке $A[1]$ — номер места, которое было записано в билете у человека, сидевшего на месте № 1, в ячейке $A[2]$ — номер места в билете у человека, сидевшего на месте № 2, и т. д. (если соответствующее место свободно, то в ячейке записано число 0);
- переменная Z — номер места в билете у человека, который решил после антракта занять свое место (инициировал процесс пересадок).

Формат выходных данных

Выведите количество человек, поменявших место в процессе пересадок.

Примеры

Входные данные	Выходные данные
$N = 10$ $A = (0,2,5,3,4,0,0,0,0,0)$ $Z = 4$	3
$N = 3$ $A = (1,2,3)$ $Z = 2$	0
$N = 3$ $A = (0,0,2)$ $Z = 2$	1

Оценка за полное решение задачи — 30 баллов.

2006 год

1. Системы счисления

Даны числа: 11101_2, 123_4, 34_8, 20_{16}. Запишите их в порядке возрастания.

Оценка за полное решение задачи — 5 баллов.

2. Единицы измерения информации

Решите систему уравнений

$$\begin{cases} 2^{x+2}\ (\text{бит}) = 8^{y-5}\ (\text{Кбайт}), \\ 2^{2y-1}\ (\text{Мбайт}) = 16^{x-3}\ (\text{бит}). \end{cases}$$

Оценка за полное решение задачи — 10 баллов.

3. Терминология

Опишите общие и отличительные черты *системного* и *прикладного программного обеспечения*. Дайте развернутый ответ.

Оценка за полное решение задачи — 10 баллов.

4. Программирование

Даны два целых положительных числа m и n. Напишите фрагмент программы, не содержащий условных операторов, в котором переменной x присваивается значение 1, если $n \leqslant m$, или любое другое число — в противном случае.

Формат входных данных

Вводятся целые положительные числа m и n.

Формат выходных данных

Вывести значение x.

Оценка за полное решение задачи — 15 баллов.

5. Последовательности

Дано целое число k ($1 \leqslant k \leqslant 150$). Рассмотрим последовательность цифр 101102103...149150, получаемую записью подряд всех трехзначных чисел от 101 до 150. Составьте программу для определения k-й цифры последовательности, не используя переменные строкового типа.

Оценка за полное решение задачи — 25 баллов.

6. Электронный замок

Электронный замок управляет открытием и закрытием ворот дворца. Каждый раз, когда кто-то хочет пройти через ворота, на электронном табло замка высвечивается число N. Затем предлагается ввести число M и последовательность из M знаков «+» или «–». После этого, если сумма чисел от 1 до M, полученная таким образом, что каждое число I участвует в суммировании с i-м введенным знаком, оказывается равным N, то ворота во дворец открываются, иначе на табло замка высвечивается новое число N.

Составьте программу, вводящую число N и выводящую число M и последовательность из M знаков «+» или «–», необходимых для открытия ворот. Если решений несколько, то достаточно вывести одно из них. Если для заданного N решения не существует, то программа должна вывести соответствующее текстовое сообщение.

Формат входных данных

Вводится исходное число N.

Формат выходных данных

Вывести число M, а затем M знаков «+» или «–», открывающих замок.

Примеры

Входные данные	Выходные данные	Примечание
5	5 + – – + +	1 – 2 – 3 + 4 + 5 = 5
4	3 – + +	–1 + 2 + 3 = 4

Оценка за полное решение задачи — 35 баллов.

2007 год

1. Системы счисления

В какой системе счисления разность $14^2 - 11^2$ равна десятичному числу 69? Ответ обоснуйте.

Оценка за полное решение задачи — 5 баллов.

2. Единицы измерения информации

Модем передает данные со скоростью 14 400 бит в секунду. Какова максимальная ширина прямоугольного изображения (в формате без сжатия), которое может передать модем за 3 минуты постоянной работы, если в изображении используется палитра из 65 536 цветов, а высота изображения составляет 400 точек?

Оценка за полное решение задачи — 5 баллов.

3. Високосный год

Год является високосным, если его порядковый номер кратен 4. Однако при этом годы, номера которых кратны 100, являются високосными только при условии, что их номера кратны 400.

Напишите логическое выражение, принимающее значение `ИСТИНА`, если год с порядковым номером N является високосным. Операция вычисления остатка от деления одного целого числа на другое обозначается как `mod`.

Оценка за полное решение задачи — 5 баллов.

4. Терминология

Раскройте понятие «системная шина» с точки зрения функционирования компьютера. Дайте развернутый ответ, проиллюстрируйте его с помощью схемы и укажите возможные характеристики системной шины.

Оценка за полное решение задачи — 10 баллов.

5. Программирование. Работа с текстом

Напишите программу для определения количества слов во введенном предложении. Предложение записано в одной строке, в нем нет тире и отдельно стоящих знаков препинания. Отдельно

стоящие числа, а также числа, после которых стоят знаки препинания, должны считаться словами. Для записи предложения могут использоваться символы русского и английского алфавитов.

Формат входных данных

Вводится строка, содержащая исходное предложение.

Формат выходных данных

Вывести натуральное число — количество слов в предложении.

Оценка за полное решение задачи — 10 баллов.

6. Программирование. Работа с массивами

Напишите программу для определения количества разных чисел в целочисленном массиве. Например, в массиве, содержащем числа 1 5 1 5 1 5 1 5, имеется два различных числа; в массиве 1 5 2 8 7 3 6 4 — восемь различных чисел; в массиве 6 6 6 6 6 6 6 6 — одно такое число.

Оценка за полное решение задачи — 15 баллов.

7. Сравнение чисел

Одна из основных операций с числами — их сравнение. Напишите программу, которая для двух заданных чисел сообщит, какое из них больше (или сообщит, что числа равны).

Формат входных данных

Входной файл `compare.in` состоит из двух строк, в каждой из которых записано по одному вещественному числу без ведущих нулей. Целая и дробная части отделяются точкой, которая может быть опущена, если число целое. Каждое из чисел содержит не более 50 цифр.

Формат выходных данных

Запишите в выходной файл `compare.out` символ «<», если первое число меньше второго; «>» — если больше; «=» — если числа равны.

Максимальное время работы на одном тесте: 3 с.

Максимальный объем используемой памяти: 64 Мб.

Примеры

Входные данные (compare.in)	**Выходные данные** (compare.out)
2.39 3.61	<
123 12.3	>
12345678 12345678.0	=
–.005 .0005	<

Оценка за полное решение задачи — 20 баллов.

8. Электронные часы

Петя очень любил наблюдать за электронными часами. Он целыми днями смотрел на часы и считал, сколько раз встречается каждая цифра. Через несколько месяцев он научился по любому промежутку времени говорить, сколько раз на часах за это время встретится каждая цифра, и очень гордился этим. Вася решил проверить Петю, но он не знает правильных ответов. Напишите программу, которая поможет Васе узнать, правильно ли считает Петя.

Формат входных данных

Первая и вторая строки входного файла clock.in содержат соответственно начало и конец промежутка времени. Начальное время не превосходит конечное. Время задано в формате *hh*:*mm*:*ss* ($00 \leqslant hh < 24$, $00 \leqslant mm < 60$, $00 \leqslant ss < 60$), где *hh*, *mm*, *ss* при необходимости дополняются ведущими нулями до двух символов (при подсчете эти нули также должны учитываться).

Формат выходных данных

Выходной файл clock.out должен содержать 10 чисел, где i-е число обозначает, сколько раз на часах встречается цифра $(i - 1)$ за указанный промежуток времени (начальный и конечный моменты также включаются в этот промежуток).

Максимальное время работы на одном тесте: 3 с.

Максимальный объем используемой памяти: 64 Мб.

Примеры

Входные данные (clock.in)	Выходные данные (clock.out)
23:59:58 23:59:59	0 0 2 2 0 4 0 0 1 3
13:24:09 13:24:40	5 45 45 45 36 3 3 3 3 4
00:00:00 00:00:02	16 1 1 0 0 0 0 0 0 0

Оценка за полное решение задачи — 30 баллов.

2008 год

1. Системы счисления

В классе 1000_x учеников, из них 120_x девочек и 110_x мальчиков. В какой системе счисления записано количество учеников?

Оценка за полное решение задачи — 5 баллов.

2. Единицы измерения информации (1)

Модем передает данные со скоростью 28 800 бит в секунду. Какова максимальная высота прямоугольного изображения (в формате без сжатия), которое может передать модем за 2 минуты постоянной работы, если в изображении используется палитра из 65 536 цветов, а ширина изображения составляет 500 точек?

Оценка за полное решение задачи — 5 баллов.

3. Логическое выражение

Укажите значения переменных K, L, M, N, при которых логическое выражение $(\neg K \vee M) \rightarrow (\neg L \vee M \vee N)$ **ложно**. Ответ запишите в виде строки из четырех символов: значений переменных K, L, M и N (в указанном порядке). Например, строка 1101 соответствует тому, что $K = 1$, $L = 1$, $M = 0$, $N = 1$.

Оценка за полное решение задачи — 10 баллов.

4. Единицы измерения информации (2)

Решите систему уравнений

$$\begin{cases} 2^{x-9}\ (\text{байт}) = 8^{y-7}\ (\text{Мбайт}), \\ 2^{2y-9}\ (\text{Кбайт}) = 4^{2x-5}\ (\text{бит}). \end{cases}$$

Оценка за полное решение задачи — 10 баллов.

5. Терминология

Дайте развернутое описание алгоритма, его свойств, исполнителей алгоритма, способов записи алгоритма и основных алгоритмических структур.

Оценка за полное решение задачи — 15 баллов.

6. Программирование

Напишите программу для решения следующей задачи: «Дано натуральное число N. Найти наибольшее число M, на которое сумма цифр в десятичной записи числа N делится без остатка. Если такого числа нет, то вывести слово НЕТ. ($M > 1$, $I < S$, где S — сумма цифр в десятичной записи числа N».

Задание выполняется без использования компьютера.

Формат входных данных

Вводится натуральное число N.

Формат выходных данных

Вывести число M — наибольший делитель суммы цифр числа N.

Примеры

Входные данные	Выходные данные
12345	5
917	НЕТ

Оценка за полное решение задачи — 25 баллов.

7. Программирование на компьютере

Вводятся координаты вершин треугольника и координаты некоторой точки (вещественные числа). Определить, лежит ли точка внутри треугольника. Если точка лежит на стороне треугольника, то считается, что она лежит вне его.

Оценка за полное решение задачи — 30 баллов.

■ ШКОЛЬНЫЕ ОЛИМПИАДЫ

1. Разность радикалов

Ответ: $\sqrt{1999} - \sqrt{1998} = 0{,}0118453496229$.

Комментарий. Идея этой задачи взята из книги С. В. Филичева «Занимательный Basic» [14]. Искомая разность может быть вычислена двумя разными способами: непосредственно или с помощью преобразования данного выражения через сумму сопряженных чисел, причем расхождение в результатах наблюдается лишь в 14-м знаке после запятой. Следовательно, чтобы правильно решить эту задачу, надо:

1) уметь работать с числами двойной точности;

2) при записи ответа отбросить знаки после запятой, начиная с 14-го.

```
CLS:COLOR 14,0
a$ = CHR$(251): c$ = CHR$(95): d$ = c$ + c$ + c$ + c$
e$ = "                         " + d$ + "    " + d$: PRINT e$
f$ = "    " + d$ + "           " + d$
b$ = a$ + "1999 - " + a$ + "1998"
PRINT "Вычисление разности:";b$;: COLOR 5,0
PRINT "x - непосредственный счёт, y - вычисление через сумму."
PRINT "Расхождение в ответах составляет величину: z = x - y."
a# = SQR(1999): b# = SQR(1998): PRINT
x# = a# - b#: y# = 1 / (a# + b#): z# = x# - y#: COLOR 6,0
PRINT "          Стандартный вид:          Десятичная запись:"
COLOR 9, 0:PRINT "x ="; x#, : PRINT " x = ";
PRINT USING "#.##################"; x#
PRINT "y ="; y#, : PRINT "  y = ";
PRINT USING "#.##################"; y#
PRINT "z ="; z#, : PRINT "  z = ";
PRINT USING "#.##################"; z#
PRINT : COLOR 2,0: PRINT f$
PRINT TAB(12); "Ответ: "; b$; " = ";
PRINT USING "#.################"; y#
LOCATE 13, 51: PRINT " ": END
```

2. Экстремальные значения

Ответ:

```
Экстремальные значения:

минимальное значение функции y = Sin (n) на [1; 1999], где n – на-
туральное число, равно -9.9999928077939803 при n = 7444

максимальное значение функции y = Sin (n) на [1; 1999], где n – на-
туральное число, равно .9999929216672852 при n = 7799
```

```
SCREEN 9: REM *** значение функции y=Sin(n) на [1;1999] ***
min = 1: max = -1: DIM y#(8000)
FOR n = 1 TO 8000
  y#(n) = SIN(n)
  IF y#(n) < min THEN
    min = y#(n): k = n
  END IF
  IF y#(n) > max THEN
    max = y#(n): t = n
  END IF
NEXT: PRINT
a$ = " значение функции y=Sin(n) на [1;1999], где n – натуральное"
b$ = " число равно ": c$ = " при n ="
a1$ = " минимальное ": a2$ = " максимальное "
COLOR 14, 0: PRINT a1$; a$: PRINT b$; y#(k); c$; k: PRINT
COLOR 15, 0: PRINT a2$; a$: PRINT b$; y#(t); c$; t: END
```

3. Синус 2000 раз

Ответ: $A = -0{,}0384684$.

```
CLS : pi = 3.141593: DIM y(2000)
x = 2000 * pi / 180: y(1) = SIN(x)
FOR i = 2 TO 2000
    y(i) = SIN(y(i - 1))
NEXT
PRINT "A = "; y(2000)
```

4. Шалтаи и болтаи

Ответ:

```
                Задача про шалтаев и болтаев.
1   Цена одного шалтая 46 коп., а цена одного болтая 64 коп.
2   Цена одного шалтая 56 коп., а цена одного болтая 78 коп.
3   Цена одного шалтая 66 коп., а цена одного болтая 92 коп.
                    Всего решений – 3
```

Комментарий. Это типичная задача на перебор вариантов (ее несколько видоизмененный текст взят из одного из сборников задач для физико-математических школ при МГУ им. М. В. Ломоносова).

Алгоритм задачи прост: в сложном цикле по x и по y производится проверка всех условий задачи, и если они выполняются, то на экран выводится очередной ответ. Кроме того, в теле сложного цикла предусмотрен счетчик количества найденных решений.

Если немного подумать, то цикл по x (цене шалтая в копейках) можно было бы открыть, начиная с 24, а цикл по y — с 32.

```
SCREEN 9: COLOR 14, 1: LOCATE 2, 17 :
PRINT "Задача про шалтаев и болтаев."
k1 = 125 / 175: k2 = 126 / 175
FOR x = 2 TO 100 STEP 2
FOR y = 2 TO 100 STEP 2
IF 3 * x + y < 100 THEN 1
a = k1 * y: b = k2 * y
IF x < b AND x > a THEN
  n = n + 1
  PRINT n; : LOCATE n + 2, 5
  PRINT " Цена одного шалтая"; x;   "коп.,";
  PRINT " а цена одного болтая"; y; "коп."
END IF
1 NEXT y, x
PRINT TAB(23); "Всего решений - "; n
```

5. Два корня

Ответ: два корня.

$k = .6$, где k — точка раздела корней,

$x_1 = 0.523599$ и $x_2 = 1.570796$

Комментарий. Данная задача решалась бы просто, если бы на заданном интервале находился всего один корень, — тогда для ее решения подошел бы широко известный метод «половинного деления». Следовательно, для решения задачи надо найти точку $k \in [0, \pi]$, лежащую между двумя корнями исходного уравнения $f(x) = g(x)$. Это тоже не проблема: ясно, что корень уравнения $F(x) = 0$, где $F(x) = f(x) - g(x)$, и корень исходного уравнения совпадают. Поэтому, если в цикле по x от 0 до π с достаточно малым шагом окажется, что $F(0)$ и $F(k)$ имеют разные знаки, т. е. $F(0) \cdot F(k) < 0$, то k — точка раздела корней. При этом первый корень лежит на отрезке $[0, k]$, а второй — на отрезке $[k, \pi]$.

Остается описать алгоритм метода «половинного деления». Напомним, что этот метод можно использовать для нахождения *единственного* корня уравнения $f(x) = g(x) \Leftrightarrow F(x) =$

$= f(x) - g(x) = 0$ на отрезке $[A, B]$ с заданной точностью ε, причем абсциссы точек A и B должны быть заданы. Таким образом, входными данными являются: a — левая граница отрезка, содержащего корень уравнения $F(x) = 0$, b — правая граница этого отрезка, ε — степень точности, с которой ищется корень уравнения.

Алгоритм метода «половинного деления»:

1) ищется точка c, лежащая на середине отрезка $[a, b]$: $c = \frac{a+b}{2}$;
2) если $|F(c)| < \varepsilon$, то
   вывести: «Точка c — корень данного уравнения»,
   завершить работу программы,
   иначе
   вычислить знак $F(a) \cdot F(c)$,
   конец ветвления;
3) если $F(a) \cdot F(c) \leqslant 0$, то
   корень уравнения лежит в интервале $[a, c]$, —
   принять значение b равным значению c и перейти к п. 1,
   иначе
   корень уравнения лежит в интервале $[c, b]$, —
   принять значение a равным значению c и перейти к п. 1,
   конец ветвления.

```
SCREEN 9: COLOR 14, 1: PRINT : PRINT TAB(10); " Два корня"
pi# = 3.141592654#
DEF fnf (x) = pi# * pi# * (2 * SIN(x) - 5) - 54 * x * x + 33 * pi# * x
m = fnf(0): e = .00001
FOR x = 0 TO pi# STEP .1
  n = fnf(x)
  IF n * m < 0 THEN 1
NEXT
1 k = x: PRINT " k ="; k; ", где k – точка раздела корней"
a# = 0: b# = k: GOSUB 10
PRINT " x1 = "; USING "#.######"; c#; : PRINT " и";
a# = k: b# = pi#: GOSUB 10
PRINT " x2 = "; USING "#.######"; c#
END
10 c# = (b# + a#) / 2
  IF ABS(fnf(c#)) < e THEN 30
  IF fnf(a#) * fnf(c#) <= 0 THEN 20
  a# = c#: GOTO 10
20 b# = c#: GOTO 10
30 RETURN
```

6. Тригонометрическая система

Ответ: $x \approx 4{,}620509$ rad $\approx 264°44'8''$; $y \approx 3.573309$ rad $\approx$ $\approx 204°44'8''$.

Комментарий. В процессе отладки необходимо выявить искомые значения x и y и сузить промежутки их поиска: $x \in [4{,}62014; 4{,}62051]$, $y \in [3{,}5732; 3{,}5734]$. Далее необходимо перевести эти значения в градусную меру.

```
CLS : PRINT "Решение системы двух тригонометрических уравнений"
PRINT
pi = 3.1415926#: e = .000001: r = -SQR(2): COLOR 14
FOR x = 4.62051 TO 4.62049 STEP -e
  FOR y = 3.5734 TO 3.5732 STEP -e
    p = SIN(x) + SIN(y): q = COS(x) + COS(y)
    IF ABS(p - r) < e AND ABS(q + 1) < e THEN
      PRINT "x ="; x; "rad", "y ="; y; "rad": a = x: b = y: GOTO 1
    END IF
NEXT y, x
1 COLOR 10: PRINT : u = a: GOSUB 2: grad = u2: min = u4: sec = u6
PRINT "Величина угла x ="; u2; "°"; u4; "'"; u6; CHR$(34)
u = b: GOSUB 2: grad = u2: min = u4: sec = u6
PRINT "Величина угла y ="; u2; "°"; u4; "'"; u6; CHR$(34)
END
2 REM ---- перевод радианной меры угла в градусную ----
u1 = 180 * u / pi: u2 = INT(u1)
u3 = (u1 - u2) * 60: u4 = INT(u3)
u5 = (u3 - u4) * 60: u6 = CINT(u5)
IF u6 = 60 THEN
  u4 = u4 + 1: u6 = 0
END IF
IF u4 = 60 THEN
  u4 = 0: u2 = u2 + 1
END IF: RETURN
```

7. «Хитрое» квадратное уравнение

Ответ: $x = -2$ или $x = 0{,}00000000000001$.

Комментарий. Если выбрать тип данных с одинарной точностью, то ответы окажутся неправильными. Поэтому при вычислениях необходимо использовать числовой тип с двойной точностью.

```
SCREEN 9: COLOR 14, 1: PRINT "Хитрое уравнение": DEFDBL A-Z
a = 1: b = 1.99999999999991#: c = -.00000000000002#
d = b * b - 4 * a * c: f = SQR(d)
```

```
x1 = -(f + b) / (2 * a): x = INT(x1)
x2 = (f - b) / (2 * a)
PRINT "Ответ: x = "; x; " или x = ";
PRINT USING "#.###############"; x2
```

8. «Сила» и «Мощь»

Ответ:

«Количество тренажеров»		Стоимость
«Сила»	«Мощь»	
3	19	252 000
8	15	244 000
13	11	236 000
18	7	228 000
23	3	220 000

Комментарий. Это типичная задача на перебор чисел.

```
SCREEN 9: COLOR 14, 1: REM *** Сила и Мощь ***
max1 = INT(102 / 4): max2 = INT(103 / 5): PRINT : PRINT
PRINT " | Тренажёры | СИЛА | МОЩЬ | Стоимость |"
PRINT " ---------------------------------------"
FOR x = 1 TO max1
  FOR y = 1 TO max2
    IF (4 * x + 5 * y = 107) THEN
      I = I + 1: x(i) = x: y(i) = y
      s(i) = 8000 * x + 12000 * y
      PRINT " количество ";
      PRINT USING "##"; x(i);
      PRINT "        ";
      PRINT USING "##"; y(i);
      PRINT "       "; s(i)
    END IF
  NEXT
NEXT
LOCATE 11, 6: PRINT "Всего решений -"; i
```

9. Точки с аппликатой 1

Ответ:

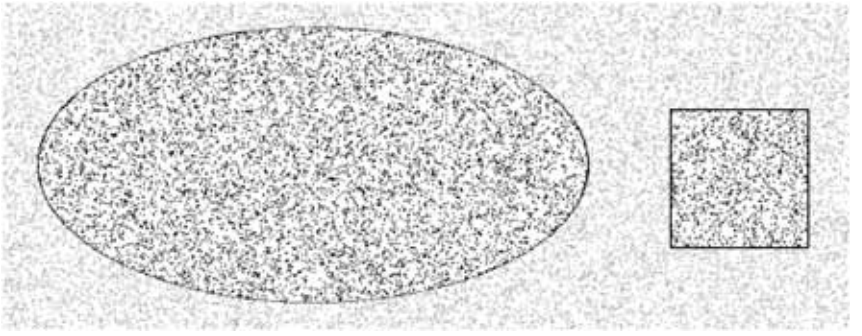
```
Точки с аппликатой 1
Ответ: всего 4 точки:
(6; 8; 1)
(11; 15; 1)
(16; 22; 1)
(21; 29; 1)
```

Комментарий. Это также типичная задача на перебор чисел.

```
SCREEN 9: COLOR 14, 1: PRINT : PRINT " Точки с аппликатой 1"
PRINT " Ответ: всего 4 точки:"
FOR x = 2 TO 30
  FOR y = 2 TO 30
    a = 7 * x - 5 * y
    IF a = 2 THEN PRINT " ("; x; ";"; y; ";"; " 1 )"
  NEXT
NEXT
```

10. Метод Монте-Карло

Результат работы программы



```
Всего было брошено 20164 песчинок, из них
в эллипс попало 8602 штук, а в квадрат - 1398 штук
Теоретическая площадь эллипса - 6.283185, теоретическая площадь
квадрата - 1
Статистическая площадь эллипса - 6.153076
```

Комментарий. Из условия задачи понятно, что надо сделать, чтобы реализовать этот процесс. Единственная сложность — как произвести подсчет точек, попавших в эллипс и попавших в квадрат.

```
SCREEN 12: PRINT TAB(20); "Метод Монте-Карло"
RANDOMIZE TIMER: a = 240: b = 140: p = 2 * 3.1415926#
FOR t = 0 TO 6.28 STEP .01
  x = 200 * COS(t) + a: y = 100 * SIN(t) + b: PSET (x, y), 14
NEXT
LINE (500, 100)-(600, 200), 14, B
1 IF I + n = 10000 THEN 2
x = INT(RND * 620 + 10)
y = INT(RND * 240 + 20)
```

```
k = k + 1: PSET (x, y), 2
IF x <= 600 AND x >= 500 AND y <= 200 AND y >= 100 THEN
  I = I + 1: PSET (x, y), 14
END IF
c = ((x - a) ^ 2) / 40000 + ((y - b) ^ 2) / 10000
IF c <= 1 THEN
  n = n + 1: PSET (x, y), 14
END IF
GOTO 1
2 COLOR 15: LOCATE 18, 1
PRINT " Всего было брошено"; k; " песчинок, из них"
PRINT " в эллипс попало"; n; "штук, а в квадрат"; i; "штук"
PRINT " Теоретическая площадь эллипса -"; p; ",
       теоретическая площадь квадрата - 1"
PRINT " Статистическая площадь эллипса -"; n / i
```

11. Ряд из 100 натуральных чисел

Контрольные примеры:

1) на 23-м месте находится цифра 6;
2) на 147-м месте находится цифра 8.

Алгоритм решения задачи:

1) очистка экрана;
2) формирование трех символьных переменных, содержащих числа от 1 до 100, так, чтобы все эти переменные умещались на экране монитора, а сами числа были записаны в них без пробелов;
3) подсчет длины каждой из этих символьных переменных;
4) вывод этих символьных переменных на экран через одну пустую строку (по строкам между этими переменными будет «бегать» тильда — символ «~»);
5) ввод числа k — порядкового номера интересующей нас цифры; если k больше, чем суммарная длина сформированных трех символьных переменных, то возврат на начало п. 5;
6) определение строковой переменной, в которой находится этот номер;
7) формирование числа n — номера позиции искомой цифры в соответствующей символьной переменной;
8) формирование числа p — номера строки, по которой должна перемещаться тильда от первой до нужной позиции;
9) вызов подпрограммы перемещения тильды;

10) вызов подпрограммы выделения k-го символа из символьной переменной, вывод результата работы программы;
11) завершение работы программы.

```
CLS : COLOR 4
a$="12345678910111213141516171819202122232425262728293031323334353637383940414243444546474849505152535455565758596061626364656667686970717273747576777879808182838485868788899091929394959697989900"
b$="45464748495051525354555657585960616263646566676869707172737475767778798081828384"
c$="858687888990919293949596979899100"
a=LEN(a$): b=LEN(b$): c=LEN(c$)
PRINT a$: PRINT : PRINT b$: PRINT : PRINT c$
COLOR 10: LOCATE 12, 1: INPUT "Чему равно k"; k
IF k < a + 1 THEN
  n = k: p = 2: GOSUB 1
  z$ = a$: GOSUB 2
END IF
IF k > a AND k < a + b - 1 THEN
  n = k - 79: p = 4: GOSUB 1
  z$ = b$: GOSUB 2
END IF
IF k > a + b AND k < a + b + c + 1 THEN
  n = k - 159: p = 6: GOSUB 1
  z$ = c$: GOSUB 2
END IF
IF k > a + b + c THEN
  LOCATE 22, 1
PRINT "В нашей записи всего"; a + b + c; " цифры. Перезапуститесь!"
END IF
END
1 FOR i = 1 TO n
    LOCATE p, i: PRINT "~"
    FOR j = 1 TO 5000: NEXT
    LOCATE p, i: PRINT " "
  NEXT
LOCATE p, i - 1: PRINT "~": RETURN
2 t = n: k$ = MID$(z$, t, 1)
LOCATE 12, 40: PRINT "Это цифра - "; k$: RETURN
```

Результат работы программы:

```
12345678910111213141516171819202122232425262728293031323334353637
38394041424344454647484950515253545556575859606162636465666768697
07172737475767778798081828384858687888990919293949596979899100
Чему равно k? 144           Это цифра - 7
```

12. Поиск элемента по заданному признаку

Комментарий. По сути, эта задача представляет собой реализацию функции поиска заданного фрагмента в тексте документа, которая в текстовых редакторах обычно вызывается при помощи пункта меню 

. Контроль результата работы программы — визуальный.

Алгоритм решения задачи:

1) очистка экрана, вывод названия программы;
2) резервирование памяти для элементов массива;
3) формирование массива;
4) цикл по *i* от 0 до 11 (так как в данной задаче 12 элементов — названий месяцев);
   считывание *i*-го элемента массива,
   вывод *i*-го элемента на экран,
   вычисление его длины *a*(*i*),
   конец цикла (следующее значение *i*);
5) ввод с клавиатуры «слова-признака» для поиска;
6) вычисление длины этого слова (*b*);
7) если длина «слова-признака» больше длины любого элемента массива, то переход к п. 11;
8) цикл по *i* от 0 до 11;
9) цикл по *j* от 1 до (*a*(*i*) – *b* + 1) — именно столько раз «слово-признак» может поместиться в проверяемом (*i*-м) слове, начиная с его 1-й буквы;
   выделение из проверяемого слова буквосочетания, содержащего *b* букв,
   если это буквосочетание совпадает со «словом-признаком», то
      вывод на экран проверяемого слова,
      увеличение счетчика совпадений на 1,
      выбор следующего значения *i*,
   иначе
      выбор следующего значения *j*,
      конец ветвления;
10) если счетчик пуст, то переход к п. 11, иначе — завершение работы программы;
11) вывод на экран сообщения: «Слов с таким признаком нет!».

```
CLS : COLOR 9: PRINT TAB(29); "Поиск"
DIM a$(11), a(11): PRINT : COLOR 14
```

```
DATA январь, февраль, март, апрель, май, июнь, июль
DATA август, сентябрь, октябрь, ноябрь, декабрь
FOR i = 0 TO 11
  READ a$(i): PRINT a$(i), : a(i) = LEN(a$(i))
NEXT
PRINT : PRINT : COLOR 13
PRINT "По какому признаку (буквосочетанию) будем искать слова
в этом массиве"
LOCATE 7, 70: INPUT b$: b = LEN(b$)
IF b > 8 THEN 1
FOR i = 0 TO 11:
  FOR j = 1 TO a(i) - b + 1
    c$ = MID$(a$(i), j, b)
    IF c$ = b$ THEN
      PRINT a$(i), : n = n + 1: GOTO 3
    ELSE
      GOTO 2
    END IF
2 NEXT j
3 NEXT i
IF n = 0 THEN 1 ELSE END
1 PRINT "Слов с таким признаком в данном массиве нет."
```

13. Буквенное уравнение

Ответ: $\sqrt{11122225} = 3335$; $\sqrt{44422225} = 6665$.

Комментарий. Хотя эта задача на перебор чисел, ее проще решить с помощью символьной переменной.

Понятно, что в конце числа, стоящего под знаком радикала, может быть либо цифра 1, либо 5, либо 6 (других вариантов нет). Образовав символьные значения «111», «222», ..., «999» и «1111», «2222», ..., «9999» и «склеивая» их между собой, а потом попеременно — сначала с «1», затем с «5» и, наконец, с «6» и превращая полученные записи в числа двойной точности, остается проверять выполнение заданного условия и сформулировать ответ.

```
CLS : DIM x$(72), y$(72), z$(72), a#(216)
DATA 111, 222, 333, 444, 555, 666, 777, 888, 999
DATA 1111, 2222, 3333, 4444, 5555, 6666, 7777, 8888, 9999
FOR i = 0 TO 8
  READ a$(i)
NEXT
FOR i = 0 TO 8
  READ b$(i)
```

```
NEXT
FOR i = 0 TO 8
  FOR j = 0 TO 8
    IF i = j THEN 1
    n = n + 1: x$(n) = a$(i) + b$(j) + "1"
    y$(n) = a$(i) + b$(j) + "5": z$(n) = a$(i) + b$(j) + "6"
1 NEXT j
NEXT i
FOR i = 1 TO n
  a#(i) = VAL(x$(i)): a#(n + i) = VAL(y$(i)):
  a#(2 * n + i) = VAL(z$(i))
NEXT: PRINT "Ответ:"
FOR k = 1 TO 216
  b# = SQR(a#(k)): a# = INT(b#): c# = b# - a#
  IF c# = 0 THEN
    PRINT TAB(6); b#; "^ 2 ="; a#(k)
  END IF
NEXT
```

14. «Черные пятницы»

Контрольные примеры:

1) 2007 г., 1 января — понедельник, «черные пятницы» выпали на апрель и июль;

2) 2008 г., 1 января — вторник, «черная пятница» выпала на июнь;

3) 1995 г., 1 января — воскресенье, «черные пятницы» выпали на январь и октябрь.

