РАЗВИТИЕ ИНТЕЛЛЕКТА ШКОЛЬНИКОВ

ЯРОСЛАВСКИЕ ОЛИМПИАДЫ ПО ИНФОРМАТИКЕ

СБОРНИК ЗАДАЧ С РЕШЕНИЯМИ

Москва
БИНОМ. Лаборатория знаний
2010

УДК 519.85(023)
ББК 32.973-018
В67

Серия основана в 2008 г.

Авторский коллектив: С. Г. Волчёнков, П. А. Корнилов, Ю. А. Белов, Н. Л. Дашниц, В. А. Никулин, Н. И. Заводчикова

Волчёнков С. Г.

В67 Ярославские олимпиады по информатике. Сборник задач с решениями / С. Г. Волчёнков, П. А. Корнилов, Ю. А. Белов и др. — М. : БИНОМ. Лаборатория знаний, 2010. — 405 с. : ил. — (Развитие интеллекта школьников).

ISBN 978-5-9963-0106-5

Книга содержит задачи, предлагавшиеся на теоретическом и практическом турах ярославских олимпиад по информатике с 1987 по 2009 г. Ко всем задачам приведены решения или краткие указания к решению.

Для школьников, занимающихся подготовкой к олимпиадам, учителей и тех, кого увлекает решение логических задач и головоломок алгоритмического содержания.

УДК 519.85(023)
ББК 32.973-018

По вопросам приобретения обращаться:
«БИНОМ. Лаборатория знаний»
Тел.: (499) 157-52-72, e-mail: binom@Lbz.ru
http://www.Lbz.ru

ISBN 978-5-9963-0106-5 © БИНОМ. Лаборатория знаний, 2010

ПРЕДИСЛОВИЕ

Вот уже более 20 лет в Ярославле проводятся олимпиады по информатике. За это время накоплен огромный теоретический и практический материал, который сегодня представляет значительный интерес — как методический, так и исторический.

В первый год проведения олимпиады в Ярославле — в 1987 г. — школьная информатика еще делала первые шаги; еще не было какой-то единой учебной программы; учителями становились либо программисты, работавшие в вычислительных центрах, либо преподаватели высших учебных заведений, где имелись соответствующие специальности, или же учителей готовили «на скорую руку» на краткосрочных курсах. Компьютеров в школах тоже практически не было: в большинстве случаев реализовывался так называемый «бескомпьютерный» вариант преподавания информатики. Олимпиады по информатике тогда были призваны привлечь внимание к молодому учебному предмету, сплотить сообщество учителей информатики, заинтересовать талантливую молодежь. В то время проведение таких олимпиад было делом совершенно новым. Что давать в качестве олимпиадных заданий, как обеспечить выполнение этих заданий, как их проверять — на все эти вопросы ответы приходилось искать, как говорится, «в рабочем порядке». Подавляющее большинство пришедших в информатику людей имели математическую подготовку. Этим можно объяснить «математический» уклон большинства олимпиадных заданий. Кроме того, компьютеры, которые тогда использовались на практических турах, были чрезвычайно разнородными; некоторые из них имели крайне малую производительность. Поэтому не следует удивляться временны́м ограничениям, которые устанавливались в те годы для прохождения тестов на практических турах: иногда они достигали нескольких минут на самых несложных задачах.

В первые годы проведения олимпиад почти все школьники были еще слабо подготовлены в области реального программирования; в частности, многие тогдашние участники никогда не работали с файлами. Поэтому на практических турах первых олимпиад в большинстве задач разрешалось вводить данные с клавиатуры и выводить результаты на экран, а их проверка осуществлялась вручную. В заданиях теоретического тура для написания алгоритмов тогда применялся «школьный алгоритмический язык». Сегодня же для этого чаще используется язык Паскаль (или аналогичный ему). Поэтому в данной книге мы постарались использовать и то и другое, руководствуясь простотой понимания.

В большинстве теоретических задач от школьников требовалось описать тот или иной алгоритм. Язык, на котором это надо было сделать, вообще говоря, не уточнялся. Требовалось, чтобы описание было ясно проверяющему и не отягощалось излишними формальностями, присущими реальным языкам программирования. Требование написать алгоритм в ряде случаев вообще заменялось упрощенной формой: например, слова «опишите алгоритм нахождения...» могли заменяться словами «найдите...». В некоторых же заданиях предлагалось написать программу; в этом случае можно было использовать любой язык программирования. При оценке алгоритмов на теоретическом туре всегда учитывалась их трудоемкость.

Сегодня многие идеи и методы (например, *дихотомия*, *backtracking*, *динамическое программирование* и т. д.) знакомы большинству школьников, решающих олимпиадные задачи. Однако 20 лет назад это было совсем не так, поэтому в решениях часто приходилось подробно описывать кажущиеся сейчас очевидными вещи. В представленном издании для сохранения исторической достоверности в некоторых местах оставлены такие описания.

Цели и задачи олимпиад за прошедшие 20 лет сильно изменились. Изменилась и форма их проведения. Стало больше компьютеров, отпала необходимость проведения на областных олимпиадах теоретического (отборочного) тура. В последние годы для проведения областных олимпиад

предлагаются задачи, рекомендуемые центральной методической комиссией. Некоторые из подобных задач использовались и в ярославских олимпиадах. В представленном сборнике эти задачи не приводятся, так как их условия и решения так или иначе уже опубликованы.

За все время проведения олимпиад в Ярославле дважды издавались брошюры с заданиями и решениями. Однако такие издания осуществлялись крайне малыми тиражами, и сейчас найти эти брошюры практически невозможно. Поэтому ниже приведены и предисловия к указанным изданиям, чтобы можно было «уловить дух того времени».

В этой книге мы постарались сохранить все первоначальные условия задач (исправления сделаны только в исключительных случаях). Около каждой задачи в скобках указано количество баллов, которые школьник мог получить за полное и оптимальное решение данной задачи. В разные годы эта оценка не согласовывалась, поэтому 10 баллов в какой-то год могло быть «много», а в какой-то год — «мало». Баллы же за задачи практического тура рассчитывались так, чтобы практический тур был немного значимее теоретического.

За многие годы подготовку и проведение олимпиад осуществляли искренне преданные информатике люди. Часть из них продолжает эту деятельность и сейчас. Коллектив авторов искренне благодарен А.Н.Смирновой, А.А.Короткину, Я.Н.Зайдельману, а также авторам многочисленных задач, использованных на наших олимпиадах, чьи имена остались неизвестными.

Предисловие к книге «Ярославские олимпиады по программированию» 1990 г.[1]

Начиная с 1987 г. в Ярославле проводятся олимпиады по программированию (городские и областные). В них принимают участие учащиеся 9–10 классов (хотя возраст участни-

[1] Ярославские олимпиады по программированию. Методические рекомендации и методические указания. — Ярославль: Ярославский областной институт усовершенствования учителей, 1990.

ков не ограничивается) из Ярославля, Переславля, Тутаева, Рыбинска. Олимпиада организуется областным институтом усовершенствования учителей, Ярославским университетом и пединститутом.

Олимпиады проводятся в два тура. В первом туре (теоретическом) участникам предлагается некоторый список задач, из которого они могут выбрать интересные для себя задачи и в течение 4 часов их решать. Решением может быть программа на любом языке программирования, словесное либо графическое описание алгоритма, в некоторых случаях — доказательство теоретического факта и др.

В практическом туре участникам, показавшим хорошие результаты в теоретическом туре, предлагается ряд задач на выбор, из которых нужно в течение 2–3 часов написать, отладить программу и продемонстрировать ее на предложенных тестах. В зачет идут только отлаженные программы.

Часть предлагаемых задач требует не столько умения программировать на каком-либо языке, сколько логики алгоритмического мышления, а иногда — и просто здравого смысла (в сочетании с сообразительностью). Ниже приводится список задач, предлагавшихся в течение трех лет на городских и областных олимпиадах. В разборе задач, кроме необходимых методических указаний, в некоторых случаях приведены наиболее интересные решения.

Предисловие к книге «Ярославские олимпиады по информатике» 1995 г.[1]

В течение ряда лет в Ярославле проводятся городские и областные олимпиады по информатике. Информатика — один из самых молодых предметов в школе. Начавшаяся в середине 1980-х гг. всеобщая компьютеризация школьного образования породила ряд непростых проблем, которые отразились и на методах организации олимпиад по информатике.

---

[1] *Бадин Н. М., Волченков С. Г., Дашниц Н. Л., Корнилов П. А.* Ярославские олимпиады по информатике. — Ярославль: Ярославский областной институт повышения квалификации педагогических и руководящих работников образования, 1995.

Главной проблемой была и во многом остается на сегодня сильная разнородность вычислительных средств в школах: от микрокалькуляторов до персональных IBM PC AT. На олимпиадах данная проблема была решена почти сразу и, как показала практика, весьма удачно, — это ориентир не на чистое программирование и знание специфических элементов компьютера, а на умение составлять алгоритмы, четко описывать свои действия и знание разнообразных методов решения дискретных задач типа перебора, сортировки, рекурсии, динамического программирования и др.

Олимпиада проводится в два тура. На первом (теоретическом) обычно дается 6–8 задач, из которых в зачет участнику идут четыре лучших решения. На втором (практическом) туре на выбор даются две задачи для программирования непосредственно на компьютере.

Основными целями, стоящими перед такой олимпиадой, являются:

- выделение наиболее способных школьников, что позволяет в дальнейшем вести с ними целенаправленную индивидуальную работу;
- проведение анализа состояния школьной информатики в целом — оснащенности школ техникой и программным обеспечением, уровня подготовки школьников.

Олимпиада является и сильным обучающим средством. В процессе решения, а особенно на разборах задач, ребята узнают новые методы и алгоритмы. Погружение школьников в интеллектуальную среду позволяет стимулировать интерес к дальнейшей работе. Особенно это проявляется на выездных областных олимпиадах.