Комментарий. Данная задача не очень сложная, так как входные данные (день недели, выпавший на 1 января искомого года, и информация о том, високосный это год или простой) позволяют вычислить сначала все даты пятниц в этом году, а потом выбрать из них выпадающие на 13-е число месяца.

```
SCREEN 9: COLOR 14, 1: LOCATE 1, 24
PRINT "Программа выявления черных пятниц"
DIM a$(12), a(12), b(12), c(366, 13), s(12), d(366)
DATA январь,февраль,март,апрель,май,июнь,июль,август
DATA сентябрь,октябрь,ноябрь,декабрь
FOR i = 1 TO 12: READ a$(i): NEXT
DATA 31,28,31,30,31,30,31,31,30,31,30,31
FOR i = 1 TO 12: READ a(i): NEXT
DATA 31,29,31,30,31,30,31,31,30,31,30,31
FOR i = 1 TO 12: READ b(i): NEXT
INPUT "Какой год вас интересует: простой (p) или високосный (v)"; god$
```

```
IF god$ = "p" OR god$ = "P" THEN
  b = 365: FOR i = 1 TO 12: s = s + a(i): s(i) = s: NEXT
ELSE
  b = 366: FOR i = 1 TO 12: s = s + b(i): s(i) = s: NEXT
END IF
PRINT : PRINT "Какой день недели выпал на 1 января этого года:";
PRINT " понедельник – 5, вторник – 4,"
INPUT "среда – 3, четверг – 4, пятница – 1, суббота – 7,
       воскресенье – 6"; a
PRINT
FOR i = a TO b STEP 7
  d(i) = i   'номера пятниц в текущем году
NEXT
PRINT : s(0) = 0: k = 1: GOSUB 1: COLOR 15
PRINT "Черные пятницы: ";
FOR i = a TO b STEP 7
  FOR k = 1 TO 12
    IF c(i, k) = 13 THEN
      PRINT a$(k); " –"; c(i, k); " ";
    END IF
  NEXT
NEXT
END
1 FOR i = a TO b STEP 7
  IF s(k) >= d(i) AND d(i) > s(k – 1) THEN 2 ELSE k = k + 1
2 c(i, k) = d(i) – s(k – 1)
  IF c(i, k + 1) > 7 THEN c(i, k + 1) = c(i, k + 1) – 7
NEXT: RETURN
```

Результат работы программы:

```
Программа выявления черных пятниц
Какой год вас интересует: простой (p) или високосный (v)? p

Какой день недели выпал на 1 января этого года: понедельник – 5,
вторник – 4, среда – 3, четверг – 4, пятница – 1, суббота – 7,
воскресенье – 6? 5

Черные пятницы: апрель – 13, июль – 13
```

15. Числа Фибоначчи

Комментарий. Данную задачу можно решать разными способами:

1) более трудный способ — вручную выписывать все числа Фибоначчи в одну строку, одновременно подсчитывая количество знаков в этой строке, и остановиться, когда их количество будет равно 234;

2) более легкий способ — поручить формирование этой строки компьютеру, затем разбить эту строку на три символьные переменные и напечатать их через одну строку, чтобы между ними «бегал» символ тильды (см. задачу 11).

```
SCREEN 9: COLOR 14, 1: DEFDBL A-Z
LOCATE 1, 13: COLOR 15
PRINT "Числа Фибоначчи, записанные в столбики:"
DIM f(48), f$(48), a(48): PRINT : COLOR 14
f(1) = 1: f(2) = 1: f$(1) = "1": f$(2) = " 1"
REM *** формирование чисел Фибоначчи ***
FOR n = 3 TO 48
  f(n) = f(n - 1) + f(n - 2): PRINT f(n - 2),
  f$(n) = STR$(f(n)): a(n) = LEN(f$(n))
  s$ = s$ + f$(n - 2)
NEXT: b = LEN(s$)
FOR i = 1 TO b
  b$ = MID$(s$, i, 1)
  IF b$ = " " THEN b$ = " "
  c$ = c$ + b$
NEXT
REM *** формирование 3-х строк с числами Фибоначчи ***
COLOR 15: a$ = "~"
PRINT " Числа Фибоначчи, записанные в строки:": COLOR 14
d$(1) = MID$(c$, 1, 80): LOCATE 14, 1: PRINT d$(1)
d$(2) = MID$(c$, 81, 80): LOCATE 16, 1: PRINT d$(2)
d$(3) = MID$(c$, 161, 234): LOCATE 18, 1: PRINT d$(3)
LOCATE 20, 1: COLOR 15
INPUT "На каком месте находится испытуемая цифра k (0<k<235)"; k
IF k < 81 THEN
  p = 15: t = 1: m = k
END IF
IF k < 161 AND k > 80 THEN
  p = 17: t = 2: m = k - 80
END IF
IF k > 160 THEN
  p = 19: t = 3: m = k - 160
END IF
r$ = MID$(d$(t), m, 1)
FOR j = 1 TO m
  LOCATE p, m: COLOR 15: PRINT a$
NEXT
LOCATE 22, 1: COLOR 14: PRINT "На"; k; "-м месте находится цифра "; r$
```

Результат работы программы:

```
Числа Фибоначчи, записанные в столбики:

1           1           2           3           5
8           13          21          34          55
89          144         233         377         610
987         1597        2584        4181        6765
10946       17711       28657       46368       75025
121393      196418      317811      514229      832040
1346269     2178309     3524578     5702887     9227465
14930352    24157817    39088169    63245986    102334155
165580141   267914296   433494437   701408733   1134903170
1836311903

Числа Фибоначчи, записанные в строки:

1123581321345589144233377610987159725844181676510946177112865746368750251213931964183178115142298320401346269217830935245785702887922746514930352241578173908816963245986102334155165580141267914296433494437701408733113490317018363119031

На каком месте находится испытуемая цифра k (0<k<235)? 170
На 170-м месте находится цифра 1
```

16. Иллюзия вращения

Комментарий. Иллюзия вращения трилистника (см. [11]) осуществляется за счет изменения абсциссы кривой, заданной параметрически: к параметру f прибавляется некоторая постоянно увеличивающаяся переменная k, формирование которой происходит до цикла по $f \in [0, 2\pi]$. Тело цикла при этом составляет собственно рисование (точнее, перерисовывание с новыми значениями параметров) трилистника. После отработки цикла производится очистка экрана и изменение значения переменной k, после чего весь процесс повторяется.

```
SCREEN 9 : COLOR 14, 9
1 k = k + .05
FOR f=0 TO 6.28 STEP .01
  p = 120 * COS(3 * f)
  x = p*COS(f + k) + 320
  y = p*SIN(f) + 170
  PSET (x, y), 14
NEXT f: CLS
IF INKEY$ = " " THEN END ELSE 1
```

17. Лифт

Решение:

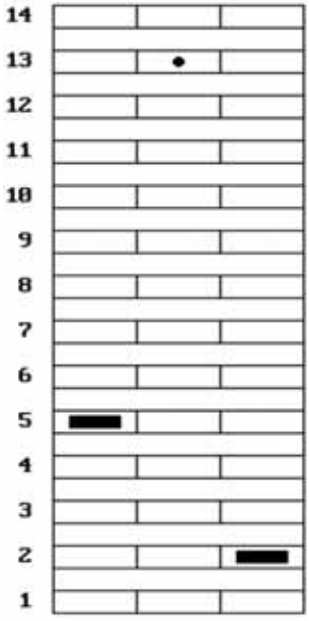
```
SCREEN 12: LOCATE 2, 53: COLOR 2: PRINT "LIFT"
DIM y(100): x1 = 125: x0 = 175: x2 = 225: RANDOMIZE TIMER
FOR i = 14 TO 1 STEP -1
  n = n + 2: LOCATE n, 10: PRINT USING "##"; i: y(i) = 472 - 32 * i
NEXT: GOSUB 1: LINE (100, 32)-(250, 432), 5, B
n0 = CINT(RND * 13 + 1): y0 = y(n0) ' ****************************
n1 = CINT(RND * 13 + 1): y1 = y(n1)   ' *  ординаты человека и лифтов  *
n2 = CINT(RND * 13 + 1): y2 = y(n2) ' ****************************
d1 = ABS(n0 - n1): d2 = ABS(n2 - n0)
LOCATE 7, 40: PRINT "Человек находится на "; n0; "-м этаже."
CIRCLE (x0, y0), 3, 15: PAINT (x0, y0), 14, 15
LOCATE 8, 40: PRINT "Он нажал кнопку вызова лифта."
LOCATE 9, 40: PRINT "Лифты находятся на этажах:"; n1; "и"; n2
a = 110: b = y1: c = 9: GOSUB 2
a = 210: b = y2: c = 9: GOSUB 2: SLEEP 2
IF d1 <= d2 THEN 10 ELSE 20
10 REM --- движение 1-го лифта до места вызова ---
p = 125: a = 110: q = 115: b = y1: f1 = y1: f2 = y0
IF n0 = n1 THEN
  GOSUB 3: GOTO 40
END IF
IF n0 < n1 THEN h = 2 ELSE h = -2
GOSUB 4: GOTO 30
20 REM --- движение 2-го лифта до места вызова ---
p = 225: a = 210: q = 215: b = y2: f1 = y2: f2 = y0
IF n0 = n2 THEN
  GOSUB 3: GOTO 40
END IF
IF n0 < n2 THEN h = 2 ELSE h = -2
GOSUB 4: GOTO 30
30 c = 9: b = y0: GOSUB 2: GOSUB 3
40 LOCATE 10, 40: INPUT "На какой этаж изволите поехать"; k
IF k = n0 OR k > 14 OR k < 1 THEN
  COLOR 5: LOCATE 12, 40: PRINT "Вы, кажется, перегрелись!"
  LOCATE 13, 40: INPUT "Так куда изволите поехать"; k
END IF
f1 = y0: f2 = y(k)
IF f1 < f2 THEN h = 2 ELSE h = -2
GOSUB 4: b = f2: GOSUB 2: GOSUB 3
COLOR 14: LOCATE 15, 48: PRINT "Всё, уже приехали!": END
1 REM *** схематичный рисунок подъезда ***
FOR z = 32 TO 472 STEP 32
  LINE (100, z)-(250, z - 16), 5, B: LINE (150, z)-(200, z - 16), 5, B
```

```
NEXT: RETURN
2 REM *** рисунок лифта ***
LINE (a, b - 4)-(a + 30, b + 4), c, BF: RETURN
3 REM *** двери лифта открываются и закрываются ***
SLEEP 2
FOR x = p TO q STEP -1
  LINE (x, b + 3)-(2 * p - x, b - 3), 14, BF
  FOR j = 1 TO 5000: NEXT
NEXT: SLEEP 2
FOR x = q TO p
  LINE (q - 5, b + 4)-(x, b - 4), 9, BF
  LINE (p + 15, b + 4)-(2 * p - x, b - 4), 9, BF
  FOR j = 1 TO 5000: NEXT
NEXT: SLEEP 2: RETURN
4 REM *** движение лифта ***
FOR y = f1 TO f2 STEP h
  GOSUB 1: LINE (a, y + 4)-(a + 30, y - 4), 9, BF
  FOR i = 1 TO 5000: NEXT
  LINE (a, y + 4)-(a + 30, y - 4), 0, BF
NEXT: RETURN
```

Результат работы программы:

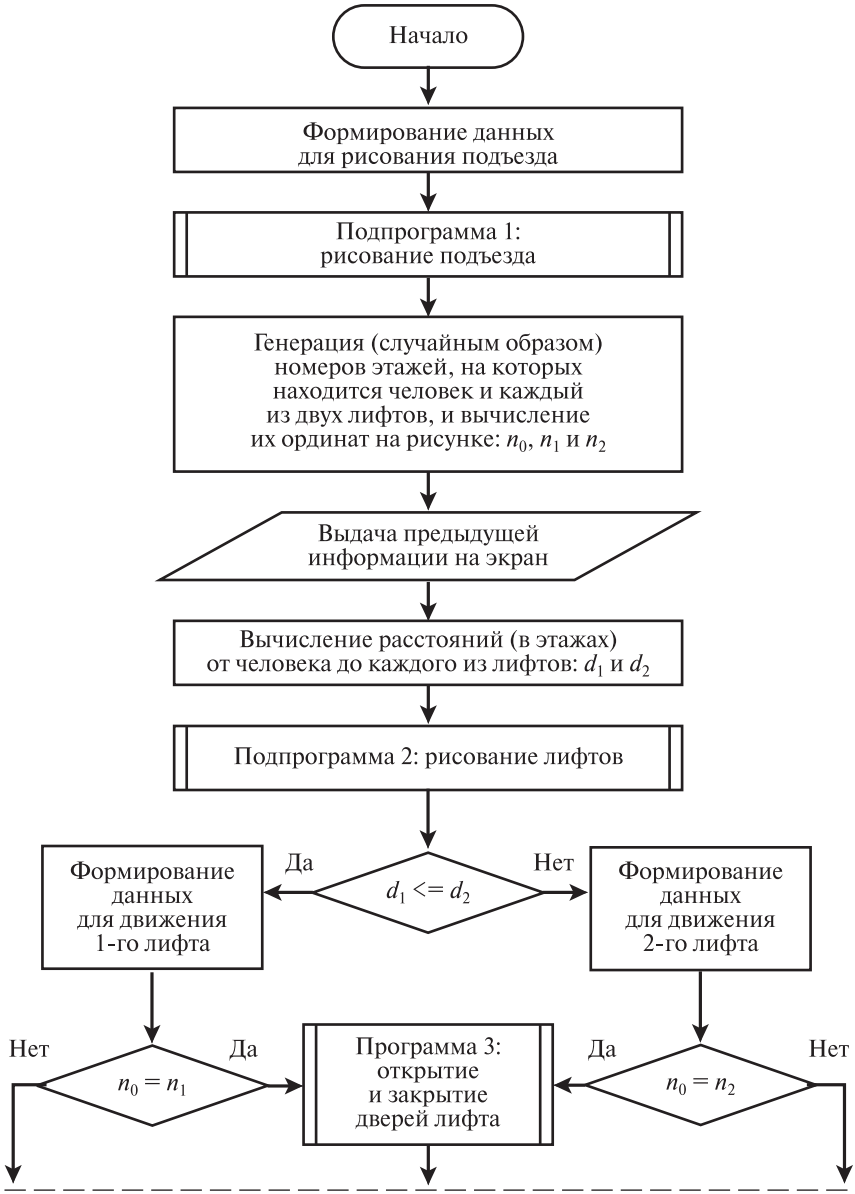
```
                LIFT
Человек находится на 13-м этаже.
Он нажал кнопку вызова лифта.
Лифты находятся на этажах: 10 и 2
На какой этаж изволите поехать? 5


          Всё, уже приехали!
```




```
Чтобы продолжить, нажмите любую клавишу
```

Блок-схема решения задачи показана на рис. 9.



(*См. продолжение схемы на следующей странице*)

(*См. начало схемы на следующей странице*)

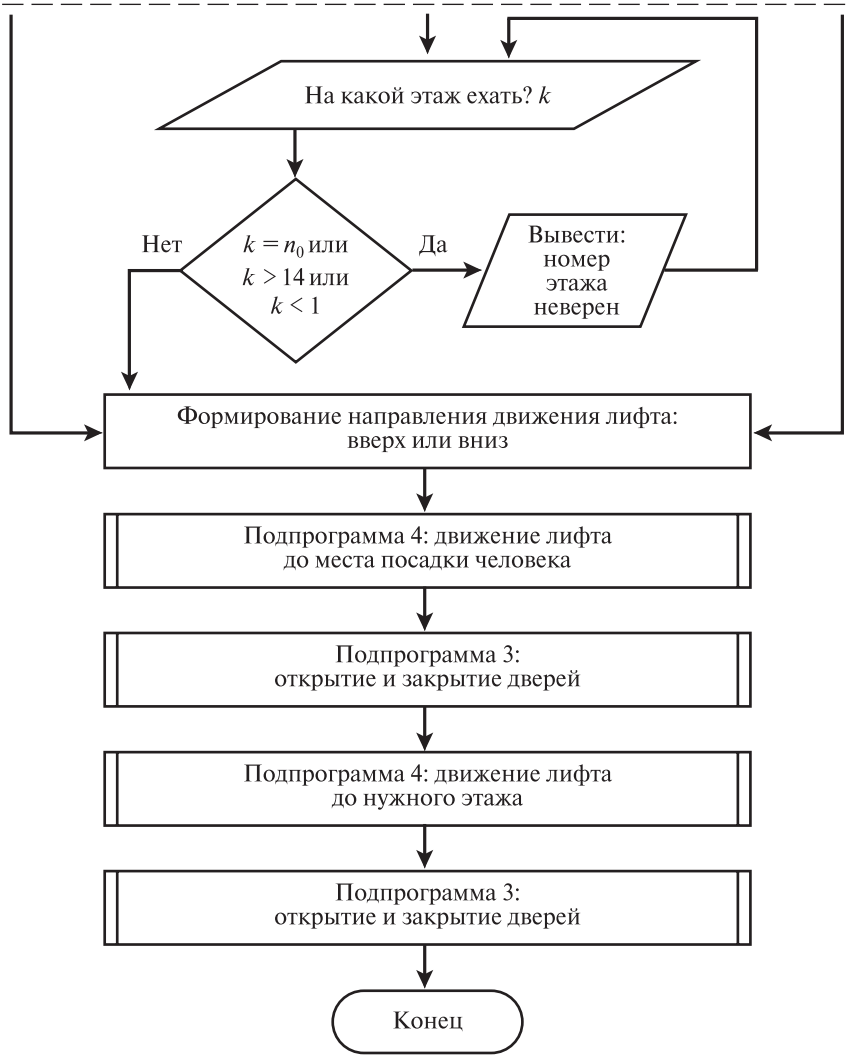


Рис. 9

18. Реостат

Комментарий. Вся трудность этой задачи состоит в том, чтобы реализовать движение стрелки вольтметра. Для этого, во-первых, необходимо знать уравнение эллипса в параметрическом виде:

$$\begin{cases} x = a \cdot \cos \varphi, \\ y = b \cdot \sin \varphi, \end{cases}$$

где φ — угол, на который может отклоняться стрелка вольтметра от левого крайнего до правого крайнего положения, а a и b — большая и малая полуоси эллипса (определение границ угла φ и подбор значений a и b производится подбором в процессе отладки программы). Во-вторых, необходимо обеспечить синхронное движение стрелки вольтметра, движение бегунка реостата и одновременную выдачу показаний вольтметра в числовом виде. В остальном решение задачи достаточно простое.

```
SCREEN 9: COLOR 15, 0: LOCATE 1, 24: PRINT "Реостат": COLOR 14, 0
a$ = "Движение бегунком осуществляется клавишами блока Num Lock:"
LOCATE 3, 12: PRINT a$: LOCATE 4, 29: PRINT "4 – влево и 6 – вправо."
REM *** рисунок вольтметра ***
  LINE (370, 220)-(560, 300), 7, B: LINE (410, 220)-(410, 300), 7
  CIRCLE (390, 252), 3, 14: CIRCLE (390, 268), 3, 14: GOSUB 4
  LINE (447, 265)-(523, 280), 7, B
  LOCATE 20, 58: COLOR 10, 0: PRINT "V = 0.0"
REM *** рисунок реостата с проводами ***
  CIRCLE (30, 248), 3, 14: CIRCLE (30, 268), 3, 14: LOCATE 19, 3
  PRINT "4 v"
  LINE (30, 268)-(390, 268), 14: CIRCLE (90, 248), 3, 14
  LINE (30, 248)-(90, 248), 14: LINE (80, 256)-(100, 218), 7, B
  LINE (80, 208)-(100, 218), 7, B: LINE (280, 256)-(300, 218), 7, B
  LINE (280, 208)-(300, 218), 7, B: CIRCLE (290, 213), 3, 14
  LINE (290, 213)-(320, 213), 14: LINE (320, 213)-(320, 252), 14
  LINE (320, 252)-(390, 252), 14: LINE (101, 212)-(279, 214), 14, BF
  LINE (278, 226)-(268, 210), 9, BF: LINE (444, 239)-(451, 255), 1
FOR x = 101 TO 279 STEP 2
  LINE (x, 224)-(x, 250), 14   'катушка
NEXT
xt = 273: GOSUB 5
1 REM *** управление бегунком ***
DO
  a$ = INKEY$
  IF a$ = "6" THEN 2
  IF a$ = "4" THEN 3
  IF a$ = " " THEN END
LOOP
```

```
2 REM *** движение бегунка вправо ***
  xt = xt + 2: GOSUB 5: LINE (xt - 7, 226)-(xt - 5, 210), 0, BF
  LINE (xt - 7, 212)-(xt - 5, 214), 14, BF
  LINE (xt - 8, 224)-(xt - 8, 250), 14
  LINE (xt - 6, 224)-(xt - 6, 250), 14: GOTO 1
3 REM *** движение бегунка влево ***
  xt = xt - 2: GOSUB 5: LINE (xt + 7, 226)-(xt + 5, 210), 0, BF
  LINE (xt + 7, 212)-(xt + 5, 214), 14, BF
  LINE (xt + 8, 224)-(xt + 8, 250), 14
  LINE (xt + 6, 224)-(xt + 6, 250), 14: GOTO 1
END
4 REM *** шкала вольтметра ***
  CIRCLE (485, 255), 60, 7, .57, 2.57, .4: LINE (536, 242)-(528, 259), 7
  CIRCLE (485, 275), 60, 7, .77, 2.37, .4: LINE (434, 242)-(442, 259), 7
  PAINT (485, 240), 7, 7
  FOR f = .8 TO 2.35 STEP .38
    x1 = 55 * COS(f) + 485: y1 = 262 - 21 * SIN(f)
    CIRCLE (x1, y1), 2, 0, , , 1.2: PAINT (x1, y1), 0, 0
  NEXT: RETURN
5 REM *** движение стрелки вольтметра ***
  t = .009337 * xt - .199096
  IF xt < 107 THEN
    xt = 107: LINE (102, 226)-(112, 210), 9, BF: GOTO 6
  END IF
  IF xt > 272 THEN
    xt = 273: LINE (278, 226)-(268, 210), 9, BF: GOTO 6
  END IF
  IF xt >= 107 AND xt < 273 THEN
    LINE (xt - 5, 226)-(xt + 5, 210), 9, BF
    LINE (xv1, yv1)-(xn1, yn1), 7
    GOSUB 4: v = 6.578313 - .0240964 * xt
    LOCATE 20, 62: COLOR 10, 0: PRINT USING "#.#"; v
  END IF
  xv = 60 * COS(t) + 485: xv1 = xv: xn = 50 * COS(t) + 485: xn1 = xn
  yv = 255 - 22 * SIN(t): yv1 = yv: yn = 269 - 20 * SIN(t): yn1 = yn
  LINE (xv, yv)-(xn, yn), 1
6 RETURN
```

19. Опыт Бюффона

Решение:

```
SCREEN 9: COLOR 14, 1: LOCATE 1, 35: PRINT "Опыт Бюффона"
RANDOMIZE TIMER: PRINT
FOR k = 1 TO 10
  FOR i = 1 TO 4040
    x = RND
    IF x < .5 THEN n = n + 1
  NEXT
  p(k) = n * 100 / 4040
  PRINT "Опыт №"; k; ". Вероятность выпадения герба – "; p(k); "%": n = 0
NEXT
```

```
FOR i = 1 TO 10
  p = p + p(i)
NEXT
p = p / 10: z = ABS(50 - p): PRINT: PRINT
PRINT "Среднее значение вероятности выпадения герба - "; p; "%"
PRINT "Отклонение от классической вероятности составляет ";
PRINT USING "#.#####"; z; : PRINT " %"
```

Результат работы программы:

```
Опыт Бюффона
Опыт № 1.  Вероятность выпадения герба - 49.77723%
Опыт № 2.  Вероятность выпадения герба - 49.9505%
Опыт № 3.  Вероятность выпадения герба - 50.12376%
Опыт № 4.  Вероятность выпадения герба - 50.14851%
Опыт № 5.  Вероятность выпадения герба - 49.92574%
Опыт № 6.  Вероятность выпадения герба - 50.81683%
Опыт № 7.  Вероятность выпадения герба - 50.37129%
Опыт № 8.  Вероятность выпадения герба - 51.11386%
Опыт № 9.  Вероятность выпадения герба - 50.79208%
Опыт № 10. Вероятность выпадения герба - 49.08416%

Среднее значение вероятности выпадения герба - 50.2104%
Отклонение от классической вероятности составляет 0.21040%
```

20. Картинги

Комментарий. Эту задачу сначала лучше решить «на бумаге».

Пусть до встречи на линии старта 1-й картинг сделал n кругов, а 2-й — m кругов. Так как они стартовали одновременно, то

$$t_1 = t_2 \Leftrightarrow \frac{2\pi R_1 n}{V_1} = \frac{2\pi R_2 m}{V_2} \Leftrightarrow \frac{R_1 n}{V_1} = \frac{R_2 m}{V_2}. \qquad (1)$$

Так как $V_1 = 54$ км/ч $= 15$ м/с, а $V_2 = 72$ км/ч $= 20$ м/с; $R_1 = 105$ м и $R_2 = 120$ м, то, подставляя эти значения в формулу (1), мы получим:

$$\frac{105n}{15} = \frac{120m}{20} \Leftrightarrow 7n = 6m,$$

а значит, первое натуральное решение получается при $n = 6$ и $m = 7$.

Ясно, что время, пройденное с момента старта до первой встречи картингов на линии старта, равно

$$t = t_1 = t_2 = \frac{2\pi \cdot 105 \cdot 6}{15} = 84\pi \text{ (с)} \approx 264 \text{ с}.$$

Следовательно, угол ежесекундного поворота для 1-го картинга: $\varphi_1 = t/6$, а для 2-го — $\varphi_2 = -t/7$ (второй угол взят отрица-

тельным, чтобы картинги двигались в противоположных направлениях). Далее в цикле по времени от 0 до 264 с надо заставить перемещаться по экрану два кружочка (картинга) и обеспечить вывод результата решения задачи.

```
SCREEN 12: COLOR 2: PRINT TAB(36); "Картинги": COLOR 15
LOCATE 15, 62: PRINT "Линия старта": LOCATE 15, 40: PRINT "O"
COLOR 15: LOCATE 2, 2: PRINT "v1 = 54 км/ч = 15 м/с": v1 = 15
COLOR 14: LOCATE 2, 57: PRINT "v2 = 72 км/ч = 20 м/с": v2 = 20
COLOR 15: LOCATE 3, 2: PRINT "r1 = 105 м": r1 = 105
COLOR 14: LOCATE 3, 57: PRINT "r2 = 120 м": r2 = 120
r1 = 105: r2 = 120: LINE (320, 220)-(590, 220), 13
FOR t = 0 TO 264
  LOCATE 20, 2: PRINT "t ="; t; "c"
  LINE (320, 220)-(580, 220), 13
  f1 = t / 6: f2 = -t / 7
  x1 = r1 * COS(f1) + 320: y1 = r1 * SIN(f1) + 220
  x2 = r2 * COS(f2) + 320: y2 = r2 * SIN(f2) + 220
  CIRCLE (x1, y1), 2, 15: PAINT (x1, y1), 15, 15
  CIRCLE (x2, y2), 2, 14: PAINT (x2, y2), 14, 14
  FOR i = 1 TO 15000: NEXT
  CIRCLE (x1, y1), 2, 0: PAINT (x1, y1), 0, 0
  CIRCLE (x2, y2), 2, 0: PAINT (x2, y2), 0, 0
  IF INKEY$ = " " THEN END
NEXT: COLOR 2
CIRCLE (425, 220), 2, 15: PAINT (425, 220), 15, 15
CIRCLE (440, 220), 2, 14: PAINT (440, 220), 14, 14
PRINT " Встреча двух картингов на линии старта произошла через 264
      секунды,";
PRINT " причем первый картинг сделал 6 кругов, а второй – 7 кругов."
```

Результат работы программы:

```
v1 = 54 км/ч = 15 м/с              v2 = 72 км/ч = 20 м/с
r1 = 105 м                         r2 = 120 м
```


```
t = 264 c
Встреча двух картингов на линии старта произошла через 264 секунды,
причем первый картинг сделал 6 кругов, а второй – 7 кругов.
```

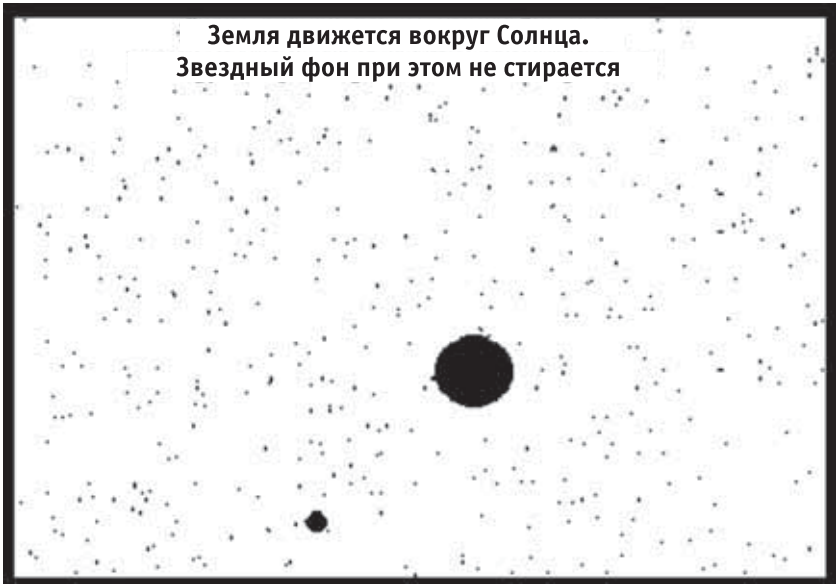
21. Планета

Комментарий. С помощью оператора GET создается и запоминается образ голубой и черной (т. е. невидимой на черном фоне звездного неба) планеты. С помощью оператора PUT с суффиксом XOR (который позволяет не изменять фон, на котором дви-

жется объект) образ планеты восстанавливается на экране. Такой способ реализации движения объекта по экрану без нарушения фонового изображения называют *спрайтовой графикой*, а сам такой движущийся объект — *спрайтом*.

```
SCREEN 9: DIM earth(400), earth1(400)
CIRCLE (50, 50), 5, 9: PAINT (50, 50), 9, 9
GET (37, 40)-(63, 60), earth
CIRCLE (150, 50), 5, 0: PAINT (150, 50), 0, 0
GET (137, 40)-(163, 60), earth1
CLS : CIRCLE (320, 150), 20, 14: PAINT (320, 150), 14, 14
FOR j = 1 TO 1000
  x1 = INT(RND * 639 + 1): y1 = INT(RND * 339 + 1): PSET (x1, y1), 15
NEXT j
LOCATE 1, 22: PRINT "Земля движется вокруг Солнца."
LOCATE 2, 18: PRINT "Звездный фон при этом не стирается."
DO
  FOR a = 0 TO 6.28 STEP .01
    x = 220 * COS(a) + 320: x2 = 220 * COS(a) + 320
    y2 = 60 * COS(a) + 150: y = 60 * SIN(a) + 150
    PUT (x, y), earth, XOR: FOR i = 1 TO 5000: NEXT
    PUT (x, y), earth, XOR: PUT (x2, y2), earth1, XOR
    IF INKEY$ = " " THEN END
NEXT
LOOP
```

Результат работы программы:



22. Анаграммы

Комментарий. Предлагаемая ниже программа универсальна, так как может выводить анаграммы не только 4-буквенных слов, но и 5- и 6-буквенных (все 7-буквенные слова не уместятся на экранной странице).

```
SCREEN 9: COLOR 14, 1: PRINT " Программа печати анаграмм"
INPUT " Анаграммы какого слова вы хотите узнать"; s$: s = LEN(s$)
IF s <= 2 THEN 1
IF s = 4 THEN n = 4
IF s = 5 THEN n = 3
IF s = 6 THEN n = 2
DIM p(s)
FOR i = 1 TO s: a$(i) = MID$(s$, i, 1): NEXT
FOR i = 1 TO s: p(i) = s + 1 - i: NEXT
z = 1: FOR i = 1 TO s: z = z * i: NEXT
PRINT : PRINT " Слово "; s$; "имеет анаграмм - "; z; : PRINT : PRINT
y = 1: GOSUB 2
FOR y = 2 TO z
  i = 2: j = 1
100 IF p(i - 1) <= p(i) THEN i = i + 1: GOTO 100
110 IF p(j) <= p(i) THEN j = j + 1: GOTO 110
  SWAP p(i), p(j): IF i = 2 THEN 120
  FOR l = 1 TO (i - 1) / 2: SWAP p(l), p(i - l): NEXT
120 GOSUB 2
NEXT y
1 END
2 REM *** формирование анаграмм ***
FOR e = 1 TO s: PRINT a$(p(e)); : NEXT
PRINT STRING$(n, " ");
FOR i = 1 TO 1000: NEXT: RETURN
```

Результат работы программы:

```
Программа печати анаграмм
Анаграммы какого слова вы хотите узнать? сорт
Слово сорт имеет анаграмм - 24

трос    ртос    торс   отрс   ротс   ортс   трсо   ртсо   тсро   стро
рсто    срто    тоср   отср   тсор   стор   остр   сотр   рост   орст
рсот    срот    осрт   сорт
```

23. Сложение многозначных чисел

Комментарий. Ясно, что сгенерировать два 36-значных числа в языке QBasic невозможно, даже если работать с данными двойной точности. Можно сгенерировать 12 трехзначных чисел для первого слагаемого и столько же — для второго, преобразовать их в символьные переменные, «склеить» их друг с другом и вывести одно слагаемое под другим. Затем в цикле от 1 до 12 нужно суммировать очередные тройки разрядов друг с другом,

запоминать эту сумму, преобразовывать ее в символьную переменную и выводить на экран либо сразу (если эта сумма не превосходит 1000), либо преобразовав ее в 3-символьную запись и к младшему разряду суммы следующей триады разрядов не забыв прибавить 1. (На самом деле алгоритм решения задачи чуть сложнее, но эти сложности касаются вывода результата сложения на экран.)

```
CLS : LOCATE 1, 25: PRINT "Поразрядное сложение 36-значных чисел"
DIM a(12), b(12), a$(12), b$(12), s(12), s$(12): RANDOMIZE TIMER
FOR i = 1 TO 12
  a(i) = INT(RND * 899 + 100): a$(i) = STR$(a(i)): a$ = a$ + a$(i)
  b(i) = INT(RND * 899 + 100): b$(i) = STR$(b(i)): b$ = b$ + b$(i)
  s(i) = a(i) + b(i)
NEXT
LOCATE 3, 20: PRINT a$: LOCATE 5, 20: PRINT b$: LOCATE 4, 19
PRINT "+": LOCATE 6, 19: PRINT STRING$(50, "-"): PRINT : PRINT : c = 0
FOR i = 12 TO 1 STEP -1
  s(i) = s(i) + c
  IF s(i) > 1000 THEN
    c = 1: s$(i) = STR$(s(i) - 1000)
  ELSE
    c = 0: s$(i) = STR$(s(i))
  END IF
  IF LEN(s$(i)) = 3 THEN s$(i) = "0" + RIGHT$(s$(i), 2)
NEXT
FOR i = 1 TO 12: s$ = s$ + s$(i): NEXT
IF s(1) > 1000 THEN s$ = "1" + s$
s = LEN(s$): t = 67 - s: k = 0
FOR i = 67 TO t + 1 STEP -1
    k = k + 1: LOCATE 7, i: PRINT RIGHT$(s$, k): SLEEP 1
NEXT
```

Результат работы программы:

```
            Поразрядное сложение 36-значных чисел:

      + 391 790 664 617 539 768 798 302 248 298 629 332
        766 356 871 675 771 891 199 970 312 201 671 583
      -----------------------------------------------------
        1 158 147 536 293 311 659 998 272 560 500 300 915
```

24. Кочерга

Контрольные примеры: «1 кочерга», «101 кочерга», «11 кочерёг», «307 кочерёг», «202 кочерги», «1404 кочерги».

Комментарий. Возможны разные способы решения этой лингвистической задачи. По ее условию ясно, что в ней возможны всего три формы согласования слова «кочерга» с числительными, а значит, все числительные надо разбить на три класса.

Кроме того, все числа, начиная с трехзначных и выше, согласовываются со словом «кочерга» так же, как числа, меньшие 100, и так же, как натуральные числа от 100 до 200.