В 1990 г. вышел сборник задач, предлагавшихся на ярославских олимпиадах по программированию 1987–1989 гг.[1] Данное пособие является его продолжением. Оно содержит задачи олимпиад 1990–1995 гг. с решениями. Многие из приводимых здесь алгоритмов могут быть запрограммирова-

---

[1] Ярославские олимпиады по программированию. Методические рекомендации и методические указания. — Ярославль: Ярославский областной институт усовершенствования учителей, 1990.

ны и использованы как в школьном курсе информатики, так и на младших курсах вузов. Представленные задачи охватывают очень большой спектр различных стандартных и нестандартных алгоритмов.

Для каждой задачи указано количество баллов, которые участник мог получить за полное и качественное ее решение. Сейчас некоторые из этих оценок кажутся завышенными, так как за прошедшие годы уровень подготовки школьников и сложность предлагаемых на олимпиадах задач значительно повысились.

ОЛИМПИАДЫ 1987/88 УЧЕБНОГО ГОДА

Городская олимпиада

Теоретический тур

1. Совершенные числа (7 баллов)

Совершенными числами называются числа, равные сумме своих собственных делителей, включая 1, — например, $6 = 1 + 2 + 3$. Найдите и напечатайте все совершенные числа из диапазона от 1 до n ($1 < n < 10000$).

2. Номер дня (7 баллов)

Опишите алгоритм нахождения номера заданного дня с начала года. День задается тремя числами: d — номер дня в месяце, m — номер месяца в году и g — номер года.

3. Квадратики (7 баллов)

Дана числовая таблица $A[n, n]$, каждый элемент которой равен 0, 1, 5 или 7. Предложите способ подсчета в таблице количества квадратиков 2×2, в каждом из которых все элементы различны.

4. Последовательность (10 баллов)

Напишите программу, выводящую первые n элементов последовательности, которая описывается следующими правилами:

1) число 1 является элементом последовательности;
2) если a — элемент последовательности, то $2a$, $3a$ и $5a$ тоже являются элементами последовательности;
3) последовательности принадлежат только элементы, заданные правилами 1 и 2.

5. Точка в треугольнике (5 баллов)

Заданы декартовы координаты (x_1, y_1), (x_2, y_2), (x_3, y_3) вершин треугольника и координаты (x, y) некоторой точки M. Опишите алгоритм, определяющий, находится ли точка M внутри треугольника.

6. Соседние клетки (10 баллов)

Клетки шахматной доски пронумерованы числами от 1 до 64 по строкам слева направо и снизу вверх. По заданному номеру клетки определите номера всех клеток, имеющих с ней общую сторону.

Областная олимпиада

Теоретический тур

7. Цикловоз (8 баллов)

На пути $P1$ стоят пять пронумерованных вагонов, а в тупике находится паровоз (рис. 1). Стрелка соединяет тупик с путем $P1$. В тупике, кроме паровоза, могут поместиться два вагона.

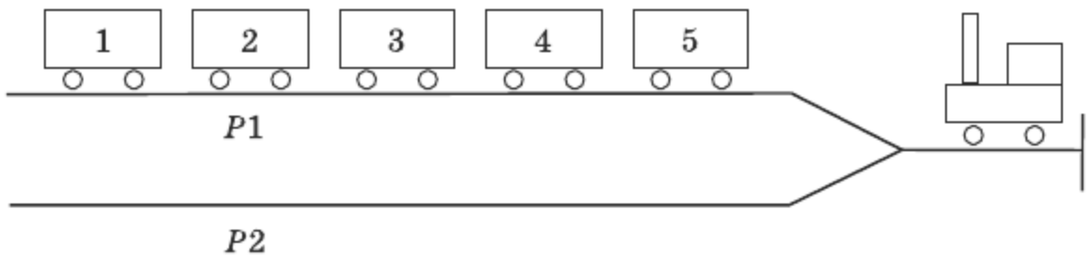


Рис. 1

Напишите возможно более короткую программу для машиниста паровоза, выполнение которой переведет состав с пути $P1$ на путь $P2$, не изменив порядка следования вагонов. Паровоз должен вернуться в тупик. Программа должна состоять из пронумерованных подряд строк, каждая из которых содержит одну из следующих команд:

1) влево (**Л**) — паровоз из тупика продвигается в крайнее левое положение на один из путей в зависимости от положения стрелки;
2) вправо (**П**) — паровоз продвигается в тупик вместе с прицепленными вагонами, если стрелка соединяет его путь с тупиком. Если стрелка установлена в другое положение, то фиксируется ошибка;
3) перевести стрелку (**С**) — стрелка переводится в противоположное положение;
4) прицепить вагон (**+**) — к паровозу или к паровозу с уже прицепленным вагоном добавляется еще один вагон;
5) отцепить вагон (**–**) — от состава отцепляется один вагон;

6) выполнить команды с номера N до номера M (**Ц** N M) — команды из программы с номерами N, $N + 1$, ..., M выполняются так, как если бы они находились на месте данной команды.

Примечание: программы, содержащие наименьшее количество команд, премируются дополнительными баллами.