Алгоритм решения задачи:

1) очистка экрана и вывод названия программы;
2) формирование трех правильных согласований;
3) ввод числа *N*;
4) преобразование числа *N* в символьную переменную *x*$;
5) запоминание копии этой символьной переменной как *y*$;
6) если переменная *x*$ содержит 1 или 2 символа, то превращение переменной *x*$ в трехсимвольную переменную, первый символ которой «1»;
7) выделение из *x*$ последнего символа и двух последних символов (причем эти два последних символа преобразуются в вещественную переменную);
8) если последний символ «1», а две последние цифры не равны 11, то число *N* согласовывается со словом «кочерга»; вывод результата и завершение работы программы;
9) если последний символ «2», «3» или «4», а две последние цифры меньше 5 или больше 21, то число *N* согласовывается со словом «кочерги»; вывод результата и завершение работы программы;
10) во всех остальных случаях число *N* согласовывается со словом «кочерёг»; вывод результата и завершение работы программы.

```
SCREEN 9: a$ = "кочерга": b$ = "кочерги": c$ = "кочерёг"
LOCATE 1, 8: COLOR 14, 0: PRINT "Привет!";
PRINT " А я умею согласовывать слово "; CHR$(34); a$; CHR$(34);
PRINT " с числительными."
LOCATE 3, 8: COLOR 5, 0: INPUT "Хочешь проверить (y/n)"; t$
IF t$ = "n" OR t$ = "N" THEN END
COLOR 10, 0: PRINT
INPUT "Ну, сколько тебе выдать кочерёжек"; x$
y$ = x$: x1 = LEN(x$)
IF x1 = 1 THEN x$ = "10" + x$
IF x1 = 2 THEN x$ = "1" + x$
z1$ = RIGHT$(x$, 1): z2$ = RIGHT$(x$, 2): z2 = VAL(z2$)
IF z1$ = "1" AND z2$ <> "11" THEN 1
IF (z1$ = "2" OR z1$ = "3" OR z1$ = "4") AND (z2 > 21 OR z2 < 5)
   THEN 2 ELSE 3
1 COLOR 5, 0: PRINT : PRINT " Правильное написание: "; y$;" "; a$: END
2 COLOR 6, 0: PRINT : PRINT " Правильное написание: "; y$;" "; b$: END
3 COLOR 4, 0: PRINT : PRINT " Правильное написание: "; y$;" "; c$: END
```

Результат работы программы:

```
Привет! А я умею согласовывать слово "кочерга" с числительными.
Хочешь проверить (y/n)? y
Ну, сколько тебе выдать кочерёжек? 1953
Правильное написание: 1953 кочерги
```

25. Палиндромы

Контрольные примеры:

1) «АРГЕНТИНА МАНИТ НЕГРА» — перевертыш;
2) «УЖ Я ВЕНИКИ НЕ ВЯЖУ» — перевертыш;
3) «ЛЁША НА ПОЛКЕ КЛОПА НАШЁЛ» — перевертыш;
4) «А В ЕНИСЕЕ СИНЕВА» — перевертыш;
5) «А В ЕНИСЕЕ ОТРАЖАЛАСЬ СИНЕВА» — не перевертыш.

Алгоритм решения задачи:

1) очистка экрана, вывод названия программы;
2) ввод испытуемой фразы, вычисление ее длины;
3) резервирование места в памяти под все буквы испытуемой фразы;
4) цикл по *i* от 1 до номера последней буквы фразы:
   выделение очередного символа фразы,
   если этот символ — «пробел», то его замена «пустым» символом (""),
   формирование новой фразы, не содержащей пробелов;
5) вычисление длины новой фразы;
6) вычисление позиции символа в середине новой фразы;
7) цикл по *i* от 1 до середины новой фразы:
   выделение *i*-й буквы с начала фразы и симметричной ей буквы с конца фразы,
   если эти буквы не совпадают, то вывод сообщения, что данная фраза не является палиндромом, и завершение работы программы, иначе — переход к п. 8;
8) вывод сообщения: «Исходная фраза — перевертыш».

```
CLS : PRINT "Перевертыши (палиндромы)"
INPUT "Какую фразу будем испытывать"; a$
a = LEN(a$): DIM m$(a)
REM **** Удаление пробелов из фразы (формирование слова t$) ****
FOR i = 1 TO a
  m$ = MID$(a$, i, 1)
  IF m$ = " " THEN m$ = ""
  t$ = t$ + m$
NEXT
a = LEN(t$): b = a \ 2: PRINT t$
REM **** Выяснение факта, является ли слово t$ палиндромом ****
```

```
FOR i = 1 TO b
  b$ = MID$(t$, i, 1): n = a - i + 1: c$ = MID$(t$, n, 1)
  IF b$ <> c$ THEN
    PRINT a$; " - не перевертыш": END
  END IF
NEXT
PRINT a$; " - перевертыш"
```

26. Кроссворд

Контрольные примеры:

1) СЕМЬЯ, НАЯДА, ВАГОН, КАНВА, ВИСОК, ГОМОН;

2) DRILL (борозда), INERT (инертный), ERECT (прямой), MATEY (приятель), DREAM (сон), LATHY (худой).

Результаты работы программы для контрольных примеров:

В	А	Г	О	Н
И		О		А
С	Е	М	Ь	Я
О		О		Д
К	А	Н	В	А

D	R	I	L	L
R		N		A
E	R	E	C	T
A		R		H
M	A	T	E	Y

Комментарий. Создаваемый кроссворд размером 5 × 5 имеет два симметричных решения относительно диагонали, проведенной из верхнего левого угла в правый нижний угол. Любое из этих решений считается правильным.

Это довольно сложная задача, если заранее не выработать следующие принципы ее решения:

1) присвоить словам по горизонтали метку a_i, а словам по вертикали — метку b_i, где $i \in [1, 5]$;

2) во всех словах запомнить 1, 3 и 5-ю буквы;

3) найти и «сцепить» три слова в виде лежащей на боку буквы «П» (слова a_1, b_1 и a_5);

4) найти слово с меткой b_3;

5) найти слово с меткой a_3;

6) оставшемуся слову присвоить метку b_5;

7) вывести результат в заранее заготовленной таблице.

Конечно, алгоритм решения этой задачи несколько сложнее, чем указанные выше принципы ее решения.

```
SCREEN 12: COLOR 14: LOCATE 1, 33: PRINT " Кроссворд ": COLOR 15
REM *** слова для отладки программы ***
```

```
'a$(1) = "DRILL": a$(2) = "INERT": a$(3) = "ERECT"
'a$(4) = "MATEY": a$(5) = "DREAM": a$(6) = "LATHY"
'a$(1) = "СЕМЬЯ": a$(2) = "НАЯДА": a$(3) = "ВАГОН"
'a$(4) = "КАНВА": a$(5) = "ВИСОК": a$(6) = "ГОМОН"
'a$(1) = "УКСУС": a$(2) = "РЫВОК": a$(3) = "СОВОК"
'a$(4) = "ПАКЛЯ": a$(5) = "УКРОП": a$(6) = "САКЛЯ"
'a$(1) = "АВРАЛ": a$(2) = "КАБАН": a$(3) = "ТАТРА"
'a$(4) = "АСКЕТ": a$(5) = "РОБОТ": a$(6) = "ЛЕНТА"
FOR i = 1 TO 6
  PRINT "Какое"; i; "-ое слово"; : INPUT " "; a$(i)
  L$(i) = LEFT$(a$(i), 1)    ' первая буква слова a$(i)
  m$(i) = MID$(a$(i), 3, 1)  ' третья буква слова a$(i)
  R$(i) = RIGHT$(a$(i), 1)   ' последняя буква слова a$(i)
NEXT
FOR i = 1 TO 6
  FOR k = 2 TO 6
    FOR n = 1 TO 6
      IF i <> k AND k <> n AND i <> n AND L$(i) = L$(k)
         AND R$(k) = L$(n) THEN
        a1$ = a$(i): b1$ = a$(k): a5$ = a$(n)
      END IF
    NEXT
  NEXT
NEXT
FOR i = 1 TO 6
  IF RIGHT$(a1$, 1) = L$(i) AND RIGHT$(a5$, 1) = R$(i)
  THEN b3$ = a$(i)
NEXT
FOR i = 1 TO 6
  IF MID$(a1$, 3, 1) = L$(i) AND MID$(a5$, 3, 1) = R$(i)
  THEN b2$ = a$(i)
NEXT
FOR i = 1 TO 6
  IF MID$(b1$, 3, 1) = L$(i) AND MID$(b3$, 3, 1) = R$(i)
  THEN a3$ = a$(i)
NEXT
c1$ = LEFT$(a1$, 1) + " " + MID$(a1$, 2, 1) + " " + MID$(a1$, 3, 1)
c1$ = c1$ + " " + MID$(a1$, 4, 1) + " " + RIGHT$(a1$, 1)
c3$ = LEFT$(a3$, 1) + " " + MID$(a3$, 2, 1) + " " + MID$(a3$, 3, 1)
c3$ = c3$ + " " + MID$(a3$, 4, 1) + " " + RIGHT$(a3$, 1)
c5$ = LEFT$(a5$, 1) + " " + MID$(a5$, 2, 1) + " " + MID$(a5$, 3, 1)
c5$ = c5$ + " " + MID$(a5$, 4, 1) + " " + RIGHT$(a5$, 1)
c2$ = MID$(b1$, 2, 1) + " " + MID$(b2$, 2, 1) + " " + MID$(b3$, 2, 1)
c4$ = MID$(b1$, 4, 1) + " " + MID$(b2$, 4, 1) + " " + MID$(b3$, 4, 1)
LOCATE 11, 30: PRINT c1$: LOCATE 12, 30: PRINT c2$
LOCATE 13, 30: PRINT c3$: LOCATE 14, 30: PRINT c4$: LOCATE 15, 30
PRINT c5$
FOR x = 225 TO 325 STEP 23
  FOR y = 158 TO 234 STEP 16
    LINE (x, y)-(x + 23, y + 16), 7, B
  NEXT
NEXT
```

27. Пароль доступа

Комментарий. Используются вспомогательные переменные: *r*$ — весь пароль записан русскими буквами, *e*$ — весь пароль записан английскими буквами. В цикле от 1 до 5 выявляются все возможные варианты написания пароля согласно условию задачи (всего возможно 10 таких вариантов), и для каждого из них вычисляется сумма кодов всех его букв.

```
CLS : COLOR 10: PRINT TAB(28); "Задача про пароль"
r$ = "ВЕСНА": e$ = "BECHA"
FOR i = 1 TO 5
  r$(i) = MID$(r$, i, 1): e$(i) = MID$(e$, i, 1)
NEXT
DATA 1,2,1,3,1,4,1,5,2,3,2,4,2,5,3,4,3,5,4,5
WHILE m < 11
  m = m + 1
  FOR i = 1 TO 2
    READ m(i)
NEXT
k = m(1): n = m(2)
FOR i = 1 TO 5
  a$(i) = MID$(r$, i, 1)
  b$(i) = MID$(e$, i, 1)
  IF i <> k AND i <> n THEN
      SWAP a$(i), b$(i)
      LOCATE m + 2, i + 30: COLOR 15: PRINT a$(i)
    ELSE
      LOCATE m + 2, i + 30: COLOR 14: PRINT a$(i)
    END IF
    s = s + ASC(a$(i)): LOCATE m + 2, 36: COLOR 15: PRINT s
  NEXT
  s = 0: IF m = 10 THEN END
WEND
```

Результат работы программы:

```
                    Задача про пароль
                       ВЕСНА  467
                       ВЕСНА  481
                       ВЕСНА  472
                       ВЕСНА  466
                       ВЕСНА  481
                       ВЕСНА  472
                       ВЕСНА  466
                       ВЕСНА  486
                       ВЕСНА  480
                       ВЕСНА  471
```

28. Перевод «16 → 10»

Контрольные примеры:
1) $0{,}FFF_{16} = 0{,}999755859375_{10}$;
2) $ABCD_{16} = 43981_{10}$.

```
CLS : PRINT «Программа перевода 16 => 10»
DEF SEG = 0
                                  '*********************************
POKE &H417, PEEK(&H417) OR 64  '** включение клавиши Caps Lock **
DEF SEG
                                  '*********************************
INPUT "Какое шестнадцатеричное число вы хотите преобразовать
      в десятичное"; a$
a = LEN(a$)
FOR i = 1 TO a
  a$(i) = MID$(a$, i, 1)
  IF a$(i) = "." THEN k = i
  a(i) = VAL(a$(i))
NEXT
'k – место, на котором стоит десятичная точка
IF a$(k) = "." THEN n = k - 2 ELSE n = a - 1
IF k = 2 THEN n = -1
FOR i = 1 TO a
  IF a$(i) = "A" THEN a(i) = 10
  IF a$(i) = "B" THEN a(i) = 11
  IF a$(i) = "C" THEN a(i) = 12
  IF a$(i) = "D" THEN a(i) = 13
  IF a$(i) = "E" THEN a(i) = 14
  IF a$(i) = "F" THEN a(i) = 15
NEXT
IF k = 0 THEN
  REM *** Формирование целого десятичного числа ***
  FOR i = 1 TO a
    b = a(i) * 16 ^ n: n = n - 1: r# = r# + b
  NEXT
ELSE
  REM *** Формирование дробного десятичного числа ***
  FOR i = 1 TO k - 1
    b = a(i) * 16 ^ n: n = n - 1: s# = s# + b
  NEXT: t = -1
  FOR j = k + 1 TO a
    c# = a(j) * 16 ^ t: t = t - 1: f# = f# + c#
  NEXT
  r# = s# + f#
END IF
LOCATE 10, 20: PRINT "*****************************************"
LOCATE 11, 20: PRINT "*             СИСТЕМА СЧИСЛЕНИЯ         *"
LOCATE 12, 20: PRINT "*****************************************"
LOCATE 13, 20: PRINT "*          16         *         10        *"
LOCATE 14, 20: PRINT "*****************************************"
LOCATE 15, 20: PRINT "*                     *                   *"
```

```
LOCATE 16, 20: PRINT "*****************************************"
LOCATE 15, 23: PRINT a$
LOCATE 15, 31: PRINT USING «#####.############»; r#
```

Результат работы программы:

Система счисления	
16	10
0.00FDE	0.003873825073

29. Неправильное сокращение дробей

Результат работы программы:

```
Неправильное сокращение. Программа работает около 2 минут.
Не волнуйтесь и ждите.
220 / 121 = 20 / 11    440 / 143 = 40 / 13    440 / 341 = 40 / 31
550 / 253 = 50 / 23    550 / 451 = 50 / 41    770 / 473 = 70 / 43
770 / 671 = 70 / 61    880 / 187 = 80 / 17    880 / 583 = 80 / 53
880 / 781 = 80 / 71
```

```
CLS: PRINT "Неправильное сокращение"
DIM x(90), y(90), v(90), n(90)
FOR x = 100 TO 999
  FOR y = 101 TO 999
    x$ = STR$(x): a = VAL(MID$(x$, 2, 1))
    b = VAL(MID$(x$, 3, 1)): c = VAL(MID$(x$, 4, 1))
    y$ = STR$(y): m = VAL(MID$(y$, 2, 1))
    a1 = VAL(MID$(y$, 3, 1)): n = VAL(MID$(y$, 4, 1))
    IF a = a1 AND a <> m THEN
      verh = 10 * b + c: niz = 10 * m + n: GOTO 3
    ELSE
      GOTO 2
    END IF
3  p = verh: q = niz:
   IF x * niz = y * verh THEN
    k = k + 1: x(k) = x: y(k) = y: v(k) = verh: n(k) = niz
   END IF
2 NEXT
NEXT
FOR k = 1 TO 90
    p = v(k): q = n(k): p1 = p: q1 = q
4 IF p1 = q1 THEN 6 ELSE 5
5 IF p1 > q1 THEN
    p1 = p1 - q1: GOTO 4
  ELSE
    q1 = q1 - p1: GOTO 4
  END IF
6 d = p1
  IF d = 1 THEN
    PRINT x(k); "/"; y(k); "="; v(k); "/"; n(k),
  END IF
NEXT
```

30. Бегущие человечки

Контроль результатов работы программы — визуальный.

```
SCREEN 9: COLOR 14, 1: LOCATE 10, 32: PRINT "Бегущие человечки"
b2$ = " () ": c1$ = " /[]\_FOOT": cr$ = "BALL_/[]\ "
d1$ = " />": dr$ = " <\ ": e1$ = " //": er$ = " \\ "
c2$ = "/[]\": LOCATE 15, 4: PRINT STRING$(73, "-")
LOCATE 12, 3: PRINT b2$: LOCATE 12, 73: PRINT b2$
LOCATE 13, 3: PRINT c2$: LOCATE 13, 73: PRINT c2$
LOCATE 14, 2: PRINT d1$: LOCATE 14, 74: PRINT dr$
FOR i = 1 TO 29
  LOCATE 12, i + 3: PRINT b2$: LOCATE 12, 73 - i: PRINT "  "; b2$
  LOCATE 13, i + 2: PRINT c1$: LOCATE 13, 70 - i: PRINT cr$
  a1 = (i + 3) / 2: a = a1 - INT(a1)
  b1 = (73 - i) / 2: b = b1 - INT(b1)
  IF a = 0 OR b = 0 THEN
    LOCATE 14, i + 2: PRINT d1$: LOCATE 14, 74 - i: PRINT dr$
  ELSE
    LOCATE 14, i + 2: PRINT e1$: LOCATE 14, 74 - i: PRINT " "; er$; " "
  END IF
  FOR k = 1 TO 100000: NEXT
NEXT: LOCATE 14, 32: PRINT "_||_": LOCATE 14, 46: PRINT "_||_"
```

31. Признак делимости на 11

Контрольные примеры:

1) число 24753487 — кратно 11;
2) число 878790 — кратно 11;
3) число 24753486 — не кратно 11.

Комментарий. Прежде всего, следует отметить, что признак делимости на 11 можно сформулировать короче: если разность суммы цифр многозначного натурального числа, стоящих на четных и на нечетных местах кратна 11, то само это число кратно 11. Далее реализовать его проверку в программе достаточно просто.

```
1 SCREEN 9: COLOR 14, 1: PRINT : PRINT " Делимость на 11"
PRINT " Какое число, содержащее не более 9 знаков,"
PRINT " будем испытывать на кратность числу 11"
LOCATE 4, 41: INPUT a$: a = LEN(a$)
IF a > 9 THEN 1
FOR i = 1 TO a
  a$(i) = MID$(a$, i, 1): a(i) = VAL(a$(i))
  b$(i + 1) = MID$(a$, i + 1, 1): b(i + 1) = VAL(b$(i + 1))
NEXT
FOR i = 1 TO a STEP 2
  s1 = s1 + a(i):
  s2 = s2 + b(i + 1)
```

```
NEXT: PRINT
PRINT " S1 ="; s1: PRINT " S2 ="; s2
delta = ABS(s1 - s2): PRINT " |S1 - S2| = "; delta
d = ABS(s1 - s2): d1 = 11 - d
IF s1 = s2 OR d1 = 0 THEN
  PRINT " Число "; a$; " кратно числу 11.": END
ELSE
  PRINT " Данное число не делится на 11."
END IF
```

32. Вовочка на катере

Ответ: Вовочка прибудет в пункт B примерно в 15 часов 38 минут.

Алгоритм решения задачи:

1) формирование данных: $L = 50$ км; $V_{\text{реки}} = 5$ км/ч; $V_{\text{кат}} = 15$ км/ч; $h = 12$ ч (время старта); $V_1 = V_{\text{по теч}}$; $V_2 = V_{\text{против теч}}$;
2) рисование реки (синий прямоугольник); вывод направления течения реки; изображение местоположения пунктов A и B;
3) цикл по t (в минутах) от 0 до 60:
   если путь S, пройденный Вовочкой, равен 50 км, то завершение работы программы,
   если $t \leqslant 45$, то движение катера производится по течению реки с одновременным вычислением расстояния $S = a + V_1 \cdot t$ от пункта A до текущего местоположения катера; подсчитывается V_0 — скорость катера в данный момент времени; вычисляется абсцисса x (в пикселях) — местоположение изображения катера на экране; вызывается подпрограмма, изображающая движение катера по течению реки;
   если $t > 45$, то выполняются аналогичные действия, но против течения реки;
4) определение времени (в часах), прошедшего с момента старта (увеличение значения счетчика: $h = h + 1$);
5) запоминание расстояния a (в км) от пункта A до текущего местонахождения катера;
6) переход к п. 3.

```
SCREEN 9: LOCATE 5, 4: PRINT "V = 5 км/ч"
L = 50: vr = 5: vk = 15: h = 12: v1 = (vr + vk) / 60
v2 = (vk - vr) / 60
LINE (0, 100)-(640, 120), 1, BF: LOCATE 6, 4: PRINT " - - - - - ->"
LINE (25, 100)-(30, 106), 5, BF: LOCATE 7, 4: PRINT "A"
LINE (525, 120)-(530, 114), 5, BF: LOCATE 10, 67: PRINT "B"
1 FOR t = 0 TO 60
```

```
    IF s >= l THEN
      CIRCLE (x + 26, 110), 3, 14: PAINT (x + 26, 110), 14, 14
      LOCATE 16, 10: PRINT "Всё, приплыли! ": END
    END IF
    IF t <= 45 THEN
      s = a + v1 * t: b = s: v0 = vr + vk: x = 10 * s: GOSUB 2
    END IF
    IF t > 45 THEN
      s = b - v2 * (t - 45): v0 = vk - vr: x = 10 * s: GOSUB 2
    END IF
  NEXT: h = h + 1: a = s: GOTO 1
  2 COLOR 5, 0: LOCATE 12, 25: PRINT h; ": "; : PRINT USING "##"; t
  COLOR 14, 0: LOCATE 14, 10: PRINT "Катер удалился от пункта А на ";
  COLOR 9, 0: PRINT USING "##.###"; s: COLOR 14, 0
  LOCATE 14, 48: PRINT "км"
  COLOR 2, 0: LOCATE 16, 10: PRINT "Скорость катера в данный момент ";
  COLOR 9, 0: PRINT v0: COLOR 2, 0: LOCATE 16, 46: PRINT "км/ч"
  CIRCLE (x + 26, 110), 3, 14: PAINT (x + 26, 110), 14, 14
  FOR i = 1 TO 1000: NEXT
  CIRCLE (x + 26, 110), 3, 1:PAINT (x + 26, 110), 1, 1: RETURN
```

33. Зеркальное отображение текста

Комментарий. В этой задаче требуется «перевернуть текст вверх ногами» (что и реализовано в теле последнего цикла программы).

```
SCREEN 9: PRINT
PRINT "Зеркальное отображение текста относительно горизонта"
LOCATE 5, 1: INPUT "Какой текст вы хотели бы отобразить"; a$
COLOR 10: a = LEN(a$): CLS : c = (80 - a) \ 2 : LOCATE 12, c: PRINT a$
DIM x(6000), y(6000), c(6000), z(6000)
REM ****** сканирование ******
FOR x = 190 TO 450: FOR y = 150 TO 170
  n = n + 1: x(n) = PMAP(x, 0): y(n) = PMAP(y, 1): c(n) = POINT(x, y)
NEXT y, x
FOR i = 1 TO n
  z(i) = 340 - y(i): PSET (x(i), z(i)), c(i)
NEXT
LINE (190, 170)-(450, 170), 14
```

Результат работы программы:

```
Зеркальное отображение текста относительно горизонта
Какой текст вы хотели бы отобразить? ЛЁША НА ПОЛКЕ КЛОПА НАШЁЛ
```

ЛЁША НА ПОЛКЕ КЛОПА НАШЁЛ

ЛЁША НА ПОЛКЕ КЛОПА НАШЁЛ

34. Шарикоподшипник

Комментарий. Шарикоподшипник — это симметричная фигура, поэтому его центр лучше всего поместить в середину экрана — точку $O(320, 220)$.

На рис. 10 ниже изображен фрагмент подшипника, у которого в обойме n шариков. Для простоты решения задачи будем допускать касания шариков между собой (хотя в реальном подшипнике это не допускается, поэтому шарики разделяются еще одной деталью — сепаратором).

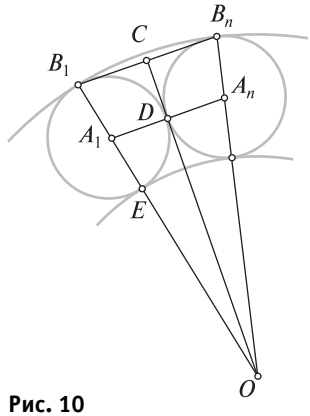


Рис. 10

Пусть радиус внешней окружности $r = |OB_1| = 200$, радиус внутренней окружности $r_1 = |OE|$, а радиус шарика $r_2 = |A_1E|$. Тогда легко понять, что радиус окружности, на которой располагаются центры шариков, $r_3 = r_1 + r_2 = |OA_1|$. Ясно также, что $r = r_1 + 2r_2$. Очевидно, что величина угла DA_1O, где D — точка касания двух соседних шариков, равна $\varphi = \pi/n$. Из $\triangle DA_1O$ в этом случае следует, что

$$\sin\varphi = \frac{|A_1D|}{|A_1O|} \Leftrightarrow \sin\varphi = \frac{r_2}{r_1 + r_2}.$$

После несложных преобразований получим:

$$r_1 = \frac{r(1 - \sin\varphi)}{1 + \sin\varphi}; \quad r_2 = \frac{r\sin\varphi}{1 + \sin\varphi}; \quad r_3 = r_1 + r_2.$$

Центры шариков находятся на окружности радиуса r_3 и отстоят друг от друга на угол 2φ.

```
SCREEN 12: PRINT "Шарикоподшипник"
1 INPUT "Сколько шариков будем заказывать (от 6 до 30)"; n
IF n < 6 OR n > 30 THEN 1
DIM x(n + 1), y(n + 1): pi = 3.141593: f = pi/n
s = SIN(f): r = 200 : r1 = r*(1 - s)/(1 + s): r2 = r*s / (1 + s)
r3 = r1 + r2
CIRCLE (320, 220), r, 14: CIRCLE (320, 220), r1, 14
REM *** вычисление координат центров шариков***
FOR t = 0 TO (2 * pi) STEP (2 * f)
  i = i + 1: x(i) = r3 * COS(f) + 320
  y(i) = r3 * SIN(f) + 220
NEXT
REM *** рисование шариков***
FOR i = 1 TO n
  CIRCLE (x(i), y(i)), r3, 14
NEXT
```

35. Летящее копье

Комментарий. Из курса физики известно, что тело, брошенное под углом к горизонту, летит по параболе. Поэтому сначала запишем уравнение параболы, проходящей через три точки на экране. Общее уравнение параболы имеет вид: $y = ax^2 + bx + c$, тогда для определения коэффициентов a, b и c достаточно подставить в это уравнение координаты трех выбранных точек. В результате получится система из трех линейных уравнений с тремя неизвестными a, b и c, решив которую можно найти все коэффициенты параболы.

Из курса алгебры известно, что тангенс угла наклона касательной к кривой $y = f(x)$ в точке K равен значению производной этой функции в точке с абсциссой $x_к$, т. е. $\operatorname{tg} \varphi = f'(x_к)$. Так как $y' = 2ax + b$, то угол наклона касательной в текущей точке $K(x, y)$ можно вычислить по формуле: $\varphi = \operatorname{arctg}(2ax + b)$.

После этого выбирается длина копья, т. е. $|AB| = r$, вычисляются координаты концов копья: $A(x - r \cdot \cos\varphi,\ y - r \cdot \sin\varphi)$ и $B(x + r \cdot \cos\varphi,\ y + r \cdot \sin\varphi)$. А заставить копье лететь совсем нетрудно...

```
SCREEN 9: r = 20: PRINT "Копье"
FOR x = 100 TO 600 STEP 2
  y = .00175 * x * x - 1.425 * x + 325 'уравнение параболы
  alfa = ATN(2 * .00175 * x - 1.425) 'угол наклона касательной в точке x
  xa = x - r * COS(alfa): ya = y - r * SIN(alfa) ' координаты концов
  xb = x + r * COS(alfa): yb = y + r * SIN(alfa) 'копья
  LINE (xa, ya)-(xb, yb), 14
  FOR i = 100 TO 1500: NEXT i 'регулятор скорости полета копья
  LINE (xa, ya)-(xb, yb), 0
NEXT: LINE (xa, ya)-(xb, yb), 14
```

36. Зеркало

Комментарий. Это задача на статичную графику, а значит, возможностей реализовать нужную картинку — бесчисленное множество. Все определяется фантазией участника олимпиады.

```
SCREEN 12: LOCATE 1, 37: COLOR 14: PRINT "Свеча и зеркало"
REM --- свеча ---
LINE (460, 120)-(460, 225), 7
CIRCLE (470, 120), 10, 7, 3.14, 6.28, .3
LINE (480, 120)-(480, 225), 7
CIRCLE (470, 225), 10, 7, 3.14, 6.28, .3
REM --- столешница ---
LINE (100, 300)-(460, 315), 7, B: LINE (460, 300)-(600, 135), 7
LINE (460, 315)-(600, 150), 7: LINE (600, 150)-(600, 135), 7
LINE (100, 300)-(260, 135), 7: LINE (270, 135)-(260, 135), 7
```

```
LINE (416, 135)-(460, 135), 7: LINE (480, 135)-(600, 135), 7
REM --- зеркало ---
LINE (320, 225)-(440, 180), 7: LINE (320, 225)-(245, 90), 7
LINE (440, 180)-(365, 45), 7: LINE (365, 45)-(245, 90), 7
LINE (240, 205)-(280, 195), 7: LINE (240, 205)-(272, 140), 7
LINE (290, 169)-(280, 195), 7: LINE (320, 210)-(260, 100), 7
LINE (320, 210)-(425, 170), 7: LINE (362, 60)-(425, 170), 7
LINE (362, 60)-(260, 100), 7: LINE (360, 172)-(328, 119), 7
LINE (375, 165)-(345, 112), 7: LINE (328, 119)-(345, 112), 7
LINE (360, 172)-(375, 165), 7: LINE (285, 147)-(334, 130), 7
LINE (350, 122)-(388, 108), 7
PAINT (470, 150), 14, 7: PAINT (360, 150), 14, 7
PAINT (340, 180), 4, 7: PAINT (420, 270), 4, 7
PAINT (320, 220), 9, 7: PAINT (260, 180), 1, 7
PAINT (260, 305), 8, 7: PAINT (465, 305), 8, 7
10 GOSUB 1: GOSUB 2: GOSUB 3: GOSUB 2
IF INKEY$ = " " THEN 20 ELSE 10
END
1 REM ---- пламя 1 ----
CIRCLE (470, 112), 7, 7, 1.57, 4.71, 2
CIRCLE (470, 97), 7, 7, 4.71, 1.7, 2
CIRCLE (470, 105), 15, 7, 4.71, 1.7, 2: PSET (321, 91)
DRAW "c7 r2 m+2,+1 m+2,+1 m+2,+1 m+2,+2 m+1,+2 m+1,+2 m+1,+2 m+1,+4
m+1,+2 d3"
PSET (321, 91): DRAW "c7 f2 m+2,+1 m+2,+3 m+1,+2 d4 m+1,+2 m+2,+2 f4"
PAINT (333, 106), 12, 7: PAINT (470, 110), 12, 7 : SLEEP 2
LINE (460, 120)-(480, 90), 0, BF: LINE (321, 90)-(338, 113), 0, BF
RETURN
2 REM ---- пламя 2 ----
CIRCLE (470, 105), 15, 7, 4.71, 1.7, 3
CIRCLE (470, 105), 15, 7, 1.7, 4.71, 3: PSET (321, 91)
DRAW "c7 r2 m+2,+1 m+2,+1 m+2,+1 m+2,+2 m+1,+2 m+1,+2 m+1,+2 m+1,+4
     m+1,+2 d3"
PSET (321, 91)
DRAW "c7 d4 m+1,+4 m+1,+2 m+1,+2 m+2,+2 m+3,+2 m+3,+2 f3"
PAINT (333, 106), 12, 7: PAINT (470, 110), 12, 7: SLEEP 2
LINE (460, 120)-(480, 90), 0, BF: LINE (321, 90)-(338, 113), 0, BF
RETURN
3 REM ---- пламя 3 ----
CIRCLE (470, 112), 7, 7, 4.71, 1.57, 2
CIRCLE (470, 97), 7, 7, 1.7, 4.71, 2
CIRCLE (470, 105), 15, 7, 1.7, 4.71, 2: PSET (321, 91)
DRAW "c7 d4 m+1,+4 m+1,+2 m+1,+2 m+2,+2 m+3,+2 m+3,+2 f3"
PSET (321, 91)
DRAW "c7 m+1,+2 m+1,+2 f2 m+3,+2 m+3,+2 f2 m+1,+1 m+1,+2 m+1,+3 d2"
PAINT (333, 106), 12, 7: PAINT (470, 110), 12, 7: SLEEP 2
LINE (460, 120)-(480, 90), 0, BF: LINE (321, 90)-(338, 113), 0, BF
RETURN
20 REM ---- финальная картинка ----
CIRCLE (470, 105), 15, 7, 4.71, 1.7, 3
CIRCLE (470, 105), 15, 7, 1.7, 4.71, 3: PSET (321, 91)
```

Рис. 11

```
DRAW "c7 r2 m+2,+1 m+2,+1 m+2,+1 m+2,+2 m+1,+2 m+1,+2 m+1,+2 m+1,+4
     m+1,+2 d3"
PSET (321, 91)
DRAW "c7 d4 m+1,+4 m+1,+2 m+1,+2 m+2,+2 m+3,+2 m+3,+2 f3"
PAINT (333, 106), 12, 7: PAINT (470, 110), 12, 7
```

Результат работы программы показан на рис. 11.

37. Летящий конверт

Комментарий. К сожалению, динамику этого процесса отобразить достаточно сложно, поэтому достаточно нарисовать дискретный ряд изображений летящего конверта (рис. 12).

```
SCREEN 12: PAINT (1, 1), 1: COLOR 14
a$ = " u20 r30 d20 l30 m+30,-20 l30 m+30,+20"
FOR y = 300 TO 200 STEP -10
  IF c = 360 THEN c = 0
  n = n + 1
  c = c + 30
  x = x + (10 * n)
  f$ = "ta" + STR$(c)
```

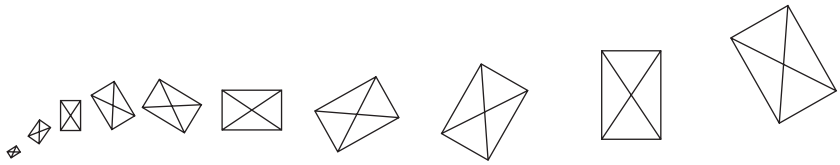


Рис. 12

```
  b$ = "s" + STR$(n)
  PSET (x + n, y), 7
  DRAW f$ + b$ + a$
  FOR t = 1 TO 50000: NEXT t
  CLS
  IF INKEY$ = " " THEN END
NEXT y
```

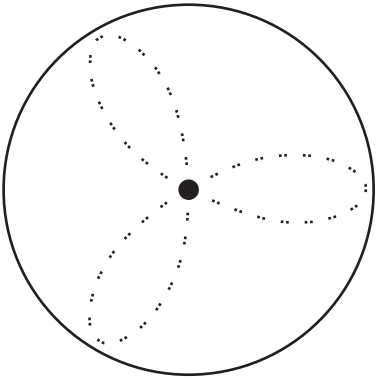
38. Корабельный винт

Решение (аналогично задаче 16):

```
SCREEN 12: PRINT "Корабельный винт"
INPUT "Вращение винта по часовой стрелке – 1, а против нее – 2.
      Что выбираем"; n
IF n = 1 THEN a = .2 ELSE a = -.2
1 CIRCLE (320, 220), 105, 14: CIRCLE (320, 220), 5, 14
c = 14: GOSUB 2: t = t + a: FOR i = 1 TO 10000: NEXT
  LINE (160, 60)-(470, 370), 0, BF
  CIRCLE (320, 220), 105, 14: CIRCLE (320, 220), 5, 14
  PAINT (320, 220), 14
  c = 0: GOSUB 2: GOTO 1
END
2 REM *** винт ***
FOR f = 0 TO 6.28 STEP .05
  X = 100 * COS(3 * f) * COS(f + t)
  Y = 100 * COS(3 * f) * SIN(f + t)
  PSET (X + 320, Y + 220), c
  IF INKEY$ = " " THEN END
NEXT: RETURN
```

Результат работы программы:

```
Корабельный винт
Как будем вращать: по часовой стрелке (1) или против (2)? 2
```



39. Закрашенная таблица

Комментарий. Это довольно простая задача — в ней всего четыре несложных момента:

1) расчет и построение таблицы (достаточно тиражировать объект — в данном случае квадратик — по горизонтали и по вертикали); запоминание координат центра каждой клетки;

2) генерация случайным образом номеров (допускается с повтором) клеток, которые тут же нужно окрасить в синий цвет;

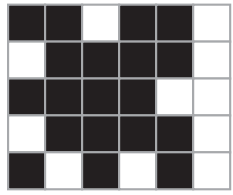
3) сканирование центров всех клеток для обнаружения нужного цвета и подсчет количества таких клеток;

4) вывод информации на экран.

```
SCREEN 12: PRINT "Закрашиваем таблицу M на N клеток."
RANDOMIZE TIMER
INPUT "Каковы ваши M и N (3<M<20 и 4<N<25)"; m, n
PRINT "Строим таблицу "; m; "*"; n
z = m * n: DIM xc(z), yc(z)
a = 60: c = 80
FOR x = a TO a + (n - 1) * 20 STEP 20
  FOR y = c TO c + (m - 1) * 20 STEP 20
    LINE (x, y)-(x + 20, y + 20), 15, B     'Построение таблицы
    i = i + 1: xc(i) = x + 10: yc(i) = y + 10
    'Координаты центров клеток
  NEXT y
NEXT x
u = i
FOR i = 1 TO z
  k = INT(RND * (i - 1) + 1)
  PAINT (xc(k), yc(k)), 9, 15
  SOUND 37, 1
NEXT
FOR y = yc(1) TO yc(u) STEP 20
  FOR x = xc(1) TO xc(u) STEP 20
    w = POINT(x, y)
    IF w = 9 THEN q = q + 1          'Подсчет числа закрашенных клеток
  NEXT
NEXT
LOCATE 4, 1: PRINT "Синих клеток -"; q; ", а черных -"; z - q
```

Результат работы программы:

```
Случайным образом закрашиваем таблицу M на N
Каковы ваши M и N (3<M<20 и 4<N<25)? 5, 6
Закрашенная таблица 5 * 6
Синих клеток - 19, а черных - 11
```



40. Путешествие по глобусу

Результат работы программы показан на рис. 13.

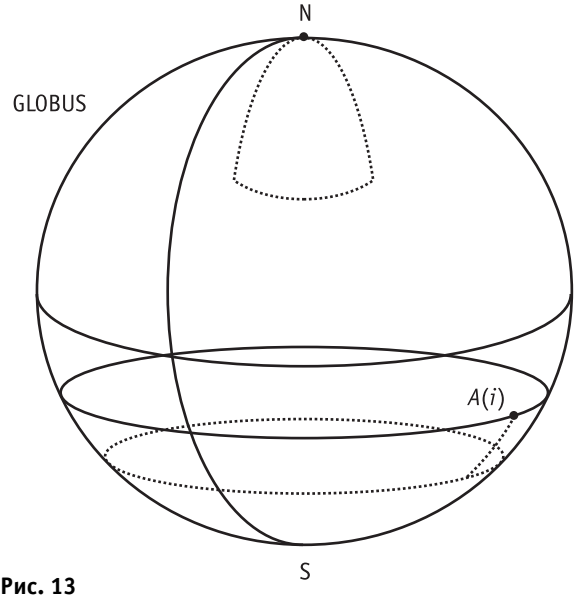


Рис. 13

```
SCREEN 12: PRINT "GLOBUS"
LOCATE 1, 41: PRINT "N": LOCATE 28, 41: PRINT "S"
CIRCLE (320, 220), 200, 7: CIRCLE (320, 220), 200, 7, 3.14, 0, .3
CIRCLE (320, 220), 200, 7, 1.57, 4.71, 2: SLEEP 2
LOCATE 20, 57: PRINT "A(i)"
CIRCLE (320, 20), 2, 4: PAINT (320, 20), 4, 4
CIRCLE (320, 300), 182, 4, , , .2        '*** искомая параллель ***
CIRCLE (477, 318), 2, 4: PAINT (477, 318), 4, 4
                                  '*** произвольная точка A(i) ***
SLEEP 2
REM *** движение вниз от N ***
FOR t = 4.71 TO 6.28 STEP .03
  x = 50 * COS(t) + 320 : y = 2.3 * 50 * SIN(t) + 135 : PSET (x, y), 2
  FOR i = 1 TO 50000: NEXT
NEXT
REM *** движение по параллели ***
FOR t = .28 TO 2.9 STEP .07
  x = 50 * COS(t) + 320: y = .3 * 50 * SIN(t) + 132 : PSET (x, y), 2
  FOR i = 1 TO 50000: NEXT
NEXT
REM *** движение вверх к N ***
FOR t = 3.14 TO 4.57 STEP .03
  x = 50 * COS(t) + 320 : y = 2.3 * 50 * SIN(t) + 135 : PSET (x, y), 2
  FOR i = 1 TO 50000: NEXT
```

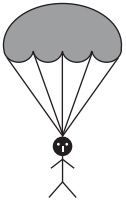
```
NEXT
REM *** движение вниз ***
FOR t = .51 TO .81 STEP .03
  x = 180 * COS(t) + 320 : y = 200 * SIN(t) + 220 : PSET (x, y), 2
  FOR i = 1 TO 50000: NEXT
NEXT
REM *** движение по параллели ***
FOR t = .62 TO 6.91 STEP .05
  x = 150 * COS(t) + 320: y = .2 * 150 * SIN(t) + 350 : PSET (x, y), 2
  FOR i = 1 TO 50000: NEXT
NEXT
REM *** движение вниз ***
FOR t = .81 TO .51 STEP -.01
  x = 180 * COS(t) + 320 : y = 200 * SIN(t) + 220 : PSET (x, y), 2
  FOR i = 1 TO 50000: NEXT
NEXT
```

41. Парашют

Комментарий. Данная задача решается довольно просто, если участник олимпиады умеет рисовать на экране фрагменты эллипса.

```
SCREEN 12: PRINT " Парашют. Автор – Пономаренко А. С."
x = 300: SLEEP 1
FOR y = 40 TO 500
  CIRCLE (x, y), 40, 7, 0, 3.14, .8        'верхний купол
  CIRCLE (x - 30, y), 10, 7, 3.14, 0, .8   ' **************
  CIRCLE (x - 10, y), 10, 7, 3.14, 0, .8   ' нижний
  CIRCLE (x + 10, y), 10, 7, 3.14, 0, .8   ' купол
  CIRCLE (x + 30, y), 10, 7, 3.14, 0, .8   ' **************
  LINE (x, y)-(x, y + 60), 7
  LINE (x - 40, y)-(x, y + 60), 7          'стропа
  LINE (x - 20, y)-(x, y + 60), 7          'стропа
  LINE (x + 20, y)-(x, y + 60), 7          'стропа
  LINE (x + 40, y)-(x, y + 60), 7          'стропа
  CIRCLE (x, y + 66), 6, 7                 'голова
  LINE (x, y + 73)-(x, y + 90), 7
  LINE (x, y + 75)-(x - 7, y + 80), 7
  LINE (x, y + 75)-(x + 7, y +   80), 7
  LINE (x, y + 90)-(x - 8, y + 100), 7
  LINE (x, y + 90)-(x + 8, y   + 100), 7
  PAINT (x - 30, y + 5), 6, 7
  PAINT (x, y - 19), 6, 7
  PAINT (x, y + 66), 7, 7: PSET (x - 3, y + 64), 1  'левый глаз
  PSET (x + 3, y + 64), 1                           'правый глаз
  LINE (x, y + 67)-(x, y + 69), 1                   'нос
  FOR i = 1 TO 12000: NEXT 'временная задержка
  IF INKEY$ = " " THEN END
 '*** рисование цветом фона ***
  CIRCLE (x, y), 40, 0, 0, 3.14, .8                 'верхний купол
```

```
  CIRCLE (x - 30, y), 10, 0, 3.14, 0, .8  ' **************
  CIRCLE (x - 10, y), 10, 0, 3.14, 0, .8  ' нижний
  CIRCLE (x + 10, y), 10, 0, 3.14, 0, .8  ' купол
  CIRCLE (x + 30, y), 10, 0, 3.14, 0, .8  ' **************
  LINE (x, y)-(x, y + 60), 0
  LINE (x - 40, y)-(x, y + 60), 0          'стропа
  LINE (x - 20, y)-(x, y + 60), 0          'стропа
  LINE (x + 20, y)-(x, y + 60), 0          'стропа
  LINE (x + 40, y)-(x, y + 60), 0          'стропа
  CIRCLE (x, y + 66), 6, 0                 'голова
 LINE (x, y + 73)-(x, y + 90), 0: LINE (x, y + 75)-(x - 7, y + 80), 0
  LINE (x, y + 75)-(x + 7, y + 80), 0
  LINE (x, y + 90)-(x - 8, y + 100), 0
  LINE (x, y + 90)-(x + 8, y + 100), 0
  PAINT (x - 30, y + 5), 0, 0: PAINT (x, y - 19), 0, 0
  PAINT (x, y + 66), 0, 0: PSET (x - 3, y + 64), 0 'левый глаз
  PSET (x + 3, y + 64), 0 'правый глаз
  LINE (x, y + 67)-(x, y + 69), 0 'нос
  IF x < 598 AND y < 200 THEN x = x + 2 ELSE x = x - 2
NEXT
```



Фрагмент работы программы: см. рис. справа

42. Уравнение Пелля

Результат работы программы:
решение уравнения Пелля $x^2 - 5y^2 = 1$ в натуральных числах:

n	x	y
1	9	4
2	161	72
3	2889	1292
4	51841	23184
5	930249	416020

Комментарий. Это типичная задача на перебор. Уже в процессе отладки программы можно установить верхнюю границу в цикле по y равной 420 000.

```
SCREEN 9: COLOR 14, 9
a$ = "Решаем уравнение x*x-5*y*y=1 в натуральных числах."
PRINT TAB(6); a$: PRINT ;
PRINT TAB(6); STRING$(57, "=");
PRINT TAB(14); "n"; TAB(34); "x"; TAB(54); "y"
PRINT TAB(6); STRING$(57, "=");
FOR y = 4 TO 420000
x = SQR(1 + 5 * y * y): x1 = INT(x): x2 = x - x1
IF x2 <> 0 THEN 1
r = x * x - 5 * y * y
IF r <> 1 THEN 1
```

```
n = n + 1
PRINT USING "#"; TAB(14); n;
PRINT USING "######"; TAB(29); x; TAB(49); y
1 NEXT
```

43. Числа, кратные трем

Контрольные примеры:

1) 51 126 225 141 66 432 99 1458 702 351 153;

2) 96 945 918 1242 81 513 153.

Решение:

```
CLS : PRINT "Числа, кратные трём"
INPUT "Введите натуральное число A, кратное 3, A меньше, чем 100"; a$
COLOR 14: PRINT : PRINT a$ + " ";
1 a = LEN(a$)
FOR i = 1 TO a
  a$(i) = MID$(a$, i, 1)
  b(i) = VAL(a$(i))
  b = b + b(i) ^ 3
NEXT
PRINT b;
a$ = STR$(b)
IF b = 153 THEN END
b = 0: SLEEP 1: GOTO 1
```

44. Задача о коровах

Ответ:

Количество коров (*n*)	270	110	70	30	20
Количество дней (*t*)	6	15	24	60	96

Решение:

```
CLS : PRINT "Задача о коровах"
FOR t = 1 TO 120
  n = (4800 + 10 * t) / (3 * t)
  n1 = INT(n): n2 = n - n1
  IF n2 = 0 THEN
    PRINT "t = "; t, "n = "; n
  END IF
NEXT
```

45. Температурные шкалы

Контрольные примеры:

1) 100 °C = 212 °F = 373 K = 80 °R;

2) 100 °F = 38 °C = 311 K = 30 °R;

3) 100 K = –173 °C = –270 °F = –138 °R;

4) 100 °R = 125 °C = 257 °F = 398 K.

Комментарий. В предложенной ниже в качестве решения программе отсутствует блок защиты от неправильного ввода данных. Однако с этим легко справиться, так как $t\,\mathrm{K} \geqslant 0$. Кроме того, в программе есть (а в условии задачи отсутствует!) блок перевода температуры тела в теплоту, которую необходимо затратить для нагрева этого тела от абсолютного нуля (0 K) до заданной величины.

```
SCREEN 9: COLOR 14, 1: LOCATE 1, 22
PRINT "СВЯЗЬ МЕЖДУ ТЕМПЕРАТУРНЫМИ ШКАЛАМИ:": LOCATE 2, 15
PRINT "Цельсий (C), Фаренгейт (F), Кельвин (K), Реомюр (R)."
PRINT
a$ = STRING$(77, "="): PRINT a$;
PRINT "|       C       |       F       |";
PRINT "|       K       |       R       |": PRINT a$;
PRINT "|               |               |";
PRINT "|               |               |": PRINT a$
LOCATE 10, 1
1 INPUT "В какой шкале вы будете вводить значение температуры
        (C,F,K или R)"; t$
INPUT "Введите значение температуры по выбранной вами шкале"; t
IF t$ = "c" OR t$ = "C" THEN 2
IF t$ = "f" OR t$ = "F" THEN 3
IF t$ = "k" OR t$ = "K" THEN 4
IF t$ = "r" OR t$ = "R" THEN 5 ELSE 1
2 tc = t: tf = CINT(1.8 * tc + 32): tk = tc + 273
  tr = CINT(.8 * tc): GOTO 6
3 tf = t: tc = CINT(5 * (tf - 32) / 9): tr = CINT(4 * (tf - 32) / 9)
  tk = CINT((5 * tf + 2297) / 9): GOTO 6
4 tk = t: tc = tk - 273: tf = CINT(1.8 * tk - 459.4)
  tr = CINT(.8 * tc): GOTO 6
5 tr = t: tc = CINT(1.25 * tr): tk = tc + 273: tf = CINT(2.25 * tr + 32)
6 LOCATE 7, 8: PRINT tc
  LOCATE 7, 27: PRINT tf
  LOCATE 7, 46: PRINT tk
  LOCATE 7, 65: PRINT tr
  LOCATE 13, 1: td = 2.07 * tk
  PRINT "Кроме того, все значения этих температур можно выразить
        в джоулях:"
  LOCATE 14, 25: PRINT td; "* 10^(-23)"
```

Результат работы программы:

```
Связь между температурными шкалами:
Цельсий (C), Фаренгейт (F), Кельвин (K), Реомюр (R)
```

C	F	K	R
125	257	398	100

```
В какой шкале вы будете вводить значение температуры
(C, F, K или R)? R
Введите значение температуры по выбранной вами шкале? 100
Кроме того, все значения этих температур можно выразить в джоулях:
                    823.86 * 10^(-23)
```

■ ОКРУЖНЫЕ (ГОРОДСКИЕ) ОЛИМПИАДЫ. МОСКОВСКАЯ ГОРОДСКАЯ ОЛИМПИАДА ПО ПРОГРАММИРОВАНИЮ

1. Дороги

Поскольку кратчайшим является один из четырех возможных путей: *ACB*, *ADB*, *ACDB* и *ABCD*, то достаточно найти, какой из этих путей короче.

Решение на языке QBasic:

```
CLS : PRINT "Дороги"
INPUT "Каковы длины пяти дорог", l1, l2, l3, l4, l5
s1 = l1 + l2 '*** маршрут ACB ***
s2 = l4 + l3 '*** маршрут ADB ***
s3 = l1 + l3 + l5 '*** маршрут ACDB ***
s4 = l4 + l5 + l2 '*** маршрут ADCB ***
CLS : LOCATE 1, 1: PRINT l1; l2; l3; l4; l5
IF s1 <= s2 AND s1 <= s3 AND s1 <= s4 THEN
  LOCATE 1, 40: PRINT s1: LOCATE 2, 41: PRINT "ACB": END
END IF
IF s2 <= s1 AND s2 <= s3 AND s2 <= s4 THEN
  LOCATE 1, 40: PRINT s2: LOCATE 2, 41: PRINT "ADB": END
END IF
IF s3 <= s2 AND s3 <= s1 AND s3 <= s4 THEN
  LOCATE 1, 40: PRINT s3: LOCATE 2, 41: PRINT "ACDB": END
END IF
IF s4 <= s2 AND s4 <= s3 AND s4 <= s1 THEN
  LOCATE 1, 40: PRINT s4: LOCATE 2, 41: PRINT "ADCB": END
END IF
```

Решение на языке Pascal:

```
var
  l1, l2, l3, l4, l5: integer;
  s1, s2, s3, s4: integer;
begin
  writeln('Введите длины дорог');
  readln(l1, l2, l3, l4, l5);
  s1:=l1+l2;
  s2:=l4+l3;
```

```
  s3:=l1+l5+l3;
  s4:=l4+l5+l2;
  if (s1<=s2) and (s1<=s3) and (s1<=s4) then
    writeln(s1,#10,#13,'ACB')
  else if (s2<=s1) and (s2<=s3) and (s2<=s4) then
    writeln(s2,#10,#13,'ADB')
  else if (s3<=s1) and (s3<=s2) and (s3<=s4) then
    writeln(s3,#10,#13,'ACDB')
  else
    writeln(s4,#10,#13,'ADCB');
end.
```

Критерии оценивания

За каждый пройденный тест из четырех начисляется 1 балл. Если пройдены все тесты, то дополнительно начисляется еще 1 балл (итого — 5 баллов).

Тесты для проверки

Входные данные	Выходные данные
10 10 10 20 20	20 ACB
100 100 99 100 100	199 ADB
20 50 10 50 10	40 ACDB
100 10 100 10 10	30 ADCB

2. «Ширли» и «Мырли»

Задача решается простым перебором возможных значений цен автомобилей.

Решение на языке QBasic:

```
CLS : REM *** "Ширли" и "Мырли" ***
FOR x = 1 TO 99 STEP 2
  FOR y = 1 TO 99 STEP 2
    IF (3 * x + y > 341) AND (6 * x > 7 * y) AND (5 * x < 6 * y) THEN
      k = k + 1
      x(k) = x: y(k) = y
      REM *** "Ширли" стоит x(k) рублей, а "Мырли" – y(k) рублей ***
    END IF
  NEXT y
NEXT x
REM *** Задача имеет k решений ***
```

```
PRINT k
FOR i = 1 TO k
  PRINT x(i); " "; y(i)
NEXT
```

Решение на языке Pascal:

```
var
  i, j, x, y, n: byte;
  a, b: array[1..255] of byte;
begin
  n:=0;
  for i:=1 to 50 do
    for j:=1 to 50 do begin
      x:=2*i-1;
      y:=2*j-1;
      if (3*x+y>341) and (6*x>7*y) and (5*x<6*y) then begin
        n:=n+1;
        a[n]:=x; b[n]:=y;
      end;
    end;
  writeln(n);
  for i:=1 to n do
    writeln(a[i],' ',b[i]);
end.
```

Критерии оценивания

Оценка в баллах вычисляется как разность количеств выведенных на экран верных (максимум 7) и неверных решений. Например, если программа выводит на экран семь решений, но из них шесть верных, а одно неверное, то начисляется 6 – 1 = = 5 баллов.

Полный перечень решений задачи:

Выходные данные
7
89 75
91 77
93 79
95 81
97 81
97 83
99 83

3. Кузнечик

Ниже приводится два различных решения задачи. Первое (на Бейсике) основано на переборе возможных сочетаний количеств прыжков вперед и назад, при этом используется тот факт, что порядок следования таких прыжков значения не имеет.

Во втором решении (на Паскале) кузнечик прыгает по следующему алгоритму: сначала вперед, пока кузнечик не окажется дальше точки назначения, а затем назад с помощью нескольких комбинаций прыжков FB (на одно деление назад).

Решение на языке QBasic:

```
CLS : PRINT "Кузнечик"
PRINT "Кузнечик сидит в точке 0 на оси абсцисс и может прыгать по ней"
PRINT "на 6 делений вправо или на 7 делений влево. Ему надо попасть в"
PRINT "точку N, где N – целое число из диапазона от –10 до 10."
INPUT "Какое ваше N"; n
CLS : PRINT n; TAB(20);
IF n = 0 THEN
  ' "Зачем мне прыгать? Я и так сижу в 0."
  PRINT n: END
END IF
FOR k = 0 TO 7
  FOR r = 0 TO 6
    x = 6 * k - 7 * r
    IF x = n THEN
      PRINT STRING$(k, "F"); STRING$(r, "B"); "";
      PRINT TAB(19); r + k
      END
    END IF
  NEXT
NEXT
```

Решение на языке Pascal:

```
var
  n, m, i, j: shortint;
begin
  write('Введите координату точки назначения: ');
  readln(n);
  if n=0 then
    writeln(0)
  else begin
    m:=0;
    if n>0 then m:=n div 6 + 1;
    for i:=1 to m do
      write('F');
    for i:=1 to m*6-n do
      write('FB');
  end;
  writeln;
end.
```

Критерии оценивания

Ответы, выдаваемые программой участника, могут не совпадать с приведенными ниже, поэтому правильность решения проверяется вручную. Так, если ответ участника отличается от приведенного ниже только порядком расположения обозначений прыжков, то такой ответ тоже является правильным.

За каждый пройденный тест начисляется 2 балла. Если пройдены все тесты, то дополнительно начисляется еще 1 балл (итого — 9 баллов).

Тесты для проверки

Входные данные	Выходные данные
3	FFBFBFB 7
10	FFFBFB 6
−3	FBFBFB 6
0	0

4. Отрезки

Решение на языке QBasic:

```
1 CLS : PRINT "Минимальная сумма расстояний"
'INPUT "Введите координаты точки A(a;b), где -16<a<16 и -12<b<11"; a, b
'INPUT "Введите координаты точки B(m;n), где -16<m<16 и -12<n<11"; m, n
'a = 4: b = 5: m = -6: n = -4
'a = -6: b = 4: m = 4: n = -1
'a = -4: b = 4: m = 0: n = -3
'a = -3: b = 5: m = -3: n = -4
a = 6: b = -1: m = 4: n = 4
SCREEN 12
REM ****** построение осей ******
LINE (0, 220)-(640, 220), 14: LINE (320, 0)-(320, 460), 14
LINE (630, 215)-(640, 220), 14: LINE (630, 225)-(640, 220), 14
LINE (315, 10)-(320, 0), 14: LINE (325, 10)-(320, 0), 14
LOCATE 1, 39: PRINT "Y": LOCATE 15, 80: PRINT "X"
LOCATE 15, 39: PRINT "0"
REM ****** разметка осей ******
FOR x = 0 TO 620 STEP 20
  LINE (x, 218)-(x, 222)
NEXT
FOR y = 20 TO 440 STEP 20
  LINE (318, y)-(322, y)
NEXT
REM ****** построение точек A и B ******
xa = 20 * a + 320: ya = 220 - 20 * b
```

```
xb = 20 * m + 320: yb = 220 - 20 * n
CIRCLE (xa, ya), 2, 1: PAINT (xa, ya), 1, 1
CIRCLE (xb, yb), 2, 1: PAINT (xb, yb), 1, 1
REM ****** расчет ординаты точки C ******
IF a * m > 0 THEN
  yc = b + a * (n - b) / (m + a)
ELSE
  yc = b - a * (n - b) / (m - a)
END IF
REM ****** построение точки C и отрезков AC и BC ******
PRINT "A ("; a; ";"; b; ")"
PRINT "B ("; m; ";"; n; ")"
PRINT "C (0 ;"; yc; ")"
c = 220 - 20 * yc
CIRCLE (320, c), 2, 1: PAINT (320, c), 1, 1
LINE (320, c)-(xa, ya), 15: LINE (320, c)-(xb, yb), 15
DIM BOX%(1 TO 576)
GET (0, 235)-(7, 252), BOX%
PUT (xa - 4, ya - 22), BOX%
GET (0, 253)-(7, 270), BOX%
PUT (xb - 4, yb - 22), BOX%
GET (0, 271)-(7, 288), BOX%
PUT (306, c - 9), BOX%
```

Решение на языке Pascal:

```
uses graph;
var
  d, r, i, xa, xb, ya, yb, yc: integer;
  a, b, m, n, c: real;
  s: string;
begin
  writeln('Введите координаты точек A(a,b) и B(m,n),
         где -16<a, m<16, -12<b, n<12:');
  readln(a,b,m,n);
  d:=0;
  initgraph(d,r,'c:\progs\bprus');
  line(0,240,640,240);      line(320,0,320,480);
  line(630,235,640,240);    line(630,245,640,240);
  line(315,10,320,0);       line(325,10,320,0);
  outtextxy(330,10,'Y'); outtextxy(250,620,'X'); outtextxy(330,230,'0');
  for i:=1 to 31 do begin
    line(i*20,238,i*20,242);
    str(i-16,s);
    outtextxy(i*20-4,245,s);
  end;
  for i:=1 to 23 do begin
    line(318,i*20,322,i*20);
    str(12-i,s);
    outtextxy(295,i*20-4,s);
  end;
  xa:=trunc(20*a+320); ya:=trunc(240-20*b);
```

```
    xb:=trunc(20*m+320); yb:=trunc(240-20*n);
    circle(xa,ya,1); circle(xb,yb,1);
    if a*m>0 then c:=b+a*(n-b)/(m+a)
    else c:=b-a*(n-b)/(m-a);
    str(c:0:2,s);
    outtextxy(10,300,'C(0;'+s+')');
    yc:=trunc(240-20*c);
    circle(320,yc,1);
    line(320,yc,xa,ya); line(320,yc,xb,yb);
    outtextxy(xa-4,ya-10,'A'); outtextxy(xb-4,yb-10,'B');
    outtextxy(325,yc-8,'C');
    readln;
    closegraph;
end.
```

Примечания. Ограничения на координаты, которые должна прежде всего выводить программа, зависят от графического режима. Ограничения, приведенные ниже, рассчитаны на разрешение 640 × 480; если установлен другой режим, то ограничения будут пропорционально отличаться от указанных.

Если крайние значения, которые в приведенных ниже тестах не включены в диапазон, у участника в него включены, то это не считается ошибкой, если при этом точки с граничными координатами попадают на экран (они должны быть на границе экрана); это необходимо проверить отдельным тестом. Если же хотя бы некоторые граничные точки попадают в диапазон, но не видны на экране, то это наказывается одним штрафным баллом.

Критерии оценивания

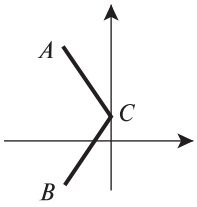
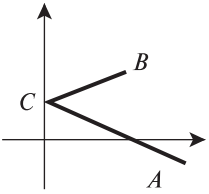
Если правильно указаны ограничения и построены оси, то начисляется 3 балла.

За каждый пройденный тест начисляется еще по 3 балла.

За неверное указание ограничений снимается от 1 до 3 штрафных баллов: 1 балл, если ошибки только в строгости неравенства (см. примечания выше), и 2—3 балла, если ошибки более грубые (неверны 1—2 неравенства — 2 балла, 3—4 неравенства — 3 балла).

Если на осях не отмечены целочисленные точки, то снимается 2 штрафных балла, если эти точки не подписаны — еще 2 штрафных балла. За прочие погрешности в изображении осей (нет стрелочек и/или подписей осей) снимается 1 штрафной балл.

Тесты для проверки

Входные данные	Выходные данные
Ограничения, сразу выводимые программой	$-16 < a, m < 16$ $-12 < b, n < 12$
–6 4 4 –1	0 1 Точка *C* находится на отрезке *AB*
–4 4 0 –3	0 –3 Точка *C* совпадает с точкой *B*
–3 5 –3 –4	0 0.5 Примерное положение отрезков: 
6 –1 4 4	0 2 Примерное положение отрезков: 

5. Рикошет

В начале движения (до удара о первую стенку) на каждом шаге к абсциссе движущейся точки добавляется величина $c_x \cdot \cos(\varphi)$, а к ординате — $c_y \cdot \sin(\varphi)$, где c_x, c_y — константы, влияющие на скорость и направление движения (равные по модулю).

На каждом шаге проверяется, не ударилась ли точка о стенку, и если ударилась, то соответствующая константа — c_x или c_y — меняет знак.

Решение на языке QBasic:

```
SCREEN 12
CONST PI = 3.141592653589#
LINE (100, 100)-(300, 300), , B
INPUT "Введите угол (в градусах): ", A
A = A / 180 * PI
X = 200: Y = 200
```

```
CX = .1: CY = -.1
DO
  PSET (INT(X), INT(Y)), 0
  IF (X >= 299) OR (X <= 102) THEN CX = -CX
  IF (Y >= 299) OR (Y <= 102) THEN CY = -CY
  X = X + CX * COS(A)
  Y = Y + CY * SIN(A)
  PSET (INT(X), INT(Y)), 15
  FOR I = 1 TO 3000: NEXT I
LOOP WHILE INKEY$ = " "
```

Решение на языке Pascal:

```
uses graph, crt;
var
  d, r, i: integer;
  a, x, y, cx, cy: real;
begin
  write('Введите угол (в градусах): ');
  readln(a);
  a:=a/180*pi;
  d:=0;
  initgraph(d,r,'c:\progs\bprus');
  rectangle(100,100,300,300);
  readkey;
  x:=200; y:=200;
  cx:=0.1; cy:=-0.1;
  repeat
    putpixel(trunc(x),trunc(y),0);
    if (x>=299) or (x<=102) then cx:=-cx;
    if (y>=299) or (y<=102) then cy:=-cy;
    x:=x+cx*cos(a);
    y:=y+cy*sin(a);
    putpixel(trunc(x),trunc(y),15);
    delay(10);
  until keypressed;
  closegraph;
end.
```

Критерии оценивания

Скорость движения точки может быть выбрана участником произвольно из указанного в условии диапазона (от 70 до 200 пикселей в секунду). Соблюдение этого ограничения проверяется визуально: от одного края экрана до противоположного по кратчайшему пути точка должна пролетать за 1—3 с (что легко увидеть по первому тесту).

Первый тест проверяется до трех ударов о стены, второй и третий — до тех пор, пока точка не ударится о каждую стену.

Каждый тест оценивается в 6 баллов.

Тесты для проверки

Входные данные	Выходные данные
180	Точка начинает лететь влево
27	Тангенс этого угла примерно равен 0,5, поэтому траектория — почти «восьмерка», состоящая из двух ромбов
315	Точка должна лететь в правый нижний угол. Проверяется, полетит ли она из него в противоположный угол

6. Степень

Данная задача требует предварительного анализа. Она сложна не столько в реализации, сколько, как говорится, «идейно».

Во-первых, если число A разложить на делители (например: $p_1^{k1} \cdot p_2^{k2} \cdot \ldots p_q^{kq}$) и взять произведение этих множителей (причем каждый множитель брать в первой степени: $p_1 \cdot p_2 \cdot \ldots \cdot p_q$), то ответ почти всегда будет равен этому числу. Исключениями являются случаи, когда у числа A мало простых делителей, — тогда ответ может быть равен удвоенному, утроенному и т. д. исходному числу.

Во-вторых, нужно отметить, что поиск нужного числа подбором дает правильный ответ только для малых чисел A, потому что для больших чисел значение N^N превысит допустимый предел значений для типа данных «длинное целое». (Для всех других алгоритмов также нужно следить, чтобы никакие результаты промежуточных вычислений не превышали этот предел.)

В приведенных ниже программах разложения числа A его делители хранятся в специально предусмотренных массивах. В программе на Бейсике для каждого разложения используются два массива (простые числа и степени) и целое число (количество множителей), а в программе на Паскале эти данные объединены в запись `TFactoring`.

Чтобы не возникало превышения допустимого предела значений длинных целых чисел, произведения больших чисел рассчитываются как произведения разложений на делители.