8. Алгоритм (6 баллов)

Что делает следующий алгоритм?

```
алг лит альфа (лит x)
нач
    если длин(x)=1
       то знач:=x
       иначе знач:= альфа (x[2:длин(x)])+x[1:1]
    все
кон
```

9. Период дроби (7 баллов)

По заданным натуральным числам N и M требуется определить и напечатать период десятичной дроби $\frac{N}{M}$. Если дробь конечна, то ее период состоит из цифры 0.

10. Печать по спирали (5 баллов)

Дана таблица чисел, у которой m строк и n столбцов. Распечатайте числа из таблицы по спирали, начиная обход с правого верхнего угла и двигаясь по часовой стрелке.

11. Обход конем (10 баллов)

Опишите алгоритм обхода шахматной доски ходом коня так, чтобы все клетки были пройдены по одному разу.

Практический тур

12. Паук и муха (10 баллов)

На поверхности единичного куба с координатами вершин (0,0,0), (0,0,1), (0,1,0), (1,0,0), (1,1,0), (1,0,1), (0,1,1), (1,1,1) заданы две точки A и B. В точке A сидит паук, а в точке B — муха. Найдите длину кратчайшего пути между пауком и мухой, проходящего по поверхности куба.

Входные данные:

- `a1, a2, a3` — координаты точки A, разделенные пробелом;
- `b1, b2, b3` — координаты точки B, разделенные пробелом.

Данные вводятся из файла `spider.in` или с клавиатуры.

Выходные данные:

- D — результат с точностью до 0.0001.

Результат выводится в файл `spider.out` или на экран.

Пример:

spider.in	spider.out
0.5 0 0 0 1 0.5	1.4142

13. Под боем (5 баллов)

На шахматной доске стоят ферзь и конь. Каждая фигура задана своими координатами в шахматной нотации. Определите количество полей, не занятых фигурами, которые находятся под боем.

Входные данные:

- xn, ym — координаты ферзя и коня в шахматной нотации, разделенные пробелом, где x, y — латинские буквы от `a` до `h`, означающие вертикаль, а n, m — числа от 1 до 8, означающие горизонталь.

Данные вводятся из файла `chess.in` или с клавиатуры.

Выходные данные:

- D — результат (натуральное число).

Результат выводится в файл `chess.out` или на экран.

Пример:

chess.in	chess.out
a1 b2	16

ОЛИМПИАДЫ 1988/89 УЧЕБНОГО ГОДА

Городская олимпиада

Теоретический тур

1. Алгоритм (3 балла)

Что делает данный алгоритм?

```
алг вещ ALFA(цел N,I,J, вещтаб A[1:N])
 арг
 рез
нач цел k
если I=J
 то знач:=A[I]
 иначе k:=цел_часть((I+J)/2)
     знач:=ALFA(N,I,k,A)+ALFA(N,k+1,J,A)
 все
кон
```

2. Взаимная простота (3 балла)

Опишите алгоритм, который определяет, являются ли три заданных натуральных числа взаимно простыми. Числа называются взаимно простыми, если их наибольший общий делитель равен 1.

3. Длинное деление (6 баллов)

Напишите программу, которая считывает натуральные чис-

N

Областная олимпиада

Теоретический тур

11. Что получится? (3 балла)

Какое значение принимает x после выполнения алгоритма?

```
алг ALFA (нат N, вещтаб A[1:N], вещ x)
     арг N,A
     рез x
   нач
   если N=1
       то x:=A[1]
       иначе ALFA (N-1, A, x)
         если A[N] > x
         то x:=A[N]
       все
   все
  кон
```

12. Путь по клеткам (13 баллов)

На клетчатом листе бумаги размером 10×10 закрашены некоторые клетки. Разрешается ходить по незакрашенным клеткам, переходя на каждом шаге вверх, вниз, вправо или влево. Описать алгоритм, отвечающий на следующие вопросы:

А) есть ли путь из левой нижней клетки в правую верхнюю (5 баллов);

Б) какое минимальное число шагов нужно сделать, чтобы пройти этот путь (3 балла);

В) по каким клеткам при этом надо идти (5 баллов).

13. Конечность алгоритма (10 баллов)

Алгоритм обрабатывает конечную последовательность из нулей и единиц. На каждом шаге он заменяет некоторую группу символов «01» на группу «10 ... 0» (число нулей — любое). Докажите, что такой алгоритм не может работать бесконечно долго.

14. Покрытие (5 баллов)

Имеется клетчатый прямоугольник $M \times N$. В некоторых внутренних узлах отмечены точки, заданные своими координатами (x_i, y_i) (пример см. на рис. 2) Каждая точка является центром некоторого квадратика 2×2.

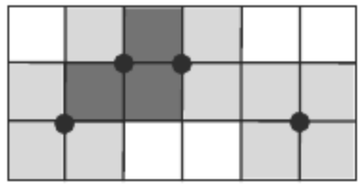


Рис. 2

Найдите:

А) количество клеток прямоугольника, не покрываемых квадратиками (на рис. 2 — белых клеток) (2 балла);

Б) количество клеток, покрытых нечетное число раз (на рис. 2 — светло-серых клеток) (3 балла).