Решение на языке QBasic:

```
DECLARE SUB FACTOR (NUMBER&, K)
DECLARE SUB MULFACT (A, B, C)
DECLARE FUNCTION MIN& (A&, B&)
CONST MaxP = 40
```

```
DIM SHARED P(0 TO 4, MaxP) AS LONG
DIM SHARED D(0 TO 4, MaxP) AS LONG
DIM SHARED NP(0 TO 4) AS INTEGER
DIM A AS LONG, N AS LONG, I AS LONG, SMALL AS LONG
INPUT A
CALL FACTOR(A, 0)
N = 1
FOR I = 1 TO NP(0)
  N = N * P(0, I)
NEXT
CALL FACTOR(N, 1)
SMALL = 1
DO
  CALL FACTOR(SMALL, 2)
  CALL MULFACT(1, 2, 3)
  FOR I = 1 TO NP(3)
    D(3, I) = D(3, I) * (-N * SMALL)
  NEXT
  CALL MULFACT(3, 0, 4)
  OK = 1
  FOR I = 1 TO NP(4)
    IF D(4, I) > 0 THEN OK = 0
  NEXT
  IF OK = 1 THEN EXIT DO
  SMALL = SMALL + 1
LOOP
PRINT N * SMALL
END

SUB FACTOR (NUMBER&, K)
  WHAT& = NUMBER&
  J = 2&
  NP(K) = 0
  WHILE J ^ 2 < WHAT& + 1
    IF WHAT& MOD J = 0 THEN
      NP(K) = NP(K) + 1
      P(K, NP(K)) = J
      D(K, NP(K)) = 0
      WHILE WHAT& MOD J = 0
        WHAT& = WHAT& \ J
        D(K, NP(K)) = D(K, NP(K)) + 1
      WEND
    END IF
    J = J + 1
  WEND
  IF WHAT& > 1 THEN
    NP(K) = NP(K) + 1
    P(K, NP(K)) = WHAT&
    D(K, NP(K)) = 1
  END IF
END SUB
```

```
FUNCTION MIN& (A&, B&) STATIC
  IF A& < B& THEN MIN = A& ELSE MIN = B&
END FUNCTION

SUB MULFACT (A, B, C)
  IA = 1&: IB = 1&: NP(C) = 0
  WHILE (IA <= NP(A)) OR (IB <= NP(B))
    IF IA > NP(A) THEN
      NP(C) = NP(C) + 1
      P(C, NP(C)) = P(B, IB)
      D(C, NP(C)) = D(B, IB)
      IB = IB + 1
    ELSEIF IB > NP(B) THEN
      NP(C) = NP(C) + 1
      P(C, NP(C)) = P(A, IA)
      D(C, NP(C)) = D(A, IA)
      IA = IA + 1
    ELSE
       NP(C) = NP(C) + 1
       P(C, NP(C)) = MIN(P(A, IA), P(B, IB))
       D(C, NP(C)) = 0
       IF P(C, NP(C)) = P(A, IA) THEN
           D(C, NP(C)) = D(C, NP(C)) + D(A, IA)
           IA = IA + 1
         END IF
         IF P(C, NP(C)) = P(B, IB) THEN
           D(C, NP(C)) = D(C, NP(C)) + D(B, IB)
           IB = IB + 1
         END IF
    END IF
  WEND
END SUB
```

Решение на языке Pascal:

```
Const
  MaxP=40;
Type
  TFactoring=Record
    NP:Integer;
    P:Array[1..MaxP] Of LongInt;
    D:Array[1..MaxP] Of LongInt;
  End;
Var
  I,A,N,Small:LongInt;
  F,F1,F2,F3,F4:TFactoring;
  Ok:Boolean;
Procedure Factor(What:LongInt; Var F:TFactoring);
Var
  I:LongInt;
Begin
  I:=2;
  F.NP:=0;
  While I*I<=What Do Begin
```

```
    If What Mod I=0 Then Begin
      Inc(F.NP);
      F.P[F.NP]:=I;
      F.D[F.NP]:=0;
      While What Mod I=0 Do Begin
        What:=What Div I;
        Inc(F.D[F.NP]);
      End;
    End;
    Inc(I);
  End;
  If What>1 Then Begin
    Inc(F.NP);
    F.P[F.NP]:=What;
    F.D[F.NP]:=1;
  End;
End;
Function Min(A,B:LongInt):LongInt;
Begin
  If A<B Then
    Min:=A
  Else
    Min:=B;
End;
Procedure MulFactorings(Const A,B:TFactoring; Var C:TFactoring);
Var
  IA,IB:LongInt;
Begin
IA:=1;
IB:=1;
C.NP:=0;
While (IA<=A.NP) Or (IB<=B.NP) Do Begin
  If IA>A.NP Then Begin
    Inc(C.NP);
    C.P[C.NP]:=B.P[IB];
    C.D[C.NP]:=B.D[IB];
    Inc(IB);
  End Else If IB>B.NP Then Begin
    Inc(C.NP);
    C.P[C.NP]:=A.P[IA];
    C.D[C.NP]:=A.D[IA];
    Inc(IA);
  End Else Begin
    Inc(C.NP);
    C.P[C.NP]:=Min(A.P[IA],B.P[IB]);
    C.D[C.NP]:=0;
    If C.P[C.NP]=A.P[IA] Then Begin
      Inc(C.D[C.NP],A.D[IA]);
      Inc(IA);
    End;
    If C.P[C.NP]=B.P[IB] Then Begin
      Inc(C.D[C.NP],B.D[IB]);
```

```
      Inc(IB);
    End;
  End;
 End;
End;
Begin
  Read(A);
  Factor(A,F);
  N:=1;
  For I:=1 To F.NP Do N:=N*F.P[I];
  Factor(N,F1);
  Small:=1;
  Repeat
    Factor(Small,F2);
    MulFactorings(F1,F2,F3);
    For I:=1 To F3.NP Do
      F3.D[I]:=F3.D[I]*(-N*Small);
    MulFactorings(F3,F,F4);
    Ok:=True;
    For I:=1 To F4.NP Do
      If F4.D[I]>0 Then Ok:=False;
    If Ok Then Break;
    Inc(Small);
  Until False;
  WriteLn(N*Small);
End.
```

Критерии оценивания

К данной задаче предусмотрено всего 10 тестов, каждый из которых оценивается в 3 балла.

Тесты для проверки

Входные данные	Выходные данные
1	1
4	2
12	6
1024	8
255255	255255
17631601	4199
387420489	9
625718308	312859154
872825381	872825381
805306368	24

■ ВСЕРОССИЙСКИЕ ОЛИМПИАДЫ. ВСЕРОССИЙСКАЯ ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

1. Последовательность

Прежде всего отметим, что данная последовательность бесконечна. Поэтому невозможно использовать для нее предварительно сформированный массив, не выполнив некоторых предварительных расчетов.

Данная задача может быть решена различными способами. Рассмотрим один из возможных вариантов ее решения чисто числовыми методами.

Приведем общее словесное описание алгоритма решения задачи.

Последовательно перебирая все натуральные числа, подсчитываем общее количество цифр в просмотренных числах (*S*), пока сумма цифр *S* не станет равной или большей заданного числа *N*. Для подсчета общего количества цифр необходимо определять количество цифр (разрядность *M*) в каждом текущем рассматриваемом числе *P*. Количество цифр в любом целом числе легко определить, воспользовавшись функцией выделения целой части от деления числа *P* на 10, причем делить надо до тех пор, пока целая часть не станет равной 0. Так как действия по определению разрядности числа многократно повторяются, а их результатом является целое число, то эту часть программы целесообразно выполнить в виде подпрограммы-функции.

Решение на языке BASIC:

```
Function razr(a) As Integer
  Dim b, c As Integer
  c = 0
  b = a     'переменной b присваиваем значение числа a,
            'разряд которого определяем
  Do While b > 0            'пока b>0 выполняем
    b = Int(b / 10)         'в b записываем целую часть от деления
                            'предыдущего значения на 10
    c = c + 1               'подсчитываем количество разрядов
  Loop
  razr = c
End Function
```

Для подсчета количества цифр *S* в части последовательности при добавлении очередного *M*-разрядного числа *P* воспользуемся следующим фрагментом программного кода:

```
P=1         'первое число в последовательности – это 1
S=1         'сумма цифр в последовательности,
            'состоящей из одного числа "1"
M=1         'первое число в последовательности – одноразрядное
Do While S<N        'считаем в цикле, пока S не превышает N
  P=P+1             'увеличиваем значение текущего числа на 1
  M=razr(P)         'определяем разрядность M числа P
  S=S+M             'увеличиваем сумму цифр последовательности на M
Loop
```

После выполнения данного фрагмента мы получим последнее рассмотренное число последовательности P и количество разрядов в нем M.

Если количество цифр S в точности совпадает с заданным числом N, то искомая цифра составит количество единиц в последнем рассмотренном числе P. Получить это значение очень просто — как остаток от деления числа P на 10.

Если $S > N$, то для решения поставленной задачи сначала необходимо определить, какому разряду числа P соответствует искомая цифра. Это тоже можно сделать разными способами. Если продолжать использовать числовые методы, то можно действовать, например, так.

Воспользуемся тем, что любое M-разрядное десятичное число A может быть записано в виде суммы:

$$A = a_0 \cdot 10^0 + a_1 \cdot 10^1 + \ldots + a_M \cdot 10^{M-1}. \quad (1)$$

Можно заметить, что величина $K = (S - N)$ дает нам информацию как раз о разряде числа P, значение которого мы ищем: при $K = 0$ это единицы, $K = 1$ — десятки, $K = 2$ — сотни и т. д. — сравните эти значения с показателями степеней в выражении (1). Тогда получить значение искомой цифры оказывается достаточно просто — надо K раз выполнить целочисленное деление числа P на 10, а затем от конечного результата получить остаток от деления на 10. Например, если число P — однозначное, то $K = 0$. Тогда делить число P на 10 не требуется, а остаток от деления P на 10 дает искомый результат. При двузначном P получим, что $K = 1$: один раз целочисленно делим P на 10, а результат еще раз делим на 10 для получения остатка.

Таким образом, задача принципиально решена.

Заметим также, что случай с $K = 0$ полностью совпадает с предыдущим рассмотрением при $S = N$, т. е. вариант при $S = N$ является частным случаем последнего варианта рассмотрения и может отдельно не выделяться.

Текст этого фрагмента программы на языке BASIC:

```
K = S - N           'искомый разряд числа P
D = P               'вспомогательная переменная для запоминания
                    'целой части от деления на 10
For i = 1 To K
  D = Int(D / 10)          'получение целой части от деления на 10
Next i
Rez = D Mod 10              'получение остатка от деления на 10 –
                            'искомая цифра
Print Rez
```

Последние два оператора можно объединить:

```
Print D Mod 10
```

Полный текст программы на языке BASIC:

```
Function razr(a) As Integer
'описание функции получения разрядности целого числа
Dim b, c As Integer
c = 0
b = a
Do While b > 0
  b = Int(b / 10)
  c = c + 1
Loop
razr = c
End Function
Dim N, S, P, M, D, Rez, i As Integer
Print "Введите номер искомой цифры"
Input N
P = 1
S = 1
M = 1
Do While S < N
  P = P + 1
  M = razr(P)
  S = S + I
Loop
K = S - N
D = P
For i = 1 To K
  D = Int(D / 10)
Next i
Rez = D Mod 10
Print Rez  'печать результата
```

2. Схема

Эта задача решается с использованием понятий алгебры логики, изучаемой в рамках предмета «Информатика и ИКТ». Тем самым школьник, с одной стороны, использует знания, полученные в школе (или самостоятельно), а с другой стороны, осваивает логические основы цифровой схемотехники.

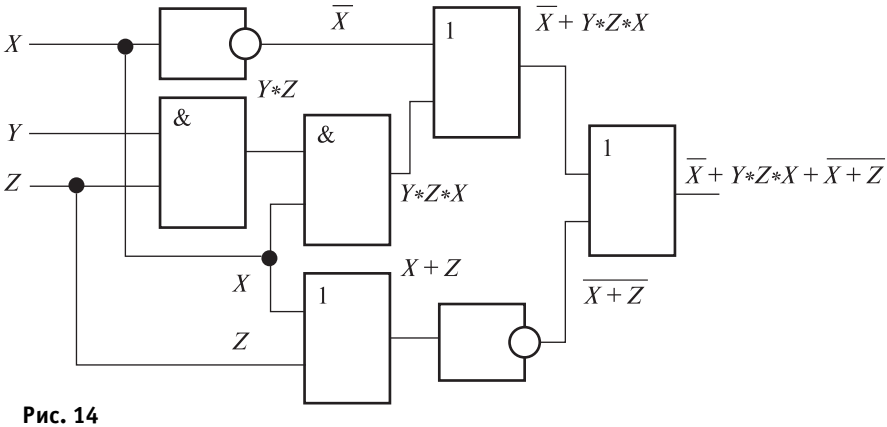


Рис. 14

Задачу можно решать различными способами — как путем построения таблиц истинности, так и с применением аппарата булевой алгебры. Рассмотрим второй из этих способов.

Прежде всего, обозначим на рисунке все логические функции, получаемые на выходе каждого из элементов (рис. 14).

В приведенных далее формулах знак «*» соответствует логической операции дизъюнкции (& — логическое умножение, функция «И»), а знак «+» — операции конъюнкции (1 — логическое сложение, функция «ИЛИ»); надчеркивание обозначает логическое отрицание.

Выходная логическая функция F записывается в виде:

$$F = \overline{X} + Y^* Z^* X + \overline{X+Z}\,.$$

Преобразуем это выражение в соответствии с законами алгебры логики.

{воспользуемся для третьего слагаемого законом де Моргана}

$$F = \overline{X} + Y^* Z^* X + \overline{X+Z} = \overline{X} + Y^* Z^* X + \overline{X} * \overline{Z} =$$

$$= \overline{X} + \overline{X} * \overline{Z} + Y^* Z^* X = \{\text{использован закон коммутации}\} =$$

$$= \overline{X} + Y^* Z^* X = \{\text{использован закон поглощения для первых двух слагаемых}\} = \overline{X^* \overline{Y^* Z^* X}} = \{\text{использован закон де Моргана}\} = \overline{X^* (\overline{Y^* Z} + \overline{X})} = \{\text{использован закон де Моргана}\} =$$

$$= \overline{X^* \overline{Y^* Z} + X^* \overline{X}} = \{\text{использован дистрибутивный закон}\} =$$

$$= \overline{X^* \overline{Y^* Z}} = \overline{X} + Y^* Z$$

{использованы закон противоречия и закон де Моргана}.

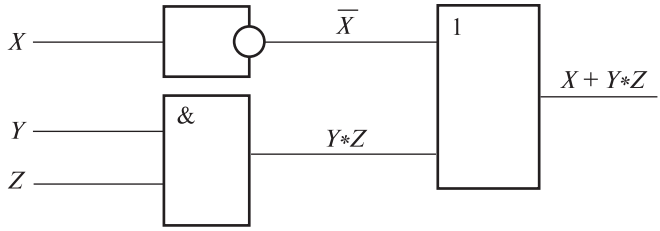


Рис. 15

Равносильность полученного результата первоначальному логическому выражению для функции F можно проверить, используя таблицы истинности.

Остается построить для полученной логической функции упрощенную схему из трех логических элементов (рис. 15).

3. Электрическая цепь

Подобная задача требует грамотного построения математической модели решения. Для построения такой модели необходимо знать (из курса физики) формулы для емкости последовательно и параллельно соединенных конденсаторов, а также принять верное решение о том, откуда начинать «сборку» цепи — слева направо или наоборот.

`Модель` для решения задачи «Электрическая цепь» мы будем строить, последовательно «собирая» данную электрическую цепь начиная с последнего звена N.

В последнем звене конденсаторы соединены последовательно. Для последовательного соединения k конденсаторов можно воспользоваться общей формулой:

$$\frac{1}{C} = \frac{1}{C_1} + \frac{1}{C_2} + ... + \frac{1}{C_k},$$

где C_1, C_2, ..., C_k — в общем случае разные величины емкостей последовательно соединенных конденсаторов.

Для двух последовательно соединенных конденсаторов в N-м звене общую емкость можно вычислить по формуле:

$$\frac{1}{C} = \frac{1}{C_1} + \frac{1}{C_2}. \qquad (1)$$

Отсюда

$$C = \frac{C_1 * C_2}{C_1 + C_2}. \qquad (2)$$

Так как для N-го звена $C_1 = C_2 = C$, то

$$C_0 = \frac{C}{2}. \tag{3}$$

Далее, начиная со звена $(N - 1)$, добавление параллельного конденсатора C дает изменение емкости

$$C_0 = C_0 + C. \tag{4}$$

Добавление к ним последовательного конденсатора C дает изменение емкости:

$$C_0 = \frac{C_0 * C}{C_0 + C}. \tag{5}$$

Так как в каждом звене, предшествующем N-му, выполняется именно порядок действий (4) и (5), то можно объединить их в одно математическое выражение. Тогда для любого из этих $(N - 1)$ звеньев получим:

$$C_0 = \frac{(C_0 + C) * C}{(C_0 + C) + C} = \frac{(C_0 + C) * C}{C_0 + 2C}. \tag{6}$$

Таким образом, достаточно сначала вычислить значение C_0 по формуле (3) для последнего звена, а затем рекурсивно вычислять (по формулам (4) и (5) или только по формуле (6)) общую емкость для оставшегося количества звеньев.

Так как для N-го звена и остальных звеньев вычисление производится по различным формулам, то при написании программы необходимо не забыть сделать начальную проверку количества звеньев (если $N = 1$, то расчет производится только по формуле (3), а по формулам (4) и (5) или по формуле (6) не выполняется). Необходимо также помнить и о том, что по условию задачи необходимо округлить результат до 3-го знака после запятой.

С учетом всего сказанного п р о г р а м м а на языке BASIC имеет вид:

```
Dim N, i As Integer
Dim C, C0 As Single
Input "Введите количество звеньев электрической цепи";N
Input "Введите значение емкости конденсатора C в мкФ";C
C0 = C/2
If N>1 Then
  For i=1 To N-1
    C0 = (C0+C)*C/(C0+2*C)
  Next i
End If
Rez=Int(C0*10000/10)/1000
            'округление результата до 3-го знака после запятой
Print Rez    'вывод результата на печать
```

Примечание для любознательных. Если запускать данную программу с одним и тем же значением емкости конденсатора C и увеличивающимся количеством звеньев цепи N, то можно заметить, что очень быстро результат перестанет менять свое значение. Подумайте, почему так получается.

4. Радуга

Для решения этой задачи также важно разработать модель ее решения. Одним из вариантов такой модели может быть использование только числовой обработки, тогда как символы используются только для вывода результатов на экран.

Для начала обратим внимание на несколько общих соображений.

Во-первых, так как цифры 1—8 фактически соответствуют номерам строк доски, то дополнительный столбец для хранения этих цифр не понадобится.

Во-вторых, так как столбцы шахматного поля имеют буквенные обозначения, а в каждом столбце располагаются кубики только одного цвета, то для буквенных обозначений номеров столбцов, букв, обозначающих цвета кубиков, и слов, обозначающих эти цвета, можно использовать дополнительный двумерный строковый массив $B(3, 8)$. Причем для определенности можно считать, что в 1-й строке будут содержаться буквенные обозначения столбцов шахматной доски, во второй строке — буквенные обозначения цветов, а в третьей — названия цветов.

Заполнение этой матрицы можно произвести вручную с клавиатуры, например так (текст программы на языке BASIC):

```
Print "Введите буквенные обозначения столбцов шахматной доски"
j=1
For i=1 To 8
  Print "Столбец»", i
  Input B(i,j)
Next i
Print "Введите буквенные обозначения цветов кубиков"
j=2
For i=1 To 8
  Print "Цвет", i
  Input B(i,j)
Next i
Print "Введите названия цветов"
j=3
For i=1 To 8
  Print "Цвет", i
  Input B(i,j)
Next i
```

Можно также ввести заполнение этой матрицы непосредственно в текст программы (см. далее).

Для заполнения шахматного поля цветными кубиками достаточно использовать квадратную матрицу $A(8, 8)$. Так как цвет кубиков является фактически «свойством» столбца и мы его «отделили» от шахматного поля, в качестве элементов матрицы A достаточно использовать два числовых значения: 1 — кубик есть, 0 — кубика нет.

Сама модель решения поставленной задачи состоит из двух частей:

1) заполнение шахматного поля кубиками в случайных клетках;

2) определение отсутствующих цветов.

Рассмотрим эти части более подробно.

Заполнение шахматного поля кубиками. По условию задачи на каждой линии может быть от 1 до 3 кубиков. Поэтому данный этап должен быть разбит на три подзадачи.

1) Заполнение всей матрицы значениями «0»:

```
For i=1 To 8
  For j=1 To 8
    A(i,j)=0
  Next j
Next i
```

2) Определение случайным образом количества кубиков в столбце от 1 до 3. Произвольный выбор заданного количества кубиков реализуется при помощи генератора псевдослучайных чисел `Rnd`. Так как значение, возвращаемое функцией `Rnd`, находится в пределах от 0 до 1, а количество кубиков является целым числом, для количества кубиков n в текущем столбце необходимо использовать следующую конструкцию:

```
n=Int(1+3*Rnd)
```

3) Определение случайного местоположения каждого из кубиков в столбце (от 1 до 8). Сделать это несколько сложнее, так как необходимо предусмотреть попадание кубиков в *различные* клетки столбца. Для этого можно, например, использовать вспомогательный одномерный массив $V(2)$, который перед распределением кубиков по клеткам столбца заполняется нулями, а затем — значениями номеров строк столбца, в клетки которых предполагается поместить кубик. Тогда при генерировании второго и третьего (если необходимо) положения кубика в столбце (т. е. номера строки) необходимо сравнивать его с предыдущими

сгенерированными для этого столбца значениями и при совпадении генерацию номера строки повторить.

Фрагмент этой части программы:

```
For i = 1 To 8                  'обнуление массива "кубиков"
  For j = 1 To 8
    A(i, j) = 0
  Next j
Next i
For i = 1 To 8                  'цикл выбора номера столбца матрицы
  Randomize                     'инициализация генератора случайных
                                'чисел Rnd
  n = Int(1 + 3 * Rnd)          'генерация количества кубиков
                                'в данном столбце
  For j = 1 To 2                'обнуление вспомогательной матрицы
    V(j) = 0
  Next j
  Randomize
  q = Int(1 + 8 * Rnd)          'генерация позиции первого кубика
                                'в текущем столбце
  V(1) = q
  A(q, i) = 1                   'запоминаем позицию первого кубика
                                'в столбце
If n > 1 Then                   'если для столбца сгенерировано
                                'более одного кубика, то
  j = 2
  Do                            'цикл, ограничивающий генерацию позиции
                                'для 2-го (и 3-го) кубика
    Do                          'цикл,
      Do                        'цикл повторной генерации позиции
                                'кубика
        Randomize
        k = Int(1 + 8 * Rnd)    'генерация новой позиции
                                'кубика в столбце
        p = 1
      Loop While V(p) = k       'если новая позиция кубика совпадает
                                'с позицией первого, то генерацию надо
                                'повторить
      p = p + 1
      If p > j - 1 Then Exit Do 'условие прекращения генерации
                                'новой позиции
    Loop While V(p) = k         'если новая позиция кубика совпадает
                                'с позицией второго, то генерацию
                                'надо повторить
    V(j) = k                    'запомнить очередную позицию кубика
                                'в столбце
    A(k, i) = 1                 'запомнить положение очередного
                                'кубика в матрице
    j = j + 1
  Loop While j < n              'если позиции всех кубиков сгенери-
                                'рованы, то выйти
  End If
Next i
```

Печать изображения расположения кубиков. Для этого используем буквенные значения столбцов шахматной доски и буквенные значения цветов из массива *B*(3, 8), а также числовые значения массива *A*(8, 8). Для корректной печати результатов необходимо предусмотреть, что первый из печатаемых цветов (слово) должен быть напечатан непосредственно за фразой «На левой проекции отсутствуют цвета:» через пробел, а все последующие печатаются в формате: <*запятая*><*пробел*><*цвет*>.

Соответствующий фрагмент программы имеет вид:

```
Print " ";                      'печать пробела
For i=1 To 9
  Print B(1,i);                 'печать буквенных обозначений столбцов
Next i
Print                           'переход на следующую строку
For i=1 To 8
  Print i;                      'печать номера строки
  For j=1 To 8
    Select Case A(i,j)          'выбор символа для печати
    Case 0
      Print "*";                'печать звездочки "*"
    Case 1
      Print B(2,j);             'печать буквенного обозначения цвета
      End Select
    Next j
    Print                        'переход на следующую строку
Next i
```

Определение отсутствующих цветов и печать ответа. Так как у нас получилась числовая матрица, состоящая из нулей и единиц, то можно поступить следующим образом. Просмотрим все элементы матрицы по столбцам, начиная со второго. Для каждого столбца соответствующий цвет на левой проекции будет невидимым, если *для всех* элементов текущего столбца, значения которых равны 1, сумма всех предшествовавших значений элементов в каждой строке больше нуля.

Можно использовать и другое логическое выражение (точнее — его отрицание): если *хотя бы для одного* из элементов текущего столбца, значение которого равно 1, сумма всех предшествовавших значений элементов в каждой строке равна нулю, то этот цвет *отсутствовать не будет*, т. е. он будет *видимым* (это проще в реализации).

Пример окончательного варианта программы:

```
Dim B(3, 8) As String
Dim A(8, 8) As Integer
Dim V(2), k As Integer
```

```
    'формирование текстового массива непосредственно в программе
B(1, 1) = "a"
B(1, 2) = "b"
B(1, 3) = "c"
B(1, 4) = "d"
B(1, 5) = "e"
B(1, 6) = "f"
B(1, 7) = "g"
B(1, 8) = "h"
B(2, 1) = "Б"
B(2, 2) = "К"
B(2, 3) = "О"
B(2, 4) = "Ж"
B(2, 5) = "З"
B(2, 6) = "Г"
B(2, 7) = "С"
B(2, 8) = "Ф"
B(3, 1) = "белый"
B(3, 2) = "красный"
B(3, 3) = "оранжевый"
B(3, 4) = "желтый"
B(3, 5) = "зеленый"
B(3, 6) = "голубой"
B(3, 7) = "синий"
B(3, 8) = "фиолетовый"
    'Блок распределения положений кубиков на доске
For i = 1 To 8                  'обнуление массива кубиков
  For j = 1 To 8
    A(i, j) = 0
  Next j
Next i
For i = 1 To 8                  'цикл выбора номера столбца матрицы
    Randomize
    n = Int(1 + 3 * Rnd)        'генерация количества кубиков
                                'в данном столбце
  For j = 1 To 2                'обнуление вспомогательной матрицы
    V(j) = 0
  Next j
  Randomize
  q = Int(1 + 8 * Rnd)          'генерация позиции первого кубика
                                'в текущем столбце
  V(1) = q
  A(q, i) = 1                   'запоминаем позицию первого
                                'кубика в столбце
  If n > 1 Then                 'если для столбца сгенерировано
                                'более одного кубика, то
    j = 2
  Do                            'цикл, ограничивающий генерацию
                                'позиции
```

```
                                'для 2-го (и 3-го) кубика
    Do                          'цикл,
      Do                        'цикл повторной генерации позиции
                                'кубика
        Randomize
        k = Int(1 + 8 * Rnd)    'генерация новой позиции кубика
                                'в столбце
        p = 1
      Loop While V(p) = k       'если новая позиция кубика совпадает
                                'с позицией первого, то генерацию
                                'надо повторить
      p = p + 1
      If p > j - 1 Then Exit Do 'условие прекращения генерации
                                'новой позиции
    Loop While V(p) = k         'если новая позиция кубика совпадает
                                'с позицией второго, то генерацию
                                надо повторить
    V(j) = k                    'запомнить очередную позицию
                                'кубика в столбце
    A(k, i) = 1                 'запомнить положение очередного
                                'кубика в матрице
    j = j + 1
  Loop While j < n              'если позиции всех кубиков
                                'сгенерированы, то выйти
  End If
Next i

   'Печать схемы расположения кубиков в соответствии с условием
Print " ",                      'печатаем пробел в первой позиции
For i = 1 To 8
  Print B(1, i),                'печатаем буквенные обозначения
                                'столбцов
Next i
  Print                         'переход на новую строку
  For i = 1 To 8                'цикл печати расположения кубиков
    Print i,                    'печать номеров строк
    For j = 1 To 8
      Select Case A(i, j)       'выбор символа для печати
      Case 0
        Print "*",              'если элемент матрицы равен "0" -
                                'печатаем "*"
     Case 1
      Print B(2, j),            'если элемент матрицы равен "1" -
                                'печатаем букву, соответствующую цвету
    End Select
Next j
Print                           'переход на печать со следующей строки
Next i

'Завершение расчетов и печать результата

Print "На левой проекции отсутствуют цвета:";
```

```
flag = 0                        'вспомогательная переменная,
                                'определяющая, есть ли отсутствующие
                                'цвета, выведенные на печать (1 – да,
                                '0 – нет)
For i = 2 To 8
  For j = 1 To 8
    If A(j, i) = 1 Then         'условие наличия кубика в данной
                                'позиции
      S = 0                     'обнуление суммы
      For p = 1 To i - 1        'цикл формирования суммы значений
                                'элементов строки до текущей позиции
        S = S + A(j, p)
      Next p
      If S = 0 Then             'если хотя бы один кубик видим слева,
                                'то
      GoTo 10                   'переходим к рассмотрению
                                'следующего столбца
      End If
    End If
  Next j
    If flag = 1 Then            'если хотя бы одно слово,
                                'обозначающее цвет,
                                'уже напечатано и есть для печати
                                'следующее слово, то
      Print ", "; B(3, i);      'печатаем сначала ",", а потом само
                                'слово
    Else
      Print " "; B(3, i);       'печать первого слова,
                                'обозначающего цвет
    flag = 1                    'фиксируем факт, что первое слово
                                'уже напечатано
  End If
10: Next i
```

5. «Бильярд Льюиса Кэрролла»

Эта задача является модификацией задачи, представленной в [9].

Можно решать данную задачу, используя чисто геометрическую двумерную модель, т. е. рассматривать геометрически проход шаром каждого отрезка до соударения с бортом, подсчитывать количество отрезков траектории и, проверяя условия попадания шара в лузу, в итоге ответить на поставленные в задаче вопросы. Но возможны и другие модели, которые существенно упрощают решение задачи. Рассмотрим три различных способа моделей решения данной задачи.

Способ 1 — геометрический

Свяжем с бильярдным столом декартову систему координат так, как показано на рис. 16.

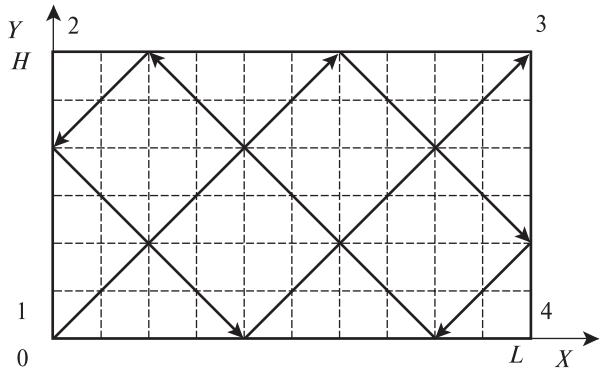


Рис. 16. Бильярдный стол в декартовой системе координат

Из этого рисунка видно, что:

- луза 1 имеет координаты (0, 0);
- луза 2 имеет координаты (0, *H*);
- луза 3 имеет координаты (*L*, *H*);
- луза 4 имеет координаты (*L*, 0).

Условие попадания шара в лузу с соответствующим номером — равенство текущих координат шара указанным выше координатам лузы.

Условие отскока (*изменения траектории*):

- от верхнего борта — $Y = H$;
- от нижнего борта — $Y = 0$;
- от левого борта — $X = 0$;
- от правого борта — $X = L$.

В этом случае *алгоритм решения задачи* — очень простой. Так как задача решается в целых числах, то мы даем единичные приращения координате *X* и координате *Y*. Знак приращения как *X*, так и *Y* зависит от того, произошло ли перед этим отражение от какого-либо борта. Кроме того, при каждом отражении от борта увеличивается значение переменной — счетчика количества отрезков ломаной траектории движения шара.

Текст программы на языке Basic:

```
Dim X, Y, H, L As Integer
Input "Введите значение H"; H
Input "Введите значение L"; L
X = 0              'Начальное значение координаты X'
```

```
Y = 0              'Начальное значение координаты Y'
ShagX = 1          'Шаг изменения координаты X'
ShagY = 1          'Шаг изменения координаты Y'
K0 = 1             'Начальное значение количества отрезков траектории'
Do                 'Цикл пошагового прохождения траектории'
  X = X + ShagX    'Меняем значение координаты X'
  Y = Y + ShagY    'Меняем значение координаты Y'
  If X = L And Y = H Then
    Print " Шар попал в лузу 3"
    Exit Do
    ElseIf X = L And Y = 0 Then Print "Шар попал в лузу 4": Exit Do
    ElseIf X = 0 And Y = H Then Print "Шар попал в лузу 2": Exit Do
    ElseIf (X = L Or X=0) And 0 < Y < H Then ShagX = -ShagX: K0 = K0 + 1
    ElseIf (Y = H Or Y=0) And 0 < X < L Then ShagY = -ShagY: K0 = K0 + 1
  End If
Loop
Print "Количество отрезков ломаной траектории шара K0="; K0
```

Этот способ решения задачи достаточно прост как в понимании модели решения, так и в ее реализации. Недостатком можно считать лишь то, что при больших размерах стола и/или достаточно длинной траектории движения шара (при большом количестве отражений) время работы программы может оказаться очень большим.

Способ 2 — поиск закономерностей

Используем для решения задачи поиск закономерностей. Это не самый оптимальный способ с точки зрения нахождения модели решения (поиск закономерностей может занять достаточно большое время), но когда такая модель найдена, это может существенно ускорить решение.

Для начала наглядно изобразим несколько возможных вариантов траекторий движения шара по бильярдному столу с различными значениями L и H: будем выбирать для четных и нечетных значений L различные значения H (большие L и меньшие L, четные и нечетные). При этом не забудем, что эти величины являются целыми числами, — для этого на рисунках нанесем сетку с единичными квадратными ячейками.

Изобразим несколько различных вариантов траекторий движения шара по столу для случая $H > L$. (Если мы решим задачу для этого случая, то для варианта $H < L$ достаточно сначала поменять местами значения L и H, решить задачу по предыдущему варианту, а затем при необходимости поменять местами номера луз 2 и 4. Это, кстати, еще один полезный прием решения задач по программированию — использование готовых решений или

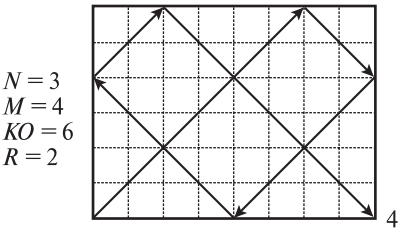


Рис. 17

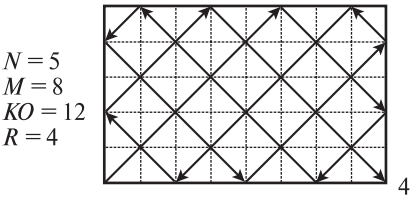


Рис. 18

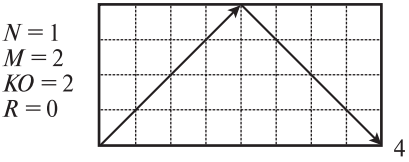


Рис. 19

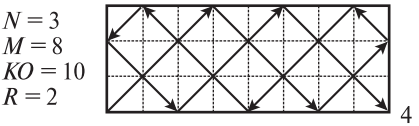


Рис. 20

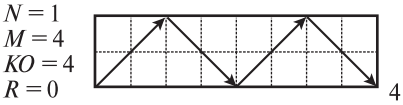


Рис. 21

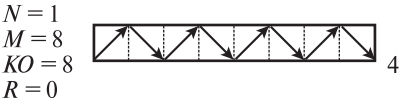


Рис. 22

сведение решения поставленной задачи к уже известному варианту.)

Для упрощения на рис. 17—22 указаны только те номера луз, в которые попадает шар. Буквами обозначены:

N — количество проходов шара вдоль длинной (горизонтальной) стороны стола (от одной боковой стороны до другой) до попадания в лузу;

M — количество проходов шара вдоль короткой (вертикальной) стороны стола (от верхнего борта до нижнего или наоборот) до попадания в лузу;

KO — количество отрезков траектории шара до попадания в лузу;

R — количество отражений от боковых сторон стола.