15. Нормализация (4 балла)

Напишите программу, которая считывает обычное десятичное число в виде строки символов и печатает его в нормализованном виде.

Пример:

Обычная запись	Нормализованная форма
32.1	0.321E2
-0.002	-0.2E-2
12.345678901234567	0.12345678901234567E2

16. Вычисление выражения (6 баллов)

Напишите программу для вычисления арифметических выражений, содержащих однозначные числа, связанные знаками +, -, * и /. Скобки отсутствуют.

Пример:

-7*3/2+5*4.

Результат: 9.5.

17. Упорядочение (6 баллов)

Опишите алгоритм упорядочения списка фамилий учеников класса в лексикографическом порядке (по правилам упорядочения слов в русских словарях).

Практический тур

18. Римская арифметика (15 баллов)

Даны два натуральных числа в римской нотации. Напишите программу для вычисления их суммы. Результат выведите в римской нотации. Сумма чисел не превышает 3000.

Краткие правила римской системы счисления.

Числа от 1 до 9:

1 — I, 2 — II, 3 — III, 4 — IV, 5 — V,
6 — VI, 7 — VII, 8 — VIII, 9 — IX.

Десятки записываются по тем же правилам, только вместо I используется X, вместо V используется L, а вместо X используется C. Для записи сотен используются символы C, D и M, соответственно. Обозначение нуля отсутствует.

Пример:

CMLXXIX = 900 + 70 + 9 = 979.

Входные данные:

- N, M — две строки символов, соответствующие римским числам; $N + M \leqslant 3000$.

Данные вводятся из файла roma.in или с клавиатуры.

Выходные данные:

- S — строка символов, соответствующая сумме.

Результат выводится в файл roma.out или на экран.

Пример:

roma.in	roma.out
CMLXXIX I	CMLXXX

19. Арбузы (10 баллов)

На прилавке в один ряд лежат N арбузов. Известно, что каждый арбуз на 100 граммов легче, чем среднее арифметическое весов его соседей. Найдите вес второго арбуза с точностью до грамма, если первый арбуз весит a килограммов, а последний — b килограммов.

Входные данные:

- N — натуральное число; $2 \leqslant N \leqslant 1000$;
- a, b — два числа типа `real`.

Данные вводятся из файла `melon.in` или с клавиатуры.

Выходные данные:

- c — результат.

Результат выводится в файл `melon.out` или на экран.

Пример:

melon.in	melon.out
3 1 2.6	1.7

20. Площадь многоугольника (8 баллов)

Даны координаты вершин многоугольника в порядке обхода вдоль контура. Найдите площадь многоугольника.

Входные данные:

- в первой строке целое число N — количество вершин многоугольника; $3 \leqslant N \leqslant 1000$;
- в последующих N строках стоят по два числа типа `real`: x_i, y_i — координаты вершины i.

Данные вводятся из файла `polygon.in` или с клавиатуры (если в многоугольнике не более 10 вершин).

Выходные данные:

- c — результат.

Результат выводится в файл `polygon.out` или на экран.

Пример:

polygon.in	**polygon.out**
3 0.5 0 2.5 3.5 2.5 0	3.5

ОЛИМПИАДЫ 1989/90 УЧЕБНОГО ГОДА

Городская олимпиада

Теоретический тур

1. Множество (10 баллов)

Множество M строится следующим образом:

1) два натуральных числа a, b включены в множество M;
2) для любых x, y, входящих в M, число $x + y + xy$ включается в M;
3) других элементов в M нет.

Опишите алгоритм, определяющий, является ли z элементом множества M, если заданы числа a, b.

2. Минимальное кодирование (10 баллов)

В русском тексте на 1000 символов (букв и пробелов) обычного текста в среднем приходится букв:

а – 62	б – 14	в – 38	г – 13	д – 25	е,ё–72	ж – 7
з – 16	и – 62	й – 10	к – 28	л – 35	м – 26	н – 53
о – 90	п – 23	р – 40	с – 45	т – 53	у – 21	ф – 2
х – 9	ц – 4	ч – 12	ш – 6	щ – 3	ъ,ь–14	ы – 16
э – 3	ю – 6	я – 18				

Пробел встречается 174 раза.

Придумайте кодирование букв последовательностями из нулей и единиц (необязательно одинаковой длины) так, чтобы сообщение однозначно раскодировалось и имело минимальную длину.

3. Двоичное умножение (8 баллов)

Цифры двух положительных двоичных чисел хранятся в двух линейных таблицах. Получите третью таблицу, хранящую их произведение.

4. Клетки в круге (5 баллов)

На клетчатой бумаге нарисовали окружность целого радиуса R с центром на пересечении линий. Найдите количество клеток, целиком лежащих внутри этой окружности.

5. Робот (5 баллов)

На клетчатой бумаге по линиям сетки нарисован прямоугольник. В нем закрашена некоторая связная область, состоящая из клеток. В левом верхнем углу находится Робот, имеющий бортовой вычислитель. Робот может перемещаться в этом замкнутом прямоугольнике и выполнять следующие команды:

- шаг на север;
- шаг на юг;
- шаг на запад;
- шаг на восток;
- +1 (в счетчик бортового вычислителя);
- −1 (в счетчик бортового вычислителя);
- сброс счетчика;
- показать значение счетчика.

Робот умеет вычислять логические функции:

- на севере свободно;
- на севере закрашено;
- на юге свободно;
- на юге закрашено;
- на западе свободно;
- на западе закрашено;
- на востоке свободно;
- на востоке закрашено.

Можно также конструировать сложные логические функции с помощью операций **и**, **или**, **не**.

При составлении алгоритма для робота можно использовать операторы **ЕСЛИ** или **ПОКА** следующего формата:

- **ЕСЛИ** *условие*
  ТО *серия команд 1*
  [**ИНАЧЕ** *серия команд 2*]
  ВСЕ
- **ПОКА** *условие*
  НЦ *серия повторяющихся команд*
  КЦ

Вместо указанных разрешается использовать аналогичные конструкции:

- **if** *<условие>*
  then *<серия команд 1>*
  [**else** *<серия команд 2>*]
  endif

- **while** *<условие>*
  <серия команд>
  endwhile

Граница поля робота представляет собой непроницаемую стену и характеризуется состоянием «не свободно» с соответствующей стороны. Напишите программу, позволяющую роботу вычислить длину границы закрашенной области.

6. Новобранцы (3 балла)

Новобранцы выстроились в ряд, и старшина скомандовал: «Напра-во», после чего каждый повернулся на 90°: кто — направо, а кто — налево. Увидев лицо соседа, новобранец тут же поворачивается на 180°. Так продолжается после каждого поворота. Перестанут ли они когда-нибудь вертеться? Ответ обоснуйте.

7. Фрагмент программы (2 балла)

Что делает данный фрагмент программы?

```
A := A + B
B := B - A
A := A + B
B := -B
```

Практический тур

8. Точка в тетраэдре (12 баллов)

Задана треугольная пирамида (координатами своих вершин) и точка M (x, y, z). Требуется написать программу, проверяющую, находится ли точка M внутри пирамиды.

Указание: уравнение плоскости, проходящей через три заданные точки, имеет вид:

$$[(y2 - y1)(z3 - z1) - (y3 - y1)(z2 - z1)](x - x1) -$$
$$- [(x2 - x1)(z3 - z1) - (x3 - x1)(z2 - z1)](y - y1) +$$
$$+ [(x2 - x1)(y3 - y1) - (x3 - x1)(y2 - y1)](z - z1) = 0.$$

Входные данные:

- x, y, z — координаты точки;
- $x1$, $y1$, $z1$; $x2$, $y2$, $z2$; $x3$, $y3$, $z3$; $x4$, $y4$, $z4$ — координаты четырех вершин пирамиды.

Числа разделены пробелом или символом конца строки.

Данные вводятся из файла `tetra.in` или с клавиатуры.

Выходные данные:

- 1, если точка — внутри или на границе пирамиды, и 0 — в противном случае.

Результат выводится в файл `tetra.out` или на экран.

Пример:

tetra.in	tetra.out
0 0 0 1 0 0 0 1.5 -1 -1 -2 0 1 1 2	1

9. Телеграфный перевод (9 баллов)

Телеграфный перевод оплачивается по 2 коп. за каждый рубль перевода (полный или неполный; например, 1 руб. 01 коп. считается как 2 руб.).

Зарплата, пересылаемая телеграфом, разбивается на сумму перевода и оплату за перевод. Напишите программу, определяющую по зарплате величину перевода. Почта высылает максимально возможную сумму, но никогда не остается с отрицательным балансом.

Входные данные:

- единственное число в формате `РУБ.КОП` вводится из файла `tele.in` или с клавиатуры.

Выходные данные:

- результат — два числа (величина перевода и оплата за перевод), разделенные пробелом, в таком же формате выводятся в файл `tele.out` или на экран.

Пример:

tele.in	tele.out
3.01	2.95 0.06

10. Перестановка (5 баллов)

Напишите программу, выполняющую циклическую перестановку трех чисел, не используя дополнительной памяти.

Пояснения. В программе должны быть описаны только три переменные, введены их значения из файла или с клавиатуры, и после некоторого числа операций эти же переменные должны быть выведены в файл или на экран в том же порядке. Значения переменных должны оказаться циклически сдвинутыми.

Текст программы будет проверяться.

Областная олимпиада

Теоретический тур

11. Мультипроцессорная система (10 баллов)

Четыре процессора объединены в линейку (рис. 3).