Рассмотрим внимательно изображенные на рисунках траектории движения шара на бильярдных столах, у которых при постоянной длине в восемь единиц ширина стола меняется на одну единицу. Сразу же обратим внимание, что во *всех* этих случаях шар попадает в лузу 4. Случайность ли это или какая-то закономерность? И если закономерность, то от чего зависит попадание шара в лузу 4?

Все эти варианты объединяет одно общее свойство — *нечетное значение* $N(1, 3, 5)$ *и четное значение* $M(2, 4, 8)$. Для определения закономерности такого малого количества рисунков может показаться недостаточно. Но если немного подумать, то нетрудно догадаться, что для попадания в лузу 4 шар должен пройти именно *нечетное* число раз вдоль длинной стороны стола (в направлениях слева направо и справа налево) и *четное* число раз вдоль короткой стороны (снизу вверх и сверху вниз).

Продолжая размышлять дальше, нетрудно прийти к выводу, что для попадания шара в лузу 3 он должен пройти и вдоль длинной, и вдоль короткой стороны *нечетное* количество раз.

Аналогично, чтобы шар попал в лузы, расположенные слева, он должен пройти вдоль длинной стороны *четное* количество раз. При этом, чтобы шар попал в лузу 2, он должен пройти вдоль короткой стороны *нечетное* количество раз.

И самое главное — шар *никогда не вернется* в лузу с номером 1. Это вполне очевидно, так как для возврата шара в лузу 1 он должен пройти весь путь туда и обратно по одной и той же траектории, а это возможно только при отражении от одного из углов. Но в углах стола находятся лузы, и если шар попадает в угол стола, то он попадает в одну из луз, а это означает окончание работы программы.

На рис. 23—28 показано еще несколько вариантов изображений бильярдного стола для нечетного значения размера длинной стороны ($H = 9$) и различных значений размеров короткой стороны L (от 6 до 1) с траекториями движения шара, которые подтверждают сделанные выводы и добавляют статистическую информацию.

Итак, одну закономерность, определяющую попадание шара в ту или иную лузу, мы нашли. Если мы подсчитаем количество проходов шара вдоль длинной и короткой сторон, то по сопоставлению четности и нечетности полученных значений мы сможем определить, в какую лузу попадет шар. Поэтому принципиально эту часть задачи можно считать решенной (ее техническую реализацию мы рассмотрим чуть позже).

Теперь нам нужно определить количество отрезков, из которых состоит траектория движения шара по столу. Нет ли и здесь каких-либо закономерностей?

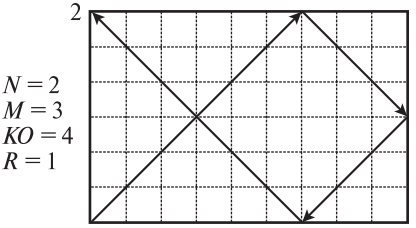


Рис. 23

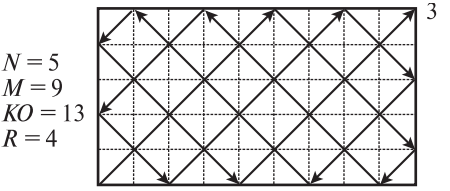


Рис. 24

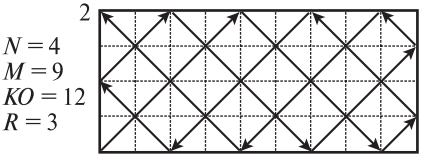


Рис. 25

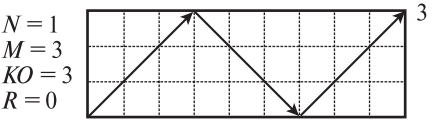


Рис. 26

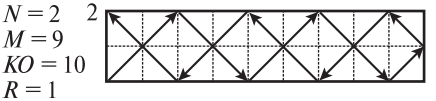


Рис. 27

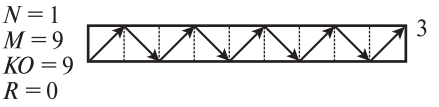


Рис. 28

Можно заметить, что в при отсутствии отражений от боковых сторон количество отрезков (O), из которых состоит траектория движения шара, будет в точности совпадать с количеством проходов шара вдоль боковой стороны (M). При наличии же таких отражений количество отрезков равно сумме числа проходов шара вдоль боковой стороны (K) и количества отражений от обеих боковых сторон (R). Для наглядности все упомянутые значения, соответствующие рассмотренным ранее рисункам, сведем в таблицу:

Номер рисунка	*N*	*M*	*R*	*KO*	Номер лузы, в которую попадает шар
2	3	4	2	6	4
3	5	8	4	12	4
4	1	2	0	2	4
5	3	8	2	10	4
6	1	4	0	4	4
7	1	8	0	8	4
8	2	3	1	4	2
9	5	9	4	13	3
10	4	9	3	12	2
11	1	3	0	3	3
12	2	9	1	10	2
13	1	9	0	9	3

Таким образом, чтобы определить количество отрезков, составляющих ломаную траекторию движения шара, необходимо подсчитать количество проходов шара вдоль короткой стороны и прибавить к нему количество отражений шара от боковых сторон.

Заметим, что при длине стола H, кратной ширине стола L, шар за один проход вдоль длинной стороны попадет в одну из луз (3 или 4 — см. рис. 19, 21, 22, 26, 28). В этом случае количество проходов вдоль длинной стороны $N = 1$, количество проходов вдоль короткой стороны $M = H/L$ (целое число), количество отражений от боковых сторон $R = 0$, а количество отрезков, из которых состоит траектория движения шара, — $KO = M + R$. Определение количества отрезков траектории движения шара в этом случае завершено.

В других случаях (при отсутствии кратности длины и ширины стола) будет происходить отражение от боковой стороны, и шар будет продолжать движение, но уже в обратную сторону (рис. 29).

В этом случае нужно зафиксировать факт отражения и продолжить расчеты, учитывая, что шар закончил очередной проход

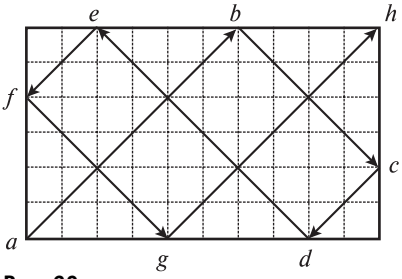


Рис. 29

вдоль длинной стороны (участок траектории *abc*), но еще не закончил очередной проход вдоль короткой стороны (шару осталось пройти отрезок *cd*). При этом шар совершает количество полных проходов вдоль короткой стороны, равное *целой части* значения L/H (запись на языке Basic: `Int(L/H)`; на рис. 29 это число равно 1). Для завершения же еще одного прохода вдоль короткой стороны мы должны «занять» расстояние, равное разности H и *дробной части* от L/H (запись на языке Basic: `H - (L Mod H)`; на рис. 29 это число равно 2 — проекция отрезка *cd* на длинную сторону стола), от последующего прохода вдоль длинной стороны стола. Таким образом, для завершения движения к левой (на рисунке) боковой стороне шар должен пройти по оставшемуся пути вдоль длинной стороны расстояние, равное `L - (H - L Mod H)` (на рис. 29 это проекция траектории *def* на длинную сторону стола, равная длине отрезка *da*). Если эта оставшаяся часть пути шара вдоль длинной стороны стола будет кратна ширине стола H, то шар попадет в лузу, и мы должны выполнить расчеты по увеличению количества проходов вдоль длинной и короткой сторон, аналогичные описанным выше. Иначе мы опять повторяем «заем» расстояния от следующего прохода вдоль длинной стороны и выполняем уже описанные действия, используя вместо значения L значение `L - (H - L Mod H)`. Тем самым цикл замкнется (см. начало этого абзаца).

Приведем здесь только блок-схему данного алгоритма решения задачи для случая $L > H$ (рис. 30).

Это — пример последовательного решения задачи (так же как и по способу 1), поскольку в обеих описанных моделях последовательно рассматривается проход шара вдоль длинной стороны стола и проверяются условия окончания расчетов (попадание шара в лузу). Но данный способ позволяет решить задачу быстрее, чем предыдущий, поскольку «шагом» здесь, по сути, является не единица длины, а размер короткой стороны H.

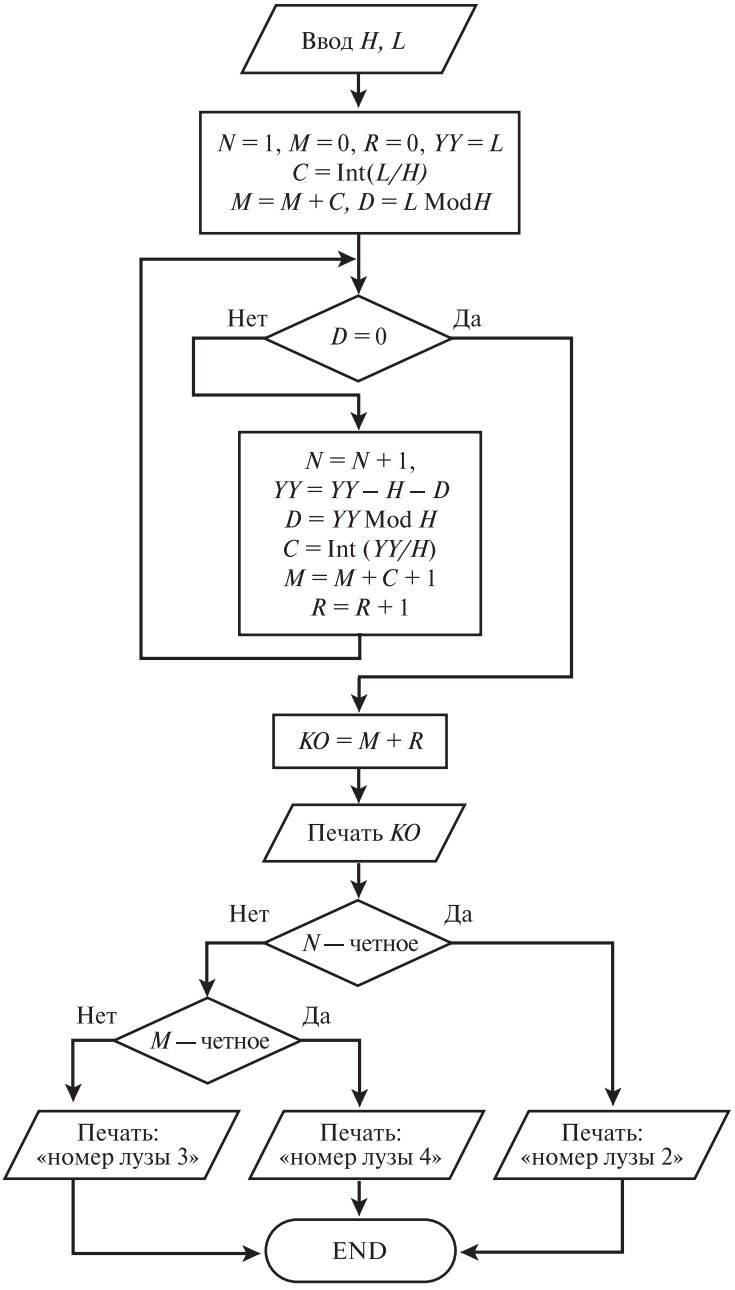

Ввод H, L
N = 1, M = 0, R = 0, YY = L
C = Int(L/H)
M = M + C, D = L ModH
Нет
D = 0
Да
N = N + 1,
YY = YY – H – D
D = YY Mod H
C = Int (YY/H)
M = M + C + 1
R = R + 1
KO = M + R
Печать KO
Нет
N — четное
Да
Нет
M — четное
Да
Печать:
«номер лузы 3»
Печать:
«номер лузы 4»
Печать:
«номер лузы 2»
END

Рис. 30

Способ 3 — развертка траектории движения шара в линию

Поставленную задачу можно решить и еще одним способом, рассмотрев третий вариант модели, который, на наш взгляд, гораздо нагляднее и проще для понимания.

Вернемся к рис. 29 и размножим изображения стола по горизонтали и вертикали, построив его зеркальные отображения относительно короткой и/или длинной сторон так, чтобы отрезки траектории движения шара вытянулись в прямую линию (в один отрезок) (рис. 31). При этом зеркальные отображения стола построены относительно тех бортов, о которые ударяется шар. Места соударения шара с бортами отмечены точками. Для наглядности обозначены номера луз. Буквенные обозначения отрезков траектории соответствуют приведенным ранее на рис. 29. Для наглядности на рисунке также изображена сетка с единичными ячейками.

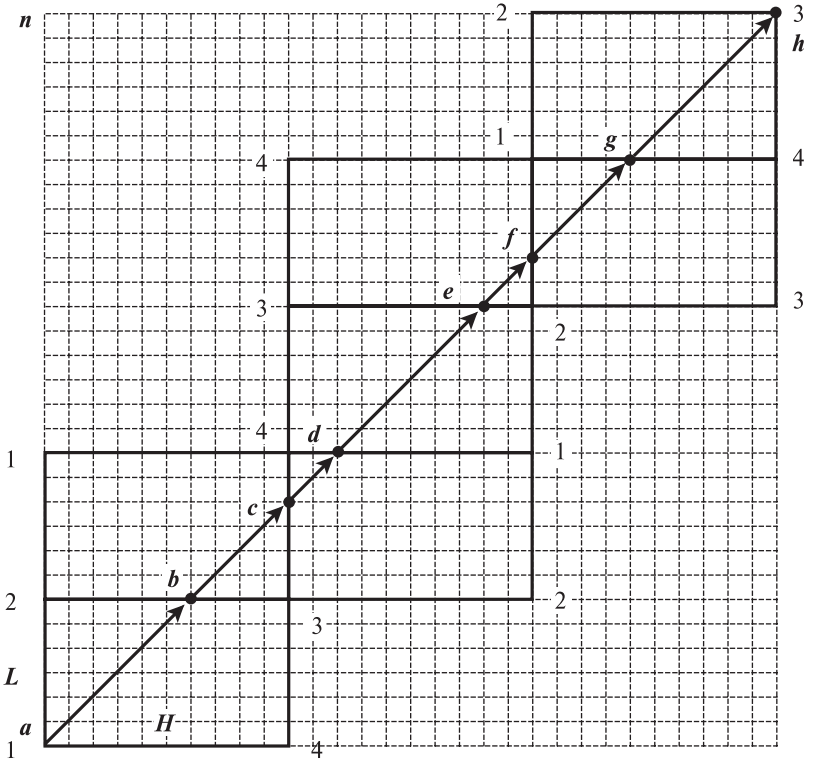


Рис. 31. Развертка траектории движения шара в линию

Из рис. 31 можно определить, что:

1) фигура *anhm* является квадратом;

2) проекция пройденного пути на вертикальную сторону квадрата (отрезок *an*) кратна величие L, а проекция пройденного пути на горизонтальную сторону квадрата (отрезок *am*) кратна величине H, т. е. сторона полученного квадрата является *наименьшим общим кратным* (переменная *NOK*) размеров бильярдного стола L и H. Это является также условием попадания шара в лузу;

3) количество отрезков траектории движения шара может быть определено как количество отражений от длинных бортов стола (NOK/H, считая отражением попадание в лузу) плюс количество отражений от боковых бортов (NOK/L) без 1. Вычитание единицы при этом требуется, поскольку последнее отражение от бокового борта (попадание в лузу) происходит одновременно с отражением от длинного борта, а оно уже учтено в первом слагаемом. В результате получаем следующую формулу: $KO = NOK/H + NOK/L - 1$;

4) номер лузы, в которую попадет шар, определяется точно так же, как и в предыдущем способе решения задачи.

Тем самым решение задачи по этому способу существенно сократилось. Единственное, что осталось здесь добавить, — это процедуру определения наименьшего общего кратного для двух целых величин L и H. Ее можно реализовать, например, в виде функции:

```
Function NOK(h,l) As Integer
Dim h, l As Integer
I=1
LL=L
Do Until LL Mod H=0
  I=I+1
  LL=L*I
Loop
NOK=LL
End Function
```

Определение четности целого числа A можно реализовать путем определения остатка от деления числа на 2. Например, если после вычисления `C = A Mod 2` значение $C = 1$, то число A — нечетное, а если $C = 0$, то четное.

Полный текст программы решения задачи по способу 3 тогда будет выглядеть следующим образом:

```
Function NOK (H, L) As Integer
i = 1
LL = L
Do Until LL Mod H = 0
```

```
    i = i + 1
    LL = L * i
  Loop
  NOK = LL
  End Function
  Dim H, L, F, KO As Integer
  Input "Введите значение H"; H
  Input "Введите значение L"; L
  F = NOK (H, L)
  KO = F / H + F / L - 1
  Print "Количество отрезков ломаной траектории шара:"; KO
  If (N Mod 2) = 0 Then
    Print "Шар попадет в лузу 2"
    ElseIf M Mod 2 = 0 Then
    Print "Шар попадет в лузу 4"
    Else
    Print "Шар попадет в лузу 3"
  End If
```

Нетрудно видеть, что реализация такого решения оказывается наиболее оптимальной как по количеству операторов в тексте программы, так и по общему количеству вычислений (выполняемых операций).

В заключение еще раз обратим ваше внимание на то, что вы сами определяете (находите) модель решения поставленной задачи, используя знания из других дисциплин (алгебра, геометрия, физика), при этом выбираемая модель должна быть максимально наглядной и понятной. А что касается конкретных реализаций решения, приводящих к верному ответу (конечному результату), то их может быть много.

■ МЕЖДУНАРОДНЫЕ ОЛИМПИАДЫ. ТУРНИР АРХИМЕДА

1. «У моего папы больше денег» (турнир III, задача C)

В этой задаче требуется всего лишь вывести одно любое число, которое больше введенного. Однако первое приходящее на ум решение — прибавить к введенному числу 1 — в большинстве языков программирования работать не будет: вводимое число может содержать до 100 знаков и не поместится ни в один из стандартных типов данных.

Впрочем, на языке Python, который поддерживает длинные числа, такое решение будет работать. Его можно записать всего в одну строку:

```
print(int(input()) + 1)
```

В этой строке кода содержатся сразу три функции:

- `input()` — читает одну строку со стандартного входа (с клавиатуры); обратите внимание — в языке Python всегда читается именно строка, а не число, символ и т. п. (преобразование входных данных к нужному типу возложено на программиста);
- `int()` — функция для преобразования данных (строки) к целому типу;
- `print()` — выводит свои параметры в поток стандартного ввода (на экран); если параметров несколько, то при выводе они по умолчанию разделяются пробелами.

Следующая достаточно очевидная идея — напечатать число, в 10 раз большее заданного. Для этого достаточно приписать к вводимому числу (к его записи) в конце цифру 0. Считывать это число тогда можно в строковую переменную:

```
print(input() + "0")
```

В предыдущем примере операция «+» использовалась для сложения чисел, а здесь ее аргументы — строки. Сложение (конкатенация) строк — это простое приписывание одной строки к другой. Обратите внимание, что нам нужна *строка* 0, поэтому этот символ взят в кавычки (вместо них можно также использовать апострофы).

Но оказывается, что есть решение еще проще (в некотором смысле)! Поскольку все числа в условии ограничены (состоят не более чем из 100 цифр), то можно независимо от вводимого числа вывести, например, число 10...0 (100 нулей) или даже число 11...1 (101 единица):

```
print(10 ** 100)
```

или

```
print("1" * 101)
```

Напомним, что знак «*» обозначает операцию умножения, а знаки «**» в языке Python означают возведение в степень. «Умножение» же строки на число в Python — это просто сложение строки самой с собой указанное число раз:

```
"1" * 101 = "1" + "1" + ... + "1" (101 раз).
```

Обратите внимание: в условии задачи имеется ограничение на значение числа первого игрока, но нет ограничений на число второго, поэтому все наши решения — корректные!

Методический комментарий. В этой задаче (при использовании третьего из представленных вариантов ее решения) не требуется даже читать входные данные. А в некоторых других олимпиадных задачах требуется прочитать лишь часть входных

данных. Вообще же правила олимпиад не требуют, чтобы программа обязательно считывала всю информацию из входного потока.

2. Карточки (турнир I, задача I)

Поскольку Петя открывает карточки по очереди слева направо, то как только он дойдет до карточки с номером *i* или *j*, он увидит, что карточка лежит не на своем месте, и по ее номеру узнает, где она должна была лежать. Тем самым он сразу определит и *i*, и *j*.

А может ли он узнать об этом раньше? Оказывается, есть одна ситуация, в которой это возможно. Если Вася поменял местами карточки с номерами 99 и 100, то, открыв 98 карточек и не обнаружив нарушений порядка, Петя сразу поймет, что обменены две оставшиеся карточки, и открывать их не станет.

Осталось написать программный код:

```
i, j = map(int, input().split())
```

Входные данные вводятся в одной строке через пробел. Функция `input()` считывает их, а затем к полученной строке применяется метод `split()`, «разрезающий» ее по пробелу или группе пробелов и превращающий в список (так в языке Python называют массивы) строк. А чтобы превратить каждую строку в этом списке в целое число, применяется функция `map`, первый аргумент которой — функция `int`, применяемая к каждому элементу списка, указанного во втором аргументе.

```
if i + j == 99 + 100:
  print(98)
else:
  print(min(i,j))
```

Обратите внимание на уникальную особенность языка Python: вместо использования фигурных скобок или конструкций `begin..end` группа операторов, выполняющихся внутри другого оператора `(if, for, while...)`, выделяется отступом. Знаки «`==`» обозначают операцию проверки на равенство, а «`=`» — оператор присваивания. Стандартная функция `min` (определение минимального из двух чисел) работает с любыми типами данных и со списками любой длины.

Обратите внимание: чтобы не проверять два случая — $i = 99$, $j = 100$ и $i = 100$, $j = 99$, мы воспользовались тем фактом, что никакие другие два числа от 1 до 100 не могут давать ту же самую сумму, что 99 и 100.

Методический комментарий. В этой программе отчетливо выделяется крайний случай — особое поведение при максимальном (иногда — минимальном) значении данных.

3. Кольцевая (турнир III, задача F)

Заметим: если бы автомобили двигались по прямой, то они удалялись бы друг от друга со скоростью $v_1 + v_2$. Тогда за время T они оказались бы на расстоянии $(v_1 + v_2) \cdot T$. А поскольку они движутся по кругу длиной L, то, чтобы найти расстояние между ними (по одной из дуг окружности), достаточно взять остаток: $r = ((v_1 + v_2) \cdot T) \bmod L$. Тогда длина второй дуги окружности между ними будет равна $(L - r)$. Остается только вывести на экран минимальное из чисел r и $(L - r)$.

```
L, v1, v2, t = map(int, input().split())
r = (v1 + v2) * T % L
print(min(r, L - r))
```

Здесь `%` — операция взятия остатка от деления.

4. Сумма цифр (турнир II, задача G)

Прежде всего, заметим, что из-за больших ограничений (диапазон может содержать до миллиарда чисел) алгоритм, перебирающий каждое число и проверяющий его сумму цифр, не будет укладываться в ограничения по времени работы программы (1 с) и не пройдет все тесты.

Для решения задачи заметим, что в каждом полном десятке содержится ровно 5 чисел (каждое второе), которые имеют четную сумму цифр. Поэтому нам достаточно подсчитать количество полных десятков, находящихся в предложенном диапазоне, и вручную проверить несколько чисел в начале и в конце диапазона. Например, если $a = 107$, а $b = 10\,342$, то потребуется рассмотреть отдельно числа 107, 108, 109, 10 340, 10 341 и 10 342, а затем подсчитать количество десятков в диапазоне от 110 до 10 339, — их там окажется $(10\,340 - 109 - 1)/10 = 1023$.

Покажем теперь, как по заданным числам a и b найти наименьшее число, оканчивающееся на 0 и не меньшее a, а также наименьшее число, оканчивающееся на 0 и не большее b:

```
bzero = b // 10 * 10
azero = (a + 9) // 10 * 10
```

или

```
azero = math.ceil(a / 10) * 10
```

Функция `ceil` из стандартной библиотеки `math` округляет число до ближайшего целого сверху. Для использования функ-

ции из этой библиотеки нужно подключить библиотеку командой `import`:

```
import math
```

Далее можно использовать функцию, обращаясь к ней в формате:

```
имя_библиотеки.имя_функции
```

5. Бегуны (турнир I, задача E)

Развернем беговую дорожку стадиона в числовую прямую, приняв за начало отсчета точку старта. Тогда, подсчитав перемещение бегунов относительно старта в результате выполнения всех заданий тренера (соответственно, X_1 и X_2), определим расстояние между ними в конце тренировки: $S = |X_2 - X_1|$. Чтобы перенести это расстояние на круговую беговую дорожку длиной в 400 м, необходимо определить остаток от деления S на 400. Для нахождения минимального расстояния между бегунами полученный остаток надо сравнить с 200, и если окажется, что он больше 200, то необходимо вычесть его из 400. Полученное число и будет искомым результатом.

6. Тапочки (турнир I, задача H)

Сразу заметим, что единственная ситуация, когда тапочки надеть не удастся, — когда все правые тапочки стоят левее левых, т. е. когда сначала стоят все 10 правых тапочек, а затем — 10 левых. В остальных случаях обязательно найдется хотя бы один левый тапочек, за которым следует правый, поэтому ответ в задаче будет равен 0. Учитывая все это, для решения задачи достаточно проверить, верно ли, что входная последовательность начинается с 10 нулей: если да, то следует вывести ответ «–1», а если нет, — то ответ «0».

```
s = input()
#если в начале вводимой строки 10 раз повторяется
#последовательность нуль-пробел
if s[:20] == '0 ' * 10:
  print(-1)
else:
  print(0)
```

Здесь мы впервые воспользовались мощным инструментом языка Python для работы со строками — *срезом*. `s[a:b]` — это подстрока строки s, начиная с символа с номером a и до символа с номером b (включая a, но не включая b). В нашем случае $a = 0$,

и это — значение по умолчанию, поэтому перед двоеточием мы ничего не указали.

7. Робот (турнир II, задача A)

Легко понять, что кратчайший путь может быть получен следующим образом: сначала делается требуемое количество ходов по диагонали, а потом при необходимости добавляется несколько ходов по вертикали и/или горизонтали. При этом количество ходов по диагонали равно min (N, M), а количество горизонтальных/вертикальных ходов равно $|M - N|$.

```
N, M = map(int, input())
print('D' * min(N, M), end='')
if N > M:
  print('U' * (N - M))
else:
  print('R' * (M - N))
```

По умолчанию каждая функция `print` после вывода данных выполняет переход на новую строку. Чтобы повлиять на это поведение, здесь мы воспользовались *именованным параметром* `end` — ему присваивается строка, которую нужно напечатать в конце вывода, в данном случае это пустая строка.

Методический комментарий. Для случая, когда $N = M$, данное решение тоже выдает верный ответ, поскольку в этом случае выполняется ветка `else` условного оператора, а если $M - N$ равно 0, то никаких дополнительных символов, кроме «D», напечатано не будет.

8. Квартиры (турнир II, задача B)

Пусть первая квартира в некотором подъезде имеет номер x, а последняя — y. Тогда в этом (а значит, и в любом другом) подъезде имеется $(y - x + 1)$ квартир. Но во всех предыдущих подъездах вместе $(x - 1)$ квартир. Следовательно, $(x - 1)$ должно делиться на $(y - x + 1)$.

```
if (x - 1) % (y - x + 1) == 0:
  print('YES')
else:
  print('NO')
```

Методический комментарий. Следует обратить внимание на правильность расстановки скобок в условном операторе: часто школьники забывают ставить их и записывают условие `x - 1 % y - x + 1 == 0`, которое имеет совсем другой смысл: поскольку операция `%` имеет больший приоритет, чем «+» и «–», в подобном случае вычисляется остаток от деления 1 на y.

9. Турнир (турнир II, задача F)

Для решения этой задачи не требуется особых навыков программирования. Она предназначена в первую очередь для тех, кто умеет решать логические задачи и только начинает изучать язык программирования.

1. Рассмотрим сначала случай, когда количество команд нечетно. Тогда можно добиться, чтобы все команды набрали одинаковое количество очков и стали победителями. Для этого мысленно расположим команды по кругу и будем считать, что команда выиграла у $N//2$ команд, следующих за ней по часовой стрелке. Тогда оставшиеся $N//2$ команд выиграли у данной команды (а это как раз $N//2$ команд, стоящих перед ней). В результате *каждая* команда выиграла ровно у $N//2$ команд. (Например, если команд 3, то 1-я выиграла у 2-й, 2-я — у 3-й, а 3-я — у 1-й.)

2. Если N четно, то добиться, чтобы все команды стали победителями, невозможно. Действительно, в этом случае каждая команда сыграла нечетное количество матчей, и если бы все команды одержали одинаковое количество побед, то либо все они должны были выиграть строго больше половины своих матчей (что, очевидно, невозможно), либо все должны были проиграть больше половины матчей (что также невозможно).

Покажем, что все, кроме одной команды, могли стать победителями. Действительно, пусть одна команда проиграла все свои матчи, а остальные $N - 1$ команд (их количество четно) сыграли друг с другом так, как описано в случае 1.

Осталось написать программу:

```
if N % 2 == 0:
  print(N - 1)
else:
  print(N)
```

10. Автомат по продаже билетов (турнир I, задача J)

Данная задача может показаться сложнее, чем есть на самом деле. Однако заметим, что нас не просят обойтись минимальным количеством монет и купюр при выдаче сдачи, поэтому достаточно было бы вообще выдать всю сдачу рублевыми монетами. Следовательно, задача сводится к тому, чтобы подсчитать сумму сдачи, т. е. найти сумму всех опущенных в автомат купюр и вычесть из нее стоимость билета.

```
#читаем все входные данные в один список
s = list(map(int, input().split()))
```

Раньше мы использовали конструкцию `a, b, c = map(...)`. В этой задаче мы хотим сохранить все считанные числа в один список *s*. Для этого мы используем функцию `list` приведения к одноименному типу *список*.

```
#s[0] – стоимость билета, s[1] – количество купюр (не используется)
res = sum(s[2:]) - s[0]
print('1' * res)
```

Мы уже применяли срезы к строкам, а теперь пришел черед списков. `s[2:]` — это все элементы списка от второго до последнего (нумерация с нуля). Функция `sum`, как несложно догадаться, вычисляет сумму всех элементов списка.

Обратите внимание: в языке Python все символы строки, начинающейся с символа #, представляют собой комментарии.

11. Конь (турнир II, задача D)

Те, кто играл в шахматы, хорошо знают, что на шахматной доске (8×8) конь может добраться с любой клетки на любую другую. С другой стороны, скажем, на доске 2×2 у коня вообще нет ходов. Отсюда ясно, что задача имеет нетривиальный ответ только для маленьких досок.

Выясним прежде всего, для каких досок ответ будет всегда YES независимо от выбора начальной клетки. Покажем сначала, что на доске 3×4 это обстоит именно так. На рисунке показано, как обойти конем всю доску (цифры — это порядковые номера «посещаемых» конем клеток). Следовательно, из любой клетки доски размером 3×4 (а значит, и любой другой доски, которая заключает в себе прямоугольник размером 3×4) можно попасть в любую другую клетку.

1	4	7	10
12	9	2	5
3	6	11	8

Для досок размером $1 \times N$ очевидно, что ответ NO для любой пары клеток.

Для доски размером 3×3 ответ YES, если ни одна из клеток (ни начальная, ни конечная) не является центральной, и NO — в противном случае (из любой клетки, кроме центральной, можно попасть в любую другую (см. рис.).

1	4	7
6		2
3	8	5

Осталось рассмотреть доску размером $2 \times N$. На такой доске все клетки разбиваются на цепочки (рис. 32). Две клетки принадлежат одной цепочке, если они находятся в одном

Рис. 32

ряду на расстоянии $4 \cdot K$, либо в разных рядах на расстоянии $4 \cdot K + 2$. Конь может попасть из одной клетки в другую тогда и только тогда, когда обе эти клетки лежат в одной цепочке.

12. Уравнение (турнир I, задача B)

Заметим, что при $c < 0$ решений нет. Если же $c \geqslant 0$, то можно возвести данное уравнение в квадрат: $ax + b = c^2$, а затем достаточно будет найти целые решения этого линейного уравнения. Такое уравнение имеет бесконечно много решений, когда $a = 0$ и $c^2 - b = 0$. Оно не имеет решений в целых числах, если $a = 0$, а $c^2 - b \neq 0$, а также когда $c^2 - b$ не делится нацело на a. В остальных случаях оно имеет единственное решение: $(c^2 - b)$ // a (напомним, что символы «//» обозначают в языке Python операцию целочисленного деления).

13. Бесконечная таблица (турнир I, задача C)

Заметим, что в i-м слева столбце расположены числа от $(i - 1)^2 + 1$ до i^2.

Рассмотрим вводимое число N. Сначала определим номер столбца, в котором оно находится:

```
from math import ceil, sqrt
```

Если при подключении библиотеки указать, какие функции из нее мы планируем использовать, то в дальнейшем их можно использовать, не указывая имени библиотеки.

```
col = ceil(sqrt(N))
```

Обратите внимание, что мы округлили полученный корень до большего целого числа.

Заметим теперь, что нижний «сосед» на 1 больше N, верхний «сосед» на 1 меньше N, правый «сосед» на $(2 \cdot \text{col})$ больше N, а левый «сосед» на $(2 \cdot \text{col} - 2)$ меньше N. Осталось определить, какие из этих «соседей» присутствуют в таблице:

```
if N != col ** 2 and N != (col - 1) ** 2 + 1:  #если N - не верхнее
                                               #и не нижнее число,
  print(N - (2 * col - 2))                     #у него есть левый сосед
if N != (col - 1) ** 2 + 1:                    #если N - не верхнее
                                               #число,
  print(N - 1)                                 #у него есть верхний
                                               #сосед
if N != col ** 2:                              #если N - не нижнее число,
  print(N + 1)                                 #у него есть нижний сосед
print(N + 2 * col)                             #у любого числа есть
                                               #правый сосед
```

Напомним, что запись «!=» — это операция «не равно».

14. Теннис (турнир II, задача H)

Сначала заметим, что, поскольку в каждой партии участвуют два игрока, сумма чисел x, y и z должна быть четной.

Обозначим через a количество игр Васи с Петей, через b — количество игр Пети с Колей, через c — количество игр Коли с Васей. Тогда:

$$\begin{cases} a + b = y, \\ b + c = z, \\ c + a = x \end{cases} \Leftrightarrow \begin{cases} a = (x + y - z) / 2, \\ b = (y + z - x) / 2, \\ c = (z + x - y) / 2. \end{cases}$$

Заметим, что игр Пети с Васей не может быть слишком много. В лучшем случае они встретятся в первой, третьей, пятой, ... и последней партии, т. е. $a \leqslant (b + c + 1)$. Аналогичные неравенства можно записать для b и c. Переписав их через x, y, z, получим:

$$\begin{cases} 3z + 2 \geqslant x + y, \\ 3x + 2 \geqslant y + z, \\ 3y + 2 \geqslant x + z. \end{cases}$$

В результате мы получаем такую программу:

```
if (x + y + z) % 2 == 0   and  3 * z + 2 >= x + y
and 3 * x + 2 >= y + z    and  3 * y + 2 >= x + z:
  print('YES')
else:
  print('NO')
```

Обратите внимание: при составлении логических условий (например, в операторе `if` или `while`) можно использовать логические операции `and` ("и"), `or` ("или") и `not` ("не").

15. Хорошие стихи (турнир III, задача G)

Рассмотрим две строки s и t и научимся определять, сколько последних символов в них совпадает. Для этого нужно одновременно продвигаться по этим строкам с конца к началу до тех пор, пока одна из них не закончится или пока не встретятся несовпадающие символы:

```
i = 1
while i <= len(s) and i <= len(t) and (s[-i] == t[-i]):
  i += 1
```

Здесь `len(s)` — функция, вычисляющая длину (количество элементов) строки или списка. `s[k]` — это k-й слева элемент строки или списка, если $k \geqslant 0$ (элементы нумеруются с нуля), или $|k|$-й справа элемент списка, если $k < 0$ (элементы нумеруются с минус единицы). То есть `s[0]` — это самый левый элемент списка, а `s[-1]` — самый правый. Оператор же `x += y` представляет собой сокращенную запись оператора `x = x + y` (аналогично расшифровываются сокращенные операторы `-=`, `*=`, `/=`, `//=` и т. д.).

После выполнения указанного фрагмента программы переменная i будет содержать количество равных символов в конце строк s и t. Эту процедуру нужно проделать для первой и третьей строк стихотворения, а затем для второй и четвертой и из двух полученных чисел выбрать большее.

16. Число (турнир II, задача C)

В этой задаче нужно очень четко сформулировать для себя, что именно требуется выводить на экран. Сначала надо вывести одну, две или три цифры (в зависимости от длины числа), затем вывести запятую и три очередные цифры числа, и так до тех пор, пока число не закончится. При этом если количество цифр в числе делится на 3, то первая (левая) группа будет состоять из трех цифр; если количество цифр дает остаток 1 при делении на 3, то первая группа будет состоять из одной цифры; если остаток равен 2 — из двух цифр. Тогда количество цифр в первой группе можно задать следующей формулой (проверьте ее самостоятельно):

```
N = input()
d = (len(N) - 1) % 3 + 1
```

Затем надо вывести первую группу цифр:

```
print(N[:d])
```

(это цифры с номерами от 0 до $d - 1$).

Далее печатаются группы по 3 цифры, каждая из которых предваряется запятой:

```
while d + 3 <= len(N):
  print(',' + N[d:d + 3], end = '')
  d += 3
```

Методический комментарий. Обратите внимание на формулу, по которой мы вычисляли количество цифр в первой группе. Мы сначала уменьшили число на 1, а после взятия остатка увеличили результат на 1. Таким способом мы добились, чтобы вместо числа из диапазона 0, 1, 2... у нас получилось число из диапазона 1, 2, 3... . Этот прием часто используется, когда нужно получить остаток при делении на p в диапазоне от 1 до p (т. е. вместо 0 получить p).

Также следует обратить внимание, что мы печатаем запятую *перед* каждой группой, а не *после* нее, чтобы избежать вывода запятой после последней цифры числа (и именно из-за этого мы добивались, чтобы в первой группе была хотя бы одна цифра, иначе перед числом появилась бы ненужная нам запятая).

17. Трудная задача из ЕГЭ (турнир III, задача A)

Возможно несколько подходов к решению этой задачи. Первый из них — *моделирование*. Будем брать поочередно каждую цифру, проверять, не превосходит ли она 5, при необходимости делить ее на 2 и выводить результат, если получится нечетное число.

Другой подход — заранее рассмотреть все возможности.

Исходная цифра	Что выводить
0	
1	1
2	
3	3
4	
5	5
6	3
7	3
8	
9	

Из этой таблицы видно, что:

- при вводе 1 или 5 нужно вывести ту же самую цифру;
- при вводе 3, 7 или 8 нужно вывести цифру 3;
- в остальных случаях ничего выводить не нужно.

Перейдем теперь к реализации этих идей. Здесь также возможно несколько способов.

Способ 1

Входные данные можно считать как строку, а затем обработать отдельно каждый символ (*c*):

```
s = input()
for c in s:
  обработать c
```

Здесь мы используем цикл `for`, который присваивает переменной *c* последовательно каждый символ из строки *s* (каждый элемент из списка) и выполняет операторы, выделенные отступом. При реализации первого подхода (с моделированием) требуется затем преобразовать символ в число.

```
t = int(c)
if t > 5:
  t = t // 2
if t % 2 == 1:
  print(t)
```

При использовании второго подхода (с таблицей всех возможных вариантов) можно работать сразу с символами:

```
if c in "15":
  print(c)
else:
  if c in "367":
print(3)
```

Здесь `in` — это операция проверки, содержится ли данный символ (элемент) в указанной строке (списке), а `not in` — противоположная операция — проверка, что объект **не** содержится в списке или строке.

Способ 2

Входные данные считываются именно как число. В этом случае потребуется выделить из него отдельные цифры.

Чтобы получить соответствующую цифру, нужно разделить число на такую степень 10, чтобы данная цифра оказалась последней, а затем взять остаток при делении на 10. Например:

```
(8372 // 1000) % 10 = 8
(8372 // 100) % 10 = 3
(8372 // 10) % 10 = 7
(8372 // 1) % 10 = 2
```

Конечно, операцию взятия остатка в первом примере и деление на 1 в последнем примере можно опустить, но ведь нам требуется написать единый цикл для выделения каждой цифры. Получаем программу:

```
n = int(input())
b = 1000
for i in range(4):
  c = (n // b) % 10
  обработка c
  b //= 10 # b = 1000, 100, 10, 1
```

Здесь `range(a,b,c)` — это диапазон целых чисел от a (включительно) до b (не включительно) с шагом c (т. е. a, $a + c$, $a + 2c$, ...). Конструкция `range(a,b)` при этом эквивалентна `range(a,b,1)`, а `range(b)` — это сокращенная запись диапазона `range(0,b,1)`. Цикл же `for` *переменная* `in` *диапазон* последовательно присваивает переменной каждое значение диапазона и выполняет операторы, выделенные отступом.

Далее обработка выделенной цифры производится аналогично решению первым способом.

18. Считалочка (турнир III, задача E)

Составим план решения задачи:

1) подсчитать количество слов в считалочке;

2) подсчитать количество людей;

3) выяснить номер человека, которому выпадет водить;

4) найти в списке имен нужное имя по его номеру.

Теперь перейдем к реализации этого плана.

1) Заметим, что количество слов на 1 больше, чем количество пробелов. Поэтому достаточно пройти по строке и подсчитать в ней количество пробелов:

```
spaces = 0
for c in s1:
  if c == " ":
    spaces += 1
```

2) Аналогично можно подсчитать количество слов N во второй строке.

3) Заметим, что номер человека, на котором заканчивается считалочка, — это остаток от деления количества слов в считалочке на количество людей (если остаток равен 0, то номер человека равен N):

```
number = (spaces + 1) % N;
if number == 0:
  number = N
```

Заметим также, что если нумеровать людей с нуля, то эту формулу можно записать проще:

```
number = spaces % N
```

Поэтому далее мы будем предполагать, что нумерация — именно с нуля.

4) Осталось найти нужное имя — оно будет следовать сразу после пробела с номером *number*:

```
i = 1
k = 0
while k < number:
  if s2[i] == " ":
    k += 1
  i += 1
```

и вывести его:

```
while i < len(s2) and a[i] != " ":
  print(s2[i], end="");
  i += 1
```

А теперь приведем решение этой задачи на языке Python, состоящее всего из одной строки:

```
print(people[input().count(' ') % len(input().split())])
```

Здесь метод `s.count(t)` подсчитывает количество непересекающихся вхождений строки *t* в строку *s*, например:

```
"1212121".count("121") = 2.
```

19. Негласный палиндром (турнир I, задача G)

Будем сравнивать пары символов. Левому из них присвоим номер l, а правому — m. Изначально $l = 0$ (первая буква слова), а $m =$ `len`$(s) - 1$ (последняя буква слова).

Указатель l увеличиваем (двигаем вправо), а указатель m — уменьшаем до тех пор, пока они не укажут на согласные буквы. Когда эти указатели одновременно указывают на согласную букву, производим обмен соответствующих согласных букв и после этого обмена продвигаем указатели к следующим согласным. Процесс повторяем, пока разность значений указателей больше нуля: $m - l > 0$.

Остается сравнить исходное слово с преобразованным и в случае их равенства вывести YES, а в противном случае — NO.

Обратите внимание: удобно определить множество (строку) гласных букв — «aeiouy» и проверять принадлежность той или иной буквы слова этой строке при помощи оператора `in`. Если

буква слова принадлежит этому множеству, то соответствующий указатель можно передвинуть к следующей букве и проверять уже ее на принадлежность этому множеству. Кроме того, вместо строки следует модифицировать список символов, поскольку строка в языке Python является неизменяемым объектом.

20. Прыжки с трамплина (турнир III, задача H)

Решение задачи состоит из двух этапов. Сначала нужно найти минимальную и максимальную оценки и подсчитать сумму без них, а затем вывести результат в указанном формате.

Начнем с цикла, который будет искать минимум, максимум и подсчитывать суммы всех оценок.

```
maxres = 0 # номер максимальной оценки
minres = 0 # номер минимальной оценки
s = a[0]
for i in range(1,5):
  if a[i] < a[minres]:
    minres = i
  if a[i] >= a[maxres]:
    maxres = i;
  s += a[i]
```

Обратите внимание на знаки «<» и «>=» при поиске максимума и минимума. Нам нужно найти самый левый минимум, поэтому минимумы, равные предыдущему, нас уже не интересуют. Что же касается максимума, то здесь нам, наоборот, нужно самое правое число, поэтому мы обновляем максимум, если встречаем равное ему число.

После того как искомые оценки найдены, осталось напечатать их:

```
for i in range(5):
  if i == minres or i == maxres:
    print("(", a[i], ")", end=" " sep=" ")
  else:
    print(a[i], end=" ");
```

и сумму:

```
print("=", s - a[minres] - a[maxres]);
```

21. Будильники (турнир II, задача E)

Эту задачу можно решать разными способами. Но прежде чем переходить к описанию решения, сделаем предварительную подготовку. Вместо того чтобы работать со сложными описаниями моментов времени, лучше переводить все записи времени

в минуты от начала недели. Тогда текущее время в формате «день *d*, *h* часов, *m* минут» превратится в значение переменной:

```
time = (d - 1) * 60 * 60 + h * 60 + m
```

Аналогично преобразуются и значения времени, установленные на будильниках.

Кроме того, мы будем рассматривать не одну, а две подряд идущие недели (на случай, если ближайший будильник прозвенит уже на следующей неделе).

Способ 1

Для каждого будильника вычислим время, когда он прозвенит в следующий раз, и среди этих значений времени найдем ближайшее к текущему.

Рассмотрим для каждого будильника три возможных случая:

1) будильник звенит один раз в неделю в день *d1*, *h1* часов и *m1* минут, и на этой неделе он еще не звенел, — тогда в следующий раз он прозвенит в момент времени

```
time1 = (d1 - 1) * 60 * 60 + h1 * 60 + m1
```

2) будильник звенит один раз в неделю, но на этой неделе он уже прозвенел, — тогда в следующий раз он прозвенит в момент времени

```
time1 + 60 * 60 * 7
```

3) будильник звенит ежедневно. Вычислим время *time1* от начала недели до первого звонка будильника (в понедельник). После этого будильник будет звонить каждые $60 \cdot 60$ минут, т. е. во время вида $time1 + 3600 \cdot K$. Найдем такое минимальное K, при котором это время звонка будет не меньше чем *time*:

$$time1 + 3600 \cdot K \geqslant time,$$
$$K \geqslant (time - time1) \,/\, 3600,$$

т. е. K равно числу $(time - time1) \,/\, 3600$, округленному вверх до ближайшего целого:

```
K = math.ceil((time - time1) / 3600)
```

Таким образом, мы для каждого будильника сможем вычислить время (в минутах от начала недели), когда этот будильник прозвенит в следующий раз. Осталось выбрать из этих значений времени минимальное.

Способ 2

Заведем список (массив), каждая ячейка которого будет соответствовать одной конкретной минуте (первая ячейка — пер-

вой минуте понедельника, вторая — второй минуте понедельника и т. д.). Тогда нам потребуется массив размером $60 \cdot 60 \cdot 7 \cdot 2 = 50400$, чтобы туда уместилась каждая минута двух последовательных недель.

Изначально заполним этот массив нулями. Далее для каждого будильника отметим ячейку массива, соответствующую времени, когда этот будильник прозвенит. При этом для еженедельных будильников будут отмечены две клетки, так как мы рассматриваем сразу две недели, а для ежедневных будильников — 14 клеток.

После этого начнем просмотр массива с клетки, соответствующей текущему времени, и будем идти по массиву вправо, пока не встретим первую отмеченную клетку. Она и будет соответствовать времени, когда впервые прозвенит будильник.

Методический комментарий. В решении этой задачи можно выделить множество методически значимых идей:

1) вместо того, чтобы работать со сложными по структуре данными (день, час, минута), мы переводим все в минуты, чем облегчаем как хранение информации, так и ее обработку (например, сравнение моментов времени);

2) мы используем промежуток в две недели вместо одной, чтобы не рассматривать отдельно случаи «перехода через границу недели»;

3) второй способ демонстрирует важный (не только в задачах, связанных со временем) прием — массив отметок. Если допустимое множество значений (в данном случае — все значения времени в минутах) невелико и соответствующий массив умещается в памяти, то можно помечать в таком массиве встречающиеся события (или сохранять о них какую-то другую информацию). Подобный прием, в частности, лежит в основе алгоритма сортировки подсчетом.

22. Магический квадрат (турнир III, задача I)

В этой задаче достаточно аккуратно реализовать описанный в условии алгоритм. Будем по очереди расставлять числа от 1 до N^2 в клетки с координатами (x, y) двумерного массива a (изначально по условию $x = (n + 1) // 2$, $y = 1$; массив заполнен нулями).

```
for i in range(1, N * N + 1):
  a[x][y] = i
    # Затем запомним эти координаты
  oldx, oldy = x, y
  # и вычислим координаты следующей клетки
```

```
    y, x = y - 1, x + 1
  # Проверим, не вышли ли мы за пределы квадрата
  if x == n + 1:
    x = 1
  if y == 0:
    y = n
    # После этого проверим, не были ли мы в этой клетке ранее
  if a[x][y] != 0:
    # спускаемся на одну клетку вниз от начальной:
    # здесь нам и пригодились запомненные координаты
      x, y = oldx, oldy + 1
        # и снова проверяем, не вышли ли мы за границы квадрата
        # в этом случае
  if y == n + 1:
    y = 1
```

Обратите внимание: в языке Python создать *одномерный* список размера *n* и заполнить его нулями можно так:

```
a = [0] * n
```

Двумерный список представляет собой список, каждый элемент которого также является списком (одномерным). Двумерный список нулей в Python можно инициализировать так:

```
a = [[0] * n for i in range(n)]
```

23. Шахматное домино (турнир I, задача F)

Прежде всего, нужно прочитать входные данные в двумерный массив (на Python — список списков):

```
a = []
for i in range(8):
  a.append(list(map(int,input().split())))
```

Теперь составим схему решения задачи.

Для каждой строки нам нужно однозначно определить цвет клеток, которые должны на ней оказаться. Этот цвет (закодированный числами 1 или –1) будет храниться в массиве (списке) *color*, который изначально заполнен нулями:

```
color = [0] * 8
```

Для определения цвета клеток необходимо сделать следующее.

1) Надо проверить каждую горизонтальную доминошку. Если она состоит из двух квадратов разного цвета, то можно вывести NO и сразу завершить выполнение программы. В противном случае мы определили цвет данной строки (по цвету этой доминошки). Если же этот цвет уже был определен раньше и не совпадает с цветом новой доминошки, то опять-таки можно вывести NO и завершить работу программы.

```
for i in range(8): #для каждой строки
  for j in range(7): #рассматриваем клетку и ее левого соседа
    if a[i][j] == -a[i][j+1]:          #если это разноцветные клетки
      print('NO')                      #одной доминошки
      sys.exit()
    if a[i][j] == a[i][j+1]:           #если одноцветные:
      if (a[i][j]>0 and color[i]>=0):  #цвет совпадает
        color[i] = 1                   #с цветом строки,
      if a[i][j]>0 and color[i] >=0:   #или цвет строки
        color[i] = -1                  #еще не определен
    if a[i][j] * color[i] < 0:         #цвет не совпадает
        print('NO')                    #с цветом строки
          sys.exit()
```

Функция `exit` из модуля `sys` позволяет завершить выполнение программы в момент ее вызова.

2) Надо просмотреть все вертикальные одноцветные доминошки и убедиться, что их цвет не противоречит цвету строки, а если цвет строки еще не определен, то определить его согласно расположению этих доминошек.

```
for i in range(7): #для каждой строки
  for j in range(8): #рассматриваем клетку и ее нижнего соседа
    if a[i][j] == a[i+1][j]: #если они одного цвета
      #и этот цвет совпадает с цветом обеих строк
      #либо цвет какой-то из строк еще не определен
      if color[i]*a[i][j] >= 0 and color[i+1]*a[i+1][j]>=0:
        color[i] = a[i][j]
        color[i+1] = a[i+1][j]
      else:
        print('NO')
        sys.exit()
```

3) И наконец, надо рассмотреть вертикальные доминошки, состоящие из клеток разного цвета. При этом возможны три ситуации:

а) цвета обеих строк, которые занимает такая доминошка, уже определены, — тогда достаточно убедиться, что эти цвета различны, а в противном случае вывести NO и завершить работу программы;

б) цвет первой строки, которую занимает такая доминошка, определен, а цвет второй — нет, — в этом случае нужно определить цвет второй строки как противоположный цвету первой;

в) цвет первой строки (верхней из двух, которые занимает доминошка), не определен. Это означает, что в данной строке нет ни горизонтальных, ни одноцветных вертикальных доминошек, ни даже доминошек разного цвета, уходящих вверх (иначе бы мы определили цвет данной строки, рассматривая такую доминошку в предыдущей строке). Это также означает, что вся

данная строка занята двухцветными вертикальными доминошками, уходящими вниз, а это значит, что мы можем перевернуть их так, чтобы данная строка была одного цвета, а следующая — противоположного.

Особо отметим, что такую проверку мы можем выполнить только один раз для первой клетки данной строки, и если условия этого пункта выполняются, то надо переходить сразу к строке через одну вниз от данной.

```
i = 0
while i < 7:
  if color[i] == 0 and a[i][0] == -a[i+1][0]:
    i += 2
  else:
    for j in range(8):
      if a[i][j] == -a[i+1][j]:
        if color[i] != color[i+1]:
           color[i+1] = -color[i]
        else:
           print('NO')
           sys.exit()
    i += 1
```

В случае же, если ни одна из вышеприведенных проверок не прервала программу с ответом NO, нам остается вывести ответ YES.

Обратите внимание на некоторые технические детали реализации описанного выше алгоритма.

Во-первых, в последней части мы использовали цикл `while` вместо `for`, чтобы иметь возможность пропускать строки, если в двух соседних строках все доминошки двухцветные и вертикальные.

Во-вторых, мы использовали условия вида

```
if color[i] * a[i][j] >= 0
```

вместо равносильного, но более длинного

```
if (color[i] >= 0 and a[i][j] > 0) or
         (color[i] <= 0 and a[i][j] < 0)
```

воспользовавшись тем, что произведение отрицательно только тогда, когда сомножители имеют разные знаки.

24. Деление с остатком (турнир III, задача D)

Способ 1 (перебор)

Возьмем какое-нибудь число, которое меньше, чем максимально возможное число в ответе, например 48 000 (при его делении на 2, а затем на 3 и на 4 получится 2000, а согласно ограничениям в условии, K не превосходит 1000). Остается перебирать

все возможные ответы (от 1 до 48 000) по возрастанию и выводить на экран подходящие:

```
for i in range(1,48001):
  if i // 2 // 3 // 4 == K:
    print(i, end = " ")
```

Это решение верное, но, увы, не самое эффективное.

Способ 2 (вычисление ответа)

Ясно, что Вася мог изначально взять одно из нескольких подряд идущих натуральных чисел. Поэтому достаточно определить минимальное и максимальное такое число.

Начнем с минимального. Очевидно, что самое маленькое число должно каждый раз делиться без остатка, поэтому это число равно $2 \cdot 3 \cdot 4 \cdot K = 24 \cdot K$.

Теперь, вместо того чтобы искать максимальное число, найдем следующее за ним число, т. е. минимальное число, которое в результате проделанных операций даст уже ответ $K + 1$. Аналогично предыдущему рассуждению, это число равно $24 \cdot (K + 1)$.

Вывод: Вася мог взять любое число от $24 \cdot K$ до $24 \cdot (K + 1) - 1 = 24 \cdot K + 23$. Осталось написать программу, печатающую эти числа через пробел:

```
for i in range(24 * K, 24 * K + 23 + 1):
    print(i, end = " ")
```

25. Разложение на простые множители (турнир III, задача B)

Эта задача относится к разделу, называемому *комбинаторикой*.

Для этой задачи возможно два решения: одно — чисто математическое, а другое — алгоритмическое. Причем для обоих этих решений потребуется предварительная подготовка — нужно разложить число на множители. Причем в математическом решении нам будет важно, в какой степени входит каждый простой сомножитель, а для алгоритмического решения потребуется полное разложение числа.

Для разложения числа N на простые множители достаточно делить его сначала на 2 (столько раз, сколько оно делится без остатка), затем на 3 и т. д., пока не получится единица:

```
d = 2 # число, на которое будем делить
while n > 1:
  if n mod d == 0:
    n //= d
  else:
    d += 1
```

Это конечно же общий алгоритм, который нужно будет модифицировать для каждого из решений.

Способ 1 (математический)

Заметим, что если бы у числа N было K простых делителей и каждый из них входил бы в разложение в первой степени, то количество возможных записей было бы равно $1 \cdot 2 \cdot \ldots \cdot K = K!$ (факториал числа K). Действительно, при записи делителей мы на первое место могли бы поставить любой из K делителей, на второе — любой из $(K - 1)$ оставшихся и т. д., а на последнем месте тогда окажется единственный не использованный ранее делитель.

Пусть теперь один из делителей p входит в разложение дважды. Представим себе, что эти два делителя не равны, и запишем $K!$ разложений. Но после этого мы обнаружим, что все такие разложения разбиваются на пары, в которых наши два делителя стоят на одних и тех же местах, но в обратном порядке. Например:

$$12 = 2_1 \cdot 2_2 \cdot 3 = 2_2 \cdot 2_1 \cdot 3 = 2_1 \cdot 3 \cdot 2_2 = 2_2 \cdot 3 \cdot 2_1 = 3 \cdot 2_1 \cdot 2_2 = 3 \cdot 2_2 \cdot 2_1.$$

Поэтому, чтобы получить количество *различных* записей, нужно разделить $K!$ на 2. Аналогично, если бы какой-то делитель входил в разложение в 3-й степени, то каждая запись имела бы еще 5 «двойников» — ведь переставить 3 одинаковых делителя можно $3! = 6$ способами. И вообще, если делитель входит в степени m, то $K!$ придется делить на $m!$.

Если же сразу несколько делителей входят в разложение в степени выше 1, то для каждого из них нужно будет разделить $K!$ на соответствующий факториал. Например, $72 = 2^3 \cdot 3^2$. Общее количество делителей равно 5, а искомое количество записей равно $5! / (3! \cdot 2!)$, поскольку один из делителей входит в разложение в степени 3, а другой — в степени 2.

Написание реализующей данную формулу программы мы оставляем читателям в качестве самостоятельного упражнения.

Способ 2 (перебор)

Вместо того чтобы вычислять количество записей, можно попытаться перебрать и пересчитать их все (благо их не очень много — самостоятельно оцените наибольший возможный ответ в задаче). Программа при этом получится менее эффективной и более сложной, поэтому подробное ее описание мы здесь приводить не будем.

26. Вася (турнир I, задача D)

Для объяснения решения воспользуемся *кругами Эйлера* — диаграммой, демонстрирующей множества задач на циклы (круг A), задач на массивы (круг B), задач на строки (круг C) и их взаимные пересечения:

$$D = A \cap B, \quad E = A \cap C, \quad F = B \cap C, \quad G = A \cap B \cap C.$$

Поскольку множество всех задач есть объединение множеств A, B и C, то его мощность равна:

$$|A| + |B| + |C| - (|A \cap B| + |A \cap C| + |B \cap C|) + |A \cap B \cap C|.$$

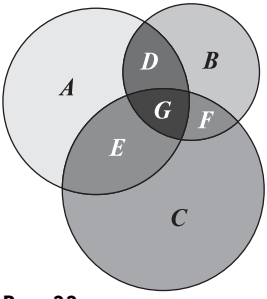


Рис. 33

Эта формула является частным случаем *формулы включения-исключения* для трех множеств. Ее легко доказать, глядя на приведенную диаграмму (рис. 33).

Подставляя в эту формулу ранее введенные обозначения, получим ответ:

$$A + B + C - D - E - F + G.$$

Программное решение задачи сводится к выводу ответа, вычисленного по этой формуле.

■ ОЛИМПИАДЫ, ПРОВОДИМЫЕ ВУЗАМИ. МОСКОВСКИЙ ГОРОДСКОЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ. МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

2005 год

1. Система счисления

В системе счисления с основанием p число 4840 можно записать в виде

$$4p^3 + 8p^2 + 4p$$

(так называемая «развернутая форма записи числа»). Преобразуем это выражение:

$$4840 = 4p^3 + 8p^2 + 4p = 4p(p^2 + 2p + 1) = 4p(p + 1)^2.$$

По условию число 4840 делится на простое число 11. Значит, на 11 должно делиться произведение $4p(p + 1)^2$. Тогда либо p, либо $p + 1$ делится на 11, т. е. в случае $p < 11$ нам подходит значение $p = 10$, а для случая $p < 31$ подходят значения p, равные 10, 11, 21 или 22.

Критерии оценивания

Получен правильный обоснованный ответ для обоих вариантов — 8—10 баллов.

Получен правильный обоснованный ответ для обоих вариантов, но во втором варианте найдены не все решения — 6—7 баллов.

Получен правильный ответ только на второй вариант — 4—5 баллов.

Получен правильный ответ только на первый вариант — 1—3 балла.

2. Терминология

Критерии оценивания

Ответ учащегося, записанный в произвольной форме, содержит корректное определение цилиндра как совокупности дорожек магнитного диска, имеющих один и тот же порядковый номер, — 1—3 балла.

В ответе указаны основные параметры, характеризующие цилиндр (объем памяти, количество сторон или дорожек, порядковый номер, количество секторов и др.), — 1—3 балла.

Дан развернутый ответ, содержащий дополнительные сведения, определения, рисунки и т. п., — 1—2 балла.

Корректность, грамотность и логичность изложения — 1—2 балла.

При определении итогового балла за выполнение данного задания баллы, выставленные по вышеотмеченным критериям, суммируются.

3. Логическая задача

Оптимальным является решение, согласно которому семья перейдет мост за минимально возможное время, — 17 минут. В этом случае последовательность действий членов семьи должна быть следующей:

Этап	Длительность
1. Переходят папа и мама	2 минуты
2. Папа с фонариком возвращается	1 минута
3. Переходят бабушка и малыш	10 минут
4. Мама с фонариком возвращается	2 минуты
5. Переходят папа и мама	2 минуты
Итого	**17 минут**

Критерии оценивания

Получен правильный обоснованный ответ — 10 баллов.

Получен обоснованный ответ, согласно которому требуемое время равно или превышает 19 минут, — 4—6 баллов.

Получен правильный ответ (17 минут), но не указана или указана неправильно последовательность действий членов семьи — 1—3 балла.

4. Программирование

Ответ: $c = 8$, $d = 3$. Задача решается путем пошагового выполнения записанных в программе команд с одновременным построением таблицы текущих значений переменных («трассировка» переменных).

Критерии оценивания

Получен правильный ответ — 5 баллов.

Правильно найдено только одно из значений — 2 балла.

5. Последовательности

Основные этапы решения задачи:

1) ввод номера цифры k;

2) определение номера пары цифр, в которую входит k-я цифра;

3) нахождение двузначного числа, образованного этой парой цифр;

4) определение искомой цифры;

5) вывод ответа.

Программа на школьном алгоритмическом языке:

```
алг Задача5
нач цел k, номер_пары, число2, цифра
| |1. Ввод номера цифры k
| вывод "Задайте номер цифры k"
| ввод k
| |2. Определение номера пары цифр, в которую входит k-я цифра
| номер_пары := div((k + 1), 2)
| |3. Нахождение двузначного числа, образованного этой парой цифр
| число2 := 10 + номер_пары - 1
| |4. Определение искомой цифры
| если mod(k, 2) = 1
| |то
| | цифра := div(число2, 10)
| |иначе
| | цифра := mod(число2, 10)
| все
| |5. Вывод ответа
| вывод нс, "Искомая цифра равна", цифра
кон
```

Критерии оценивания

Разработана программа, решающая поставленную задачу во всех случаях, — 14—15 баллов.

Разработана программа, решающая поставленную задачу не во всех случаях, — 12—13 баллов.

Составлен только корректный алгоритм решения задачи — 8—11 баллов.

6. Три приятеля

Да, мог. Действительно, из ответа Андрея ясно, что у Бориса и Вадима не может быть двух белых шапок (иначе Андрей сразу определил бы, что на нем — черная шапка, ведь в мешке было только две белые шапки, а остальные — черные). Значит, у Бориса и Вадима либо одна белая и одна черная шапка, либо обе шапки черные.

Если при этом у Вадима была бы белая шапка, то Борис (услышав ответ Андрея) легко определил бы цвет своей шапки (черный), а так как он этого сделать не смог, то Вадим должен понять, что его шапка — черная.

Критерии оценивания

Получен обоснованный ответ «Мог и должен сказать, что у него черная шапка» — 15—20 баллов.

Получен обоснованный ответ «Мог», но цвет шапки не определен — 11—14 баллов.

Получен правильный ответ, но ответ не обоснован — 1—3 балла.

7. Пересадки

Пример правильной программы на языке Pascal (в данном случае ввод/вывод производится из файла `input.txt` и в файл `output.txt`):

```
const inf='input.txt';
      ouf='output.txt';
const nmax=100;
var   n:integer;
      a:array[1..nmax] of integer;
      b:integer;
      res:integer;
procedure init;
  begin
  assign(input,inf);
  reset(input);
  assign(output,ouf);
  rewrite(output);
```

```
  end;
procedure readdata;
  var i:integer;
  begin
  read(n);
  for i:=1 to n do
    read(a[i]);
  read(b);
  end;
function find(q:integer):integer;
  var i:integer;
  begin
  for i:=1 to n do
    if a[i]=q then begin
      find:=i;
      exit;
      end;
  runerror(239); {Некорректные данные}
  end;
procedure work;
  var q:integer;
  begin
  res:=0;
  q:=find(b);
  if q=b then exit;
  a[q]:=0;
  res:=1;
  while a[b]<>0 do begin
    b:=a[b];
    inc(res);
    end;
  end;
procedure done;
  begin
  writeln(res);
  close(output);
  close(input);
  end;

begin
  init;
  readdata;
  work;
  done;
end.
```

Критерии оценивания

Разработана программа, решающая поставленную задачу во всех случаях, — 25—30 баллов.

Разработана программа, решающая поставленную задачу не во всех случаях, — 20—24 балла.

2006 год

1. Системы счисления

Так как все числа даны в различных системах счисления, для их сравнения необходимо вначале перевести их в одну систему счисления, например в десятичную:

$11101_2 = 29_{10}$;
$123_4 = 27_{10}$;
$34_8 = 28_{10}$;
$20_{16} = 32_{10}$.

Тогда запись заданных чисел в порядке возрастания: 123_4, 34_8, 11101_2, 20_{16}.

Критерии оценивания

Получен правильный обоснованный ответ — 5 баллов.

Получен обоснованный ответ, но допущена одна ошибка при записи чисел в порядке возрастания — 3 балла.

2. Единицы измерения информации

Решение:

$$\begin{cases} 2^{x+2}\text{ (бит)} = 8^{y-5}\text{ (Кбайт)}, \\ 2^{2y-1}\text{ (Мбайт)} = 16^{x-3}\text{ (бит)}; \end{cases}$$

$$\begin{cases} 2^{x+2}\text{ (бит)} = 2^{3(y-5)} \cdot 2^{10} \cdot 2^3\text{ (бит)}, \\ 2^{2y-1} \cdot 2^{20} \cdot 2^3\text{ (бит)} = 2^{4(x-3)}\text{ (бит)}; \end{cases}$$

$$\begin{cases} 2^{x+2}\text{ (бит)} = 2^{3(y-5)+10+3}\text{ (бит)}, \\ 2^{(2y-1)+20+3}\text{ (бит)} = 2^{4(x-3)}\text{ (бит)}; \end{cases}$$

$$\begin{cases} x+2 = 3(y-5)+13, \\ 2y+22 = 4(x-3); \end{cases} \qquad \begin{cases} x+2 = 3(y-5)+13, \\ 2y+22 = 4(x-3); \end{cases}$$

$$\begin{cases} x = 3y-4, \\ y = 2x-17; \end{cases} \qquad \begin{cases} y = 2(3y-4)-17, \\ y = 5; \end{cases} \qquad \begin{cases} y = 5, \\ x = 11. \end{cases}$$

Ответ: $x = 11$, $y = 5$.

Критерии оценивания

Мегабайты и килобайты правильно переведены в биты. Уравнения выражены через степень с основанием, равным 2. Приведение системы показательных уравнений к системе двух уравнений с двумя переменными. Решение системы одним из методов и получение правильного решения — 8—10 баллов.

Мегабайты и килобайты правильно переведены в биты. Уравнения выражены через степень с основанием, равным 2.

Приведение системы показательных уравнений к системе двух уравнений с двумя переменными — 6—7 баллов.

Мегабайты и килобайты правильно переведены в биты. Уравнения выражены через степень с основанием, равным 2, — 4—5 баллов.

Мегабайты и килобайты правильно переведены в биты — 1—3 балла.