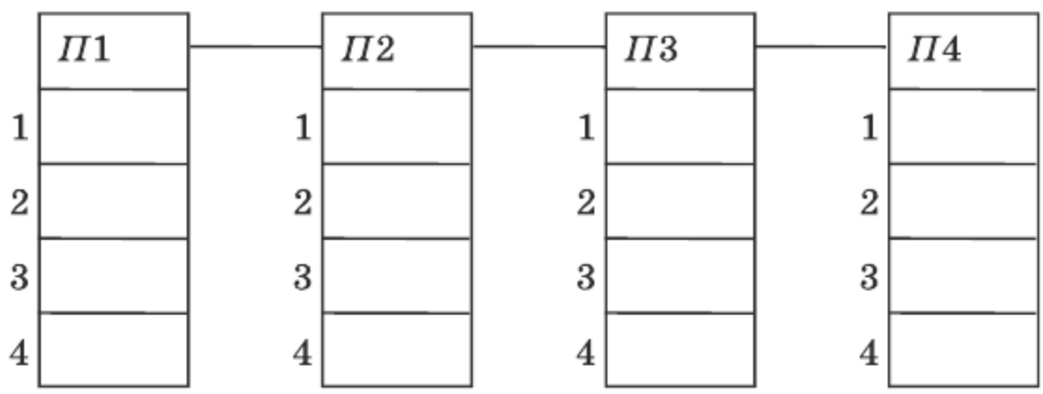


Рис. 3

У каждого процессора есть память, состоящая из четырех занумерованных ячеек для хранения чисел. Все процессоры работают синхронно, каждый по своей программе, выполняя за один такт одну команду из следующей системы команд:

- `A i, j, k` — сложить содержимое i-й ячейки с содержимым j-той ячейки и результат поместить в k-ю ячейку;
- `M i, j, k` — умножить содержимое i-й ячейки на содержимое j-той ячейки и результат поместить в k-ю ячейку;
- `R i, j` — переслать содержимое i-й ячейки в j-ю ячейку процессора, расположенного справа;
- `L i, j` — переслать содержимое i-й ячейки в j-ю ячейку процессора, расположенного слева;
- N — один такт ничего не делать.

Напишите программу для каждого процессора, чтобы результатом их работы являлось значение многочлена: $x^6 + x^5 + ax^4 + bx^3 + cx^2 + dx + e$.

Числа a, b, c, d, e, x заранее разместите в ячейках памяти процессоров (можно — не по одному разу). Система должна работать минимальное количество тактов.

12. Преобразование таблицы (7 баллов)

На квадратном листе клетчатой бумаги размером $N \times N$ в каждой клетке записано число от 0 до 7. Если числа в двух клетках с общей стороной отличаются ровно на 1, то в обе клетки записывается большее из них. Поиск нужной пары клеток осуществляется по следующим правилам: просматриваются все клетки сверху вниз, слева направо, и у каждой проверяется соседняя, сначала справа, затем снизу.

А) Разработайте алгоритм, который осуществляет описанную процедуру до тех пор, пока это возможно (4 балла).

Б) Зависит ли результат от порядка просмотра клеток (1 балл)?

В) Конечен ли этот процесс (2 балла)?

13. Торт (6 баллов)

На квадратном торте стоит N свечей. Можно ли одним прямолинейным разрезом разделить его на две равные по площади части, одна из которых не содержала бы ни одной свечи? Свечи считать точками, каждая из которых задана парой целочисленных координат. Начало координат находится в центре торта, оси координат параллельны краям торта; разрез не может проходить через свечу.

14. Сложение с помощью логических операций (8 баллов)

Составьте программу, которая выполняет сложение двух пятиразрядных двоичных чисел, пользуясь операциями «НЕ», «$\vee$», «$\wedge$» и «$\leftarrow$», которые выполняются так:

1) $c =$ НЕ a — переменной c присваивается значение числа a, в котором нули заменяются единицами, а единицы нулями; вместо НЕ a можно писать $\overline{a}$;
2) $c = a \vee b$ — поразрядная операция, в которой каждый разряд переменной c получает значение, вычисленное по значениям соответствующих разрядов a и b согласно правилам логической операции «ИЛИ», т. е.:

   $0 \vee 0 = 0;\quad 0 \vee 1 = 1;\quad 1 \vee 0 = 1;\quad 1 \vee 1 = 1;$

3) $c = a \wedge b$ — поразрядная операция, в которой каждый разряд переменной c получает значение, вычисленное по значениям соответствующих разрядов a и b согласно правилам логической операции «И», т. е.:

   $0 \wedge 0 = 0;\quad 0 \wedge 1 = 1;\quad 1 \wedge 0 = 1;\quad 1 \wedge 1 = 1;$

4) $c = \leftarrow a$ — поразрядный сдвиг влево, т. е. каждый i-й разряд c получает значение $(i+1)$-го разряда числа a; при этом пятый (младший) разряд c становится равным нулю.

Если результат сложения оказывается более чем пятиразрядным, то его старший разряд теряется.

15. Минимум функции (6 баллов)

$F(x)$ — процедура, вычисляющая значение непрерывной функции, которая определена на всей числовой оси, убывает при всех x, меньших некоторого числа x_0, и возрастает при всех x, больших, чем x_0. Опишите алгоритм, который определяет x_0 с точностью E.

16. Взаимодействие ЭВМ (10 баллов)

ЭВМ1 и ЭВМ2 имеют общий однобитовый регистр R, в который они могут записывать и из которого они могут читать информацию. Их система команд, кроме традиционных, дополнена тремя дополнительными командами:

- `чит(К)` — переменной K присваивается содержимое регистра R;
- `зап(К)` — значение K (0 или 1) записывается в R;
- пауза — задерживает процесс вычислений на 10 тактов.