3. Терминология

Системное программное обеспечение (*ПО*) обеспечивает целостное функционирование компьютера. Оно необходимо для корректного функционирования прикладного программного обеспечения, для управления режимами работы и для настройки аппаратного обеспечения.

К системному ПО относятся:

- операционная система (ОС) — комплекс программ, управляющих устройствами компьютера, файловой системой, диалогом с пользователем;
- сервисное ПО — программы обслуживания дисков, контролирующие и диагностирующие программы, архиваторы, антивирусы и т. д.

Прикладное программное обеспечение предназначено для непосредственного решения пользовательских (прикладных) задач.

К прикладному ПО относятся:

- прикладные программы общего назначения — текстовые, графические, звуковые редакторы и процессоры, СУБД, табличные процессоры, игры и пр.;
- прикладные программы специального назначения — математические пакеты, САПР, бухгалтерские пакеты, экспертные системы, педагогические программные средства и др.

Общие черты системного и прикладного программного обеспечения: все эти программы позволяют использовать компьютер для решения разнообразных задач.

Критерии оценивания

Ответ, записанный в произвольной форме, содержит корректное определение и общие черты системного и прикладного программного обеспечения — 1—3 балла.

В ответе перечислены программы, входящие в состав системного и прикладного программного обеспечения, — 1—3 балла.

Дан развернутый ответ, содержащий дополнительные сведения, определения, примеры, рисунки и т. п., — 1—2 балла.

Корректность, грамотность и логичность изложения — 1—2 балла.

При определении итогового балла за выполнение данного задания баллы, выставленные по вышеотмеченным критериям, суммируются.

4. Программирование

Можно вычислить $x = \texttt{div}(n, m) \cdot (n - m) + 1$, где `div` — операция целочисленного деления.

Обоснование: при $n < m$ получаем $\texttt{div}(n, m) = 0$, а при $n = m$ получаем $n - m = 0$. Тогда, если $n \leqslant m$, то в результате вычислений по приведенной выше формуле значение переменной x будет равно 1, в противном случае получается другое число.

Другие возможные правильные решения:

а) $x = (n \texttt{ div } (m + 1)) + 1$;

б) $x = n - m + \texttt{abs}(n - m) + 1$, где `abs` — определение абсолютного значения (модуля) числа.

Критерии оценивания

Разработан фрагмент программы, решающий поставленную задачу во всех случаях, — 14—15 баллов.

Разработан фрагмент программы, решающий поставленную задачу не во всех случаях, — 6—10 баллов.

5. Последовательности

Основные этапы решения задачи:

1) определение номера тройки цифр, в которую входит заданная k-я цифра; этот номер (n) может быть найден, например, так:

$$n = (k + 2) \texttt{ div } 3,$$

где `div` — операция целочисленного деления;

2) нахождение трехзначного числа m, образованного соответствующей тройкой цифр:

$$m = 100 + n;$$

3) определение в найденном трехзначном числе m искомой цифры.

На школьном алгоритмическом языке соответствующий фрагмент программы имеет вид:

```
если mod(k, 3) = 0 |число k кратно трем
| то |искомая цифра x равна последней цифре числа m
| x := mod(m, 10)
| иначе
| если mod(k, 3) = 1 |k – одно из чисел 1, 4, 7, ...
| | то
| | x := 1 |первая цифра каждой "тройки" – 1
| | иначе |k – одно из чисел 2, 5, 8, ...,
| | |т. е. искомая цифра равна средней цифре числа m
| | x := mod(div(m, 10), 10)
| все
все
```

Критерии оценивания

Разработана программа, решающая поставленную задачу во всех случаях, — 20—25 баллов.

Разработана программа, решающая поставленную задачу не во всех случаях, — 15—19 баллов.

Составлен только корректный алгоритм решения задачи — 5—10 баллов.

6. Электронный замок

Текст программы на языке Pascal, решающей поставленную задачу:

```
program Key;
var n, i, sum, dif, invNum: Integer;
    str: String;
begin
  readLn(n);
  i:=0;
  sum:=0;
  str:=' ';
  while sum < n do
  begin
    i:=i+1;
    str:=str + '+';
    sum:=sum+i;
  end;
  dif:=sum-n;
  if dif <> 0 then
  begin
    invNum:= dif div 2;
    if invNum>0 then str[invNum]:='-';
    sum:=sum-2*invNum;
    if (dif mod 2) > 0 then begin
      str:=str+'+-';
      i:=i+2;
      sum:=sum-1;
```

```
      end;
    end;
    writeLn(i);
    writeLn(str);
end.
```

Критерии оценивания

Разработана программа, решающая поставленную задачу во всех случаях, — 30—35 баллов.

Разработана программа, решающая поставленную задачу не во всех случаях, — 20—25 баллов.

Составлен только корректный алгоритм решения задачи — 15—19 баллов.

2007 год

1. Системы счисления

Обозначим основание искомой системы счисления переменной p. Тогда, зная, что в десятичной системе p-ичному числу 14 будет соответствовать значение $p + 4$, а p-ичному числу 11 — значение $p + 1$, можно записать:

$$(p + 4)^2 - (p + 1)^2 = 69.$$

Решив это уравнение, получим: $p = 9$.

Ответ: в девятеричной системе счисления.

Критерии оценивания

Получен правильный обоснованный ответ, решение грамотно оформлено — 5 баллов.

Получен правильный обоснованный ответ — 4 балла.

Получен обоснованный ответ, но допущена арифметическая ошибка в вычислениях — 3 балла.

Правильный ответ получен путем подбора — 2 балла.

2. Единицы измерения информации

$65\,536 = 2^{16}$, тогда для кодирования информации о цвете одной точки требуется 2 байта.

14 400 бит/с = 1800 байт/с.

За 3 минуты (180 с) можно передать $180 \cdot 1800 = 324\,000$ байт, т. е. информацию о 162 000 точках изображения. Поскольку высота изображения составляет 400 точек, его ширина определяется как 162 000 / 400 = 405 точек.

Ответ: 405 точек.

Критерии оценивания

Получен правильный обоснованный ответ, решение грамотно оформлено — 5 баллов.

Получен правильный обоснованный ответ — 4 балла.

Получен обоснованный ответ, но допущена арифметическая ошибка в вычислениях — 2—3 балла.

3. Високосный год

Один из возможных вариантов ответа:

(N mod 400 = 0) OR (N mod 4 = 0) AND (N mod 100 ≠ № 0)

Критерии оценивания

Получен правильный ответ — 5 баллов.

4. Терминология

Вариант ответа 1

Шина — физическая совокупность проводов, предназначенная для передачи различных сигналов.

Системная шина (*магистраль*) — набор электронных линий, связывающих воедино по адресации памяти, передаче данных и служебных сигналов процессор, память и периферийные устройства. *Системная шина* состоит из шин данных, адреса и управления.

Вариант ответа 2

Системная шина (*магистраль*) — набор электронных линий, связывающих воедино по адресации памяти, передаче данных и служебных сигналов процессор, память и периферийные устройства. Обмен информацией между отдельными устройствами ЭВМ производится по трем многоразрядным шинам, соединяющим все модули: *шине данных*, *шине адресов* и *шине управления*. Данные по шине данных могут передаваться как от процессора к какому-либо устройству, так и в обратную сторону, т. е. шина данных является двунаправленной. По *шине адреса* передается код адреса данного устройства или ячейки памяти. По *шине управления* передаются сигналы, определяющие характер обмена информацией, и сигналы, синхронизирующие взаимодействие устройств, участвующих в обмене информацией.

Вариант ответа 3

Системная шина (*магистраль*) — набор электронных линий, связывающих воедино по адресации памяти, передаче данных и служебных сигналов процессор, память и периферийные устройства. Обмен информацией между отдельными устройствами ЭВМ производится по трем многоразрядным шинам, соеди-

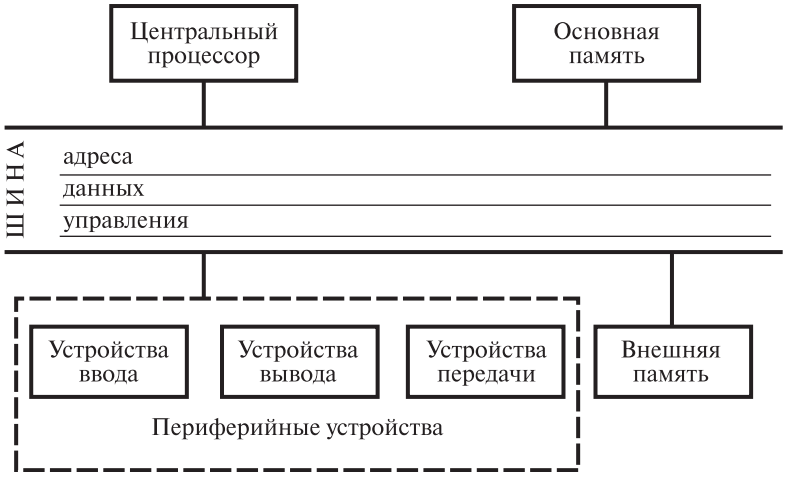


Рис. 34

няющим все модули: *шине данных*, *шине адресов* и *шине управления* (рис. 34).

Разрядность *шины данных* задается разрядностью процессора, т. е. количеством двоичных разрядов, которые процессор обрабатывает за один такт. Данные по шине данных могут передаваться как от процессора к какому-либо устройству, так и в обратную сторону, т. е. шина данных является двунаправленной. К основным режимам работы процессора с использованием шины передачи данных можно отнести следующие: запись/чтение данных из оперативной памяти и из внешних запоминающих устройств, чтение данных с устройств ввода, пересылку данных на устройства вывода. Выбор абонента по обмену данными производит процессор, который формирует код адреса требуемого устройства, а для ОЗУ — код адреса ячейки памяти. Код адреса передается по *адресной шине* в одном направлении — от процессора к устройствам, т. е. эта шина является однонаправленной. По *шине управления* передаются сигналы, определяющие характер обмена информацией, и сигналы, синхронизирующие взаимодействие устройств, участвующих в обмене информацией.

Может быть указано дополнительно

USB (*Universal Serial Bus — универсальная последовательная шина*) — обеспечивает высокоскоростное подключение к компьютеру сразу нескольких периферийных устройств (сканер, цифровая фотокамера, внешний накопитель и т. д.).

Критерии оценивания

Ответ учащегося, записанный в произвольной форме, содержит корректное определение шины — 1—3 балла.

В ответе указаны основные виды шин и характеризующие их параметры — 1—3 балла.

Дан развернутый ответ, содержащий дополнительные сведения, определения, рисунки и т. п., — 1—2 балла.

Корректность, грамотность и логичность изложения — 1—2 балла.

При определении итогового балла за выполнение задания баллы, выставленные по вышеотмеченным критериям, суммируются.

5. Программирование. Работа с текстом

Программа на школьном алгоритмическом языке:

```
алг Количество_слов_в_предложении
нач лит предложение, цел количество_слов, всего_символов, i
| вывод нс, "Введите предложение"
| ввод предложение
| количество_слов:=0
| всего_символов:=длин(предложение)
| нц для i от 2 до всего_символов
| | если предложение[i]=" " и предложение[i-1]<>" "
| | |то
| | | количество_слов:=количество_слов+1
| | все
| кц
| |При необходимости учитываем последнее слово
| если предложение[всего_символов]<>" "
| |то
| | количество_слов:=количество_слов+1
| все
| вывод нс, "Количество слов во введенном предложении", количество_слов
кон
```

Критерии оценивания

Разработана рациональная, грамотно оформленная программа, решающая поставленную задачу во всех случаях, — 10 баллов.

Разработана рациональная программа, решающая поставленную задачу во всех случаях, — 7—9 баллов.

Разработана программа, решающая поставленную задачу во всех случаях, — 5—6 баллов.

Разработана программа, решающая поставленную задачу не во всех случаях, — 1—4 балла.

6. Программирование. Работа с массивами

Программа на школьном алгоритмическом языке:

```
цел n
n:=8
алг Количество_различных_чисел_в массиве
нач цел таб массив[1:n], массив_неповт[1:n], цел кол_неповт, i,j,
лог есть
| |Заполнение исходного массива
| нц для i от 1 до n
| | массив[i]:=rnd(5)
| кц
| |Первое число – всегда оригинальное
| кол_неповт:=1
| массив_неповт[кол_неповт]:=массив[1]
| нц для i от 2 до n
| | |Проверяем, имеется ли i-й элемент в массиве массив_неповт
| | есть:=нет; j:=1
| | нц пока j<=кол_неповт и не есть
| | | если массив[i] = массив_неповт[j]
| | | |то
| | | | есть:= да
| | | |иначе
| | | | j:=j+1
| | | все
| | кц
| | если не есть
| | |то
| | | |встретилось очередное оригинальное число
| | | |Записываем его в массив массив_неповт
| | | кол_неповт:=кол_неповт+ 1
| | | массив_неповт[кол_неповт]:=массив[i]
| | все
| кц
| |Вывод исходного массива
| нц для i от 1 до n
| | вывод массив[i]
| кц
| |Вывод ответа
| вывод нс, "Количество различных чисел в заданном массиве:",
   кол_неповт
кон
```

Критерии оценивания

Разработана рациональная, грамотно оформленная программа, решающая поставленную задачу во всех случаях, — 15 баллов.

Разработана рациональная программа, решающая поставленную задачу во всех случаях, — 7—9 баллов.

Разработана программа, решающая поставленную задачу во всех случаях, — 5—6 баллов.

Разработана программа, решающая поставленную задачу не во всех случаях, — 1—4 балла.

7. Сравнение чисел

Критерии оценивания

Суммируются баллы, выставляемые за правильную работу программы на следующей системе входных данных.

Номер	Входные данные	Выходные данные	Баллы
1	239 361	<	1
2	777 6666	<	1
3	239.000 239	=	1
4	361.010 361.01	=	1
5	0.239239 0.239	>	1
6	239.361 −777.777	>	1
7	−3424.2424 453.01	<	1
8	−564352.01 −4642452.02	>	1
9	−1.00001 −1.0001	>	1
10	−10.0 −1.00	<	1
11	6229415378308207895471406199253644614572.917229 6229415378308207895471406199253644614572.91723	<	2
12	70756406344595901367458264854016802627891279.654 70756406344595901367458268864299481232186114.004	<	2
13	636316889.24427643302694786136448652606721377 636316889.2442764330269478613644865260 67	>	2
14	241.048723967889105528112526032451038471 63 241.047372004261177958353595182660608788 36	>	2
15	1111111111111111111111111111111111111111.1111111 1111111111111111111111111111111111111111.11111111	>	2

8. Электронные часы

Критерии оценивания

Суммируются баллы, выставляемые за правильную работу программы на следующей системе входных данных.

Номер	Входные данные	Выходные данные	Баллы
1	23:59:58 23:59:59	0 0 2 2 0 4 0 0 1 3	2
2	03:54:10 04:04:00	1043 219 219 569 451 569 119 119 119 119	1
3	00:00:00 23:59:59	92880 92880 71280 56880 53280 53280 24480 24480 24480 24480	2
4	12:00:00 12:00:59	136 76 76 16 16 16 6 6 6 6	1
5	11:00:59 11:01:00	5 5 0 0 0 1 0 0 0 1	1
6	19:59:59 21:00:17	5568 1949 5540 1922 1922 1924 722 722 721 724	1
7	02:57:30 02:57:30	2 0 1 1 0 1 0 1 0 0	2
8	06:55:39 06:55:39	1 0 0 1 0 2 1 0 0 1	1
9	10:53:47 10:53:47	1 1 0 1 1 1 0 1 0 0	1
10	21:50:51 21:50:51	1 2 1 0 0 2 0 0 0 0	1
11	17:12:03 19:46:26	4573 14317 5230 5227 5014 4627 1854 4703 5426 4613	2
12	07:58:42 23:11:37	44205 72403 47766 33364 32666 32746 14538 14616 18155 18197	1
13	06:58:16 10:46:15	21108 10204 7428 7428 7204 6932 2808 6288 6332 6348	1
14	04:42:11 22:36:51	60677 77618 49691 37890 39016 41670 20120 20068 20068 20068	1
15	08:41:37 18:55:21	27786 55228 23134 23135 23645 23426 10942 10943 11768 10943	1

Окончание табл.

Номер	Входные данные	Выходные данные	Баллы
16	05:46:06 14:10:09	35312 34484 19464 19464 16708 17298 9679 9685 9685 9685	1
17	06:39:33 15:10:11	32273 38250 19623 19658 20234 17246 7351 9724 9724 9751	1
18	13:42:22 23:55:13	23067 45719 37218 23900 23585 23395 10937 10937 10937 10937	1
19	13:06:05 16:51:16	7021 20857 7321 10556 10921 10399 5804 2731 2731 2731	1
20	05:31:39 14:50:25	37370 38690 21299 21795 21329 19430 10312 10312 10312 10313	1
21	04:02:27 16:55:48	49696 53339 28426 28460 31912 31788 16209 12861 12861 12860	1
22	02:00:31 17:46:04	62735 61989 37593 37633 37399 37033 18518 17678 14913 14913	1
23	07:08:18 14:03:39	26986 31584 16972 16952 13522 13302 4952 8054 8595 8613	1
24	00:22:09 02:22:43	13351 7454 5288 3854 3847 3843 1443 1443 1443 1444	1
25	03:32:54 16:53:39	52240 53834 29020 31106 33205 32831 16425 13205 13205 13205	2

2008 год

1. Системы счисления

Воспользовавшись развернутой записью чисел, можно переписать заданные количества учеников в следующем виде:

$$1000_x = x^3;$$
$$120_x = x^2 + 2 \cdot x;$$
$$110_x = x^2 + x.$$

Учитывая же, что общее количество учеников в классе должно быть равно сумме количеств девочек и мальчиков, можно записать уравнение:

$$x^2 + 2 \cdot x + x^2 + x = x^3.$$

Отсюда

$$x^3 - 2 \cdot x^2 - 3 \cdot x = 0.$$

При этом следует учесть, что, поскольку x представляет собой основание системы счисления, а по крайней мере в одном из исходных чисел, записанных в этой системе счисления, фигурирует цифра 2, значение x должно быть натуральным числом, не меньшим 2.

Решаем приведенное выше уравнение:

- так как $x \neq 0$, $x^3 - 2 \cdot x^2 - 3 \cdot x = 0 \Rightarrow x^2 - 2 \cdot x - 3 = 0$;
- дискриминант квадратного уравнения $D = (-2)^2 + 4 \cdot 3 = 16$;
- корни уравнения: $x_{1,2} = \frac{-b \pm \sqrt{D}}{2a} \Rightarrow x_1 = 3, x_2 = -1$.

Вспомнив, что x — натуральное число, второй из найденных корней отбрасываем.

Ответ: $x = 3$ — троичная система счисления.

Критерии оценивания

Получен правильный обоснованный ответ, решение грамотно оформлено — 5 баллов.

Получен правильный обоснованный ответ — 4 балла.

Получен обоснованный ответ, но при вычислениях допущена арифметическая ошибка — 3 балла.

Записана верная формула решения, но ответ не получен — 1—2 балла.

2. Единицы измерения информации (1)

$65\,536 = 2^{16}$. Тогда для кодирования информации о цвете одной точки требуется 2 байта.

28 800 бит/с = 3600 байт/с.

За 2 минуты (120 с) можно передать $120 \cdot 3600 = 432\,000$ байт, что соответствует информации о 216 000 точек изображения.

Поскольку ширина изображения составляет 500 точек, его высота определяется следующим образом: $216\,000 / 500 = 432$ точки.

Ответ: 432 точки.

Критерии оценивания

Получен правильный обоснованный ответ, решение грамотно оформлено — 5 баллов.

Получен правильный обоснованный ответ — 4 балла.

Получен обоснованный ответ, но при вычислениях допущена арифметическая ошибка — 3 балла.

Записана верная формула решения, но ответ не получен — 1—2 балла.

3. Логическое выражение

Рассмотрим исходное логическое выражение: $(\neg K \vee M) \rightarrow \rightarrow (\neg L \vee M \vee N)$.

Последней по порядку в нем выполняется логическая операция следования, или импликация (→). Согласно ее таблице истинности, результат такой логической операции равен ложному значению (0) только в одном-единственном случае — когда из значения 1 (ИСТИНА) следует значение 0 (ЛОЖЬ). Тогда:

$$\begin{cases} \neg K \vee M = 1, \\ \neg L \vee M \vee N = 0. \end{cases}$$

Вначале рассмотрим второе из этих уравнений. Оно содержит логические операции ИЛИ, результат которых может быть ложным также только в одном-единственном случае — если ложными являются все объединяемые этими операциями предпосылки. Поэтому можно однозначно сделать вывод, что $M = 0$, $N = 0$, а $L = 1$ (так как перед этой переменной стоял знак операции инверсии).

Теперь рассмотрим первое из двух уравнений, подставив в него уже найденное значение переменной M:

$$\neg K \vee 0 = 1.$$

Отсюда следует вывод, что значение переменной K должно быть равно 0 (учитывая операцию инверсии).

Остается записать полученные значения переменных в требуемой последовательности: $(KLMN) = 0100$.

Ответ: 0100.

Критерии оценивания

Получен правильный обоснованный ответ, решение грамотно оформлено — 10 баллов.

Получен правильный обоснованный ответ — 4 балла.

Получен обоснованный ответ, но при вычислениях допущена арифметическая ошибка — 3 балла.

Записана верная формула решения, но ответ не получен — 1—2 балла.

4. Единицы измерения информации (2)

Решение:

$$\begin{cases} 2^{x+3}\ (\text{байт}) = 8^{y-7}\ (\text{Мбайт}), \\ 2^{2y-9}\ (\text{Кбайт}) = 4^{2x-5}\ (\text{бит}); \end{cases}$$

$$\begin{cases} 2^{x+3} \cdot 2^3\ (\text{бит}) = 2^{3(y-7)} \cdot 2^{20} \cdot 2^3\ (\text{бит}), \\ 2^{2y-9} \cdot 2^{10} \cdot 2^3\ (\text{бит}) = 2^{2(2x-5)}\ (\text{бит}); \end{cases}$$

$$\begin{cases} 2^{(x+3)+3}\ (\text{бит}) = 2^{3(y-7)+20+3}\ (\text{бит}), \\ 2^{(2y-9)+10+3}\ (\text{бит}) = 2^{2(2x-5)}\ (\text{бит}); \end{cases}$$

$$\begin{cases} (x+3)+3 = 3(y-7)+23, \\ (2y-9)+13 = 2(2x-5); \end{cases}$$

$$\begin{cases} x+6 = 3y+2, \\ 2y+4 = 4x-10; \end{cases} \qquad \begin{cases} x = 3y-4, \\ y = 2x-7; \end{cases}$$

$$\begin{cases} y = 2(3y-4)-7, \\ y = 3; \end{cases} \qquad \begin{cases} x = 5, \\ y = 3. \end{cases}$$

Ответ: $x = 5$, $y = 3$.

Критерии оценивания

Мегабайты, килобайты и байты правильно переведены в биты. Уравнения выражены через степень с основанием, равным 2. Система показательных уравнений приведена к системе двух уравнений с двумя переменными. Система решена одним из возможных методов и получено правильное решение — 8—10 баллов.

Мегабайты, килобайты и байты правильно переведены в биты. Уравнения выражены через степень с основанием, равным 2. Система показательных уравнений приведена к системе двух уравнений с двумя переменными — 6—7 баллов.

Мегабайты, килобайты и байты правильно переведены в биты. Уравнения выражены через степень с основанием, равным 2, — 4—5 баллов.

Мегабайты, килобайты и байты правильно переведены в биты — 1—3 балла.

5. Терминология

Критерии оценивания

Ответ учащегося, записанный в произвольной форме, содержит корректные определения — 1—5 баллов.

В ответе перечислены все основные свойства алгоритма, указаны все возможные способы записи и перечислены основные

алгоритмические конструкции (линейная, ветвление, цикл) — 1—5 баллов.

Дан развернутый ответ, содержащий дополнительные сведения, определения, примеры, рисунки (блок-схемы) и т. п., — 1—3 балла.

Корректность, грамотность и логичность изложения — 1—2 балла.

При определении итогового балла за выполнение задачи баллы, выставленные по вышеотмеченным критериям, суммируются.

6. Программирование

Основные этапы решения задачи:

1) ввод значения числа *N*;

2) расчет суммы цифр числа *N* (имя переменной — *сумма*);

3) нахождение максимального делителя;

4) вывод результата.

Программа на школьном алгоритмическом языке:

```
алг Задача_
нач цел n, m, сумма, цифра, возм_делит
| |1. Ввод значения числа n
| вывод нс, "Введите значение n"
| ввод n
| |2. Расчет суммы цифр числа n
| сумма := 0
| нц пока n > 0
| | цифра := mod(n, 10)
| | сумма := сумма + цифра
| | n := div(n, 10)
| кц
| |3. Нахождение максимального делителя
| |Максимально возможный делитель
| возм_делит := div(сумма, 2)
| нц пока mod(сумма, возм_делит) <> 0
| | |Следующий возможный делитель
| | возм_делит := возм_делит - 1
| кц
| |4. Вывод результата
| если возм_делит = 1
| |то |Число n - простое
| | вывод нс, "НЕТ"
| |иначе
| | вывод нс, "М=", возм_делит
| все
кон
```

Возможен также способ решения, в котором учитывается, что у четного числа максимальный делитель равен его половине, а у нечетного не может быть четных делителей (при этом проверка возможных делителей проводится до целой части квадратного корня из числа *N*):

```
|3. Нахождение максимального делителя
если mod(сумма, 2) = 1
|то
| |Рассматриваются только нечетные числа
| |Первый возможный делитель
| возм_делит := 3
| нц пока mod(сумма, возм_делит) <> 0 и возм_делит <= цел(sqrt(сумма))
| | |Следующий возможный делитель
| | возм_делит := возм_делит + 2
| кц
| |4. Вывод результата
| если возм_делит > цел(sqrt(сумма))
| |то |Число n – простое
| |вывод нс, "НЕТ"
| |иначе
| | |Встретился делитель (минимальный), равный возм_делит
| | вывод нс, "М=", div(сумма, возм_делит)
| все
|иначе
| возм_делит := div(сумма, 2)
| |4. Вывод результата
| вывод нс, "М=", возм_делит
все
```

Критерии оценивания

Разработана рациональная, грамотно оформленная программа для всех вариантов входных данных, и полный перебор не используется (второй способ решения) — 20—25 баллов.

Разработана рациональная, грамотно оформленная программа для всех вариантов входных данных, но используется полный перебор (первый способ решения) — 15—20 баллов.

Разработана программа, правильная не для всех вариантов входных данных (например, только для четных сумм цифр), — 1—15 баллов.

7. Программирование на компьютере

1. Решение без использования вспомогательных функций.

```
алг Задача_6
нач вещ xa, ya, xb, yb, xc, yc (координаты вершин треугольника)
  xd, yd (координаты точки)
  AB, BC, AC (стороны заданного треугольника)
  AD, BD, CD (расстояние от вершин треугольника до точки)
```

```
    плABC (площадь заданного треугольника через его стороны)
    плABD, плBCD, плACD (площади трех «маленьких» треугольников)
    пл3 (их сумма)
    dx, dy (разность абсцисс и разность ординат)
    пп (полупериметр)
    точность (величина, учитывающая точность вычислений)
| точность := 1.E-05
| вывод нс, "Введите абсциссу и ординату 1-й вершины треугольника"
| ввод xa, ya
| вывод нс, "Введите абсциссу и ординату 2-й вершины треугольника"
| ввод xb, yb
| вывод нс, "Введите абсциссу и ординату 3-й вершины треугольника"
| ввод xc, yc
| вывод нс, "Введите абсциссу и ординату рассматриваемой точки"
| ввод xd, yd
| |Определяем площадь треугольника через его стороны
| |Длины сторон
| |AB
| dx := xa - xb; dy := ya - yb
| AB := sqrt(dx * dx + dy * dy)
| |BC
| dx := xb - xc; dy := yb - yc
| BC := sqrt(dx * dx + dy * dy)
| |AC
| dx := xa - xc; dy := ya - yc
| AC := sqrt(dx * dx + dy * dy)
| |Площадь
| пп := (AB + BC + AC)/2
| плABC := sqrt(пп * (пп - AB) * (пп - BC) * (пп - AC))
| |Определяем площадь треугольника как сумму 3-х площадей
| |Длины сторон
| |AD
| dx := xa - xd; dy := ya - yd
| AD := sqrt(dx * dx + dy * dy)
| |BD
| dx := xb - xd; dy := yb - yd
| BD := sqrt(dx * dx + dy * dy)
| |CD
| dx := xc - xd; dy := yc - yd
| CD := sqrt(dx * dx + dy * dy)
| |Площади отдельных треугольников
| |ABD
| пп := (AB + AD + BD)/2
| плABD := sqrt(пп * (пп - AB) * (пп - AD) * (пп - BD))
| |BCD
| пп := (BC + BD + CD)/2
| плABD := sqrt(пп * (пп - BC) * (пп - BD) * (пп - CD))
| |ACD
| пп := (AC + AD + CD)/2
| плABD := sqrt(пп * (пп - AC) * (пп - AD) * (пп - CD))
| |Сумма
```

```
| пл3 := плABD + плBCD + плACD
| если abs(пл3 - плABC)<= точность
| |то
| | вывод нс, "Точка принадлежит треугольнику"
| |иначе
| | вывод нс, "Точка не принадлежит треугольнику"
| все
кон
```

2. Решение с использованием вспомогательных функций.

Функция для расчета длины стороны:

```
алг вещ Длина(арг вещ x1, y1, x2, y2)
нач вещ dx, dy
| dx := x1 - x2
| dy := y1 - y2
| знач := sqrt(dx * dx + dy * dy) |Значение функции
кон
```

Функция для расчета площади треугольника (по формуле Герона):

```
алг вещ Площадь(арг вещ a, b, c)
нач вещ пп
| пп := (a + b + c)/2
| знач := sqrt(пп * (пп - a) * (пп - b) * (пп - c))
кон
```

Основная часть программы:

```
алг Задача_6
нач вещ xa, ya, xb, yb, xc, yc (координаты вершин треугольника)
  xd, yd (координаты точки)
  AB, BC, AC (стороны заданного треугольника)
  AD, BD, CD (расстояние от вершин треугольника до точки)
  плABC (площадь заданного треугольника через его стороны)
  плABD, плBCD, плACD (площади трех «маленьких» треугольников)
  пл3 (их сумма)
  точность (величина, учитывающая точность вычислений)
| точность := 1.E-05
| вывод нс, "Введите абсциссу и ординату 1-й вершины треугольника"
| ввод xa, ya
| вывод нс, "Введите абсциссу и ординату 2-й вершины треугольника"
| ввод xb, yb
| вывод нс, "Введите абсциссу и ординату 3-й вершины треугольника"
| ввод xc, yc
| вывод нс, "Введите абсциссу и ординату рассматриваемой точки"
| ввод xd, yd
| |Определяем площадь треугольника через его стороны
| |Длины сторон
| AB := Длина(xa, ya, xb, yb)
| BC := Длина(xb, yb, xc, yc)
| AC := Длина(xa, ya, xc, yc)
| |Площадь
```

```
| плABC := Площадь(AB, BC, AC)
| |Определяем площадь треугольника как сумму 3-х площадей
| |Длины сторон
| AD := Длина(xa, ya, xd, yd)
| BD := Длина(xb, yb, xd, yd)
| CD := Длина(xa, ya, xd, yd)
| |Площадь отдельных треугольников
| плABD := Площадь(AB, AD, BD)
| плBCD := Площадь(BC, BD, CD)
| плACD := Площадь(AC, AD, CD)
| |Сумма
| пл3 := плABD + плBCD + плACD
| если abs(пл3 - плABC)<= точность
| |то
| | вывод нс,"Точка принадлежит треугольнику"
| |иначе
| | вывод нс, "Точка не принадлежит треугольнику"
| все
кон
```

Критерии оценивания

Разработана рациональная, грамотно оформленная программа для всех вариантов входных данных с использованием вспомогательных функций (2-й вариант решения) — 30—35 баллов.

Разработана рациональная, грамотно оформленная программа для всех вариантов входных данных, но без использования вспомогательных функций (1-й вариант решения) — 25—30 баллов.

Разработана правильная программа без учета точности округления — 15—25 баллов.

Разработана правильная программа не для всех вариантов входных данных (например, решение работает, когда одна из сторон параллельна или перпендикулярна одной из осей координат) — 1—15 баллов.

Литература для подготовки к олимпиаде по информатике

1. *Акулов О. А., Медведев Н. В.* Информатика: базовый курс. — 2-е изд., испр. и доп. — М.: Омега-Л, 2005.
2. *Андреева Е. В.* Программирование — это так просто, программирование — это так сложно. Современный учебник программирования. — М.: Изд. МЦНМО, 2009.
3. *Аникеева Л. А., Гусева О. Б.* Работа над ошибками ЕГЭ. Информатика. Наиболее сложные темы. Несколько вариантов тестирования. — М.: СОЛОН-ПРЕСС, 2010.
4. *Брыков И. А., Михалин Д. А., Пономаренко А. С.* История олимпиад по программированию в Центральном округе г. Москвы // Вестник центра информационных технологий НМЦ ЦОУО ДО г. Москвы. — 2006. — Вып. 1.
5. *Волченков С. Г., Корнилов П. А., Белов Ю. А. и др.* Ярославские олимпиады по информатике. Сборник задач с решениями. — М.: БИНОМ. Лаборатория знаний, 2010.
6. *Гусева А. И.* Учимся программировать. Pascal 7.0. Задачи и методы их решения: учебное пособие. — 2-е изд. перераб. и доп. — М.: ДИАЛОГ-МИФИ, 2001.
7. *Долинский М. С.* Решение сложных и олимпиадных задач по программированию: учебное пособие. — СПб.: Питер, 2006.
8. *Кирюхин В. М., Окулов С. М.* Методика решения задач по информатике. Международные олимпиады. — М.: БИНОМ. Лаборатория знаний, 2007.
9. *Меньшиков Ф. В.* Олимпиадные задачи по программированию. — СПб.: Питер, 2006.
10. *Окулов С. М., Ашихмина Т. В., Бушмелева Н. А. и др.* Задачи по программированию / под ред. С. М. Окулова. — М.: БИНОМ. Лаборатория знаний, 2006.
11. *Пономаренко А. С.* Трехмерные иллюзии // Информатика. — 2000. — № 3.
12. *Ремнев А. А., Федотова С. В.* Курс Delphi для начинающих. Полигон нестандартных задач. — М.: СОЛОН-ПРЕСС, 2007.
13. *Сафронов И. К.* Готовимся к ЕГЭ. Информатика. — СПб.: БХВ-Петербург, 2007.
14. *Филичев С. В.* Занимательный Basic. — М.: ЭКОМ, 1997.

Содержание