Все остальные команды выполняются за один такт.

Программа `П2` запускается на ЭВМ2 раньше, чем программа `П1` на ЭВМ1. Во время работы `П1` с клавиатуры ЭВМ1 вводится последовательность двоичных цифр: 1, 0, 1, 1, 1, 0, 0, 1. Что напечатает ЭВМ2?

Ответ обоснуйте.

П1

```
10 ввести А
20 чит(К)
30 зап(1-К)
40 пауза
50 пауза
60 если А=1 то перейти к 90
70 пауза
80 пауза
85 перейти к 10
90 чит(К)
100 зап(1-К)
110 перейти к 70
```

П2

```
10 чит(L)
20 чит(М)
30 если М=L то перейти к 20
40 пауза
50 пауза
60 пауза
70 чит(L)
80 если М=L то перейти к 100
90 печать(1)
95 перейти к 10
100 печать(0)
110 перейти к 10
```

17. Диагональ (5 баллов)

Даны координаты вершин выпуклого N-угольника. Найдите наибольшую по длине диагональ и напечатайте координаты вершин, которые она соединяет.

18. Вершина прямоугольника (3 балла)

Даны координаты трех вершин прямоугольника. Найдите координаты четвертой его вершины.

19. «Странное» свойство ЭВМ (2 балла)

Докажите, что в 16-разрядной ЭВМ с объемом ОЗУ 128 000 байт в любой момент времени существуют две ячейки, содержащие одинаковые числа.

Практический тур

20. Прямоугольники (14 баллов)

Напишите программу, отвечающую на вопрос, можно ли из N заданных прямоугольников сложить один большой прямоугольник данного размера. Решите задачу отдельно для:

А) $N = 2$ (2 балла);
Б) $N = 3$ (4 балла);
В) $N = 4$ (8 баллов).

Входные данные:

- в первой строке находится целое число N;
- в последующих N строках через пробел указаны размеры прямоугольников — целые числа из диапазона от 1 до 32 000.

Данные вводятся из файла `rec.in` или с клавиатуры.

Выходные данные:

- 1, если ответ положительный, 0 — в противном случае.

Результат выводится в файл `rec.out` или на экран.

Пример:

rec.in	rec.out
3 1 1 1 2 3 2	1

21. Фрагменты последовательности (5 баллов)

Имеется последовательность, состоящая из n неповторяющихся чисел. Она задается из файла или с клавиатуры в виде всевозможных пар ее соседних чисел в произвольном порядке, — например, для последовательности 6, 3, 5, 1, 4, 2 такими парами являются (5,1), (4,2), (1,4), (6,3), (3,5).

Напишите программу, которая вводит пары и выводит исходную последовательность.

Входные данные:

- в первой строке находится число N — длина последовательности ($0 < N < 1000$);
- в последующих $N - 1$ строках через пробел указана одна из пар соседних чисел. Все числа — целые из диапазона от 1 до 32 000.

Данные вводятся из файла `seq.in` или с клавиатуры.

Выходные данные.

Результат выводится в файл `seq.out` или на экран.

Пример:

seq.in	seq.out
4 3 2 2 1 4 3	4 3 2 1

22. Шифр Кардано (10 баллов)

Некоторый текст закодирован с помощью шифровальной сетки размером $N \times N$, пример которой представлен на рис. 4. Буквы текста вписаны в клетки квадрата, имеющего тот же размер. Для того, чтобы прочитать текст, нужно наложить на квадрат сетку и выписать символы криптограммы, которые видны через отверстия сетки (светлые квадратики). Первый раз сетка накладывается на текст так, как она изображена на рисунке; затем сетка поворачивается по часовой стрелке на 90°, читается следующий набор символов и т. д. до полного оборота на 360°.

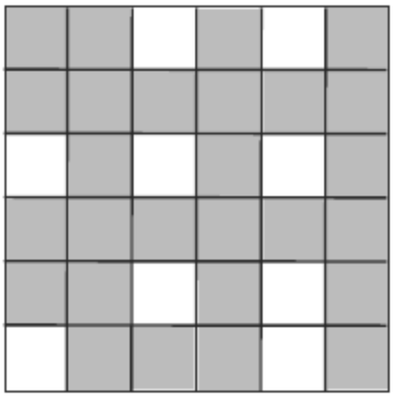


Рис. 4

Напишите программу, которая расшифровывает сообщение, зашифрованное указанным способом.

Входные данные:

- N — натуральное число, не превышающее 50;
- следующие N строк (каждая длины N) содержат информацию о сетке. При этом символы 0 обозначают клетки с отверстием, а символы 1 — остальные клетки. Первая строка соответствует верхней строке сетки;
- следующие N строк (каждая длины N) содержат зашифрованное сообщение.

Данные вводятся из файла `code.in` или с клавиатуры.

Выходные данные.

Расшифрованное сообщение выводится в файл `code.out` или на экран.

Пример:

code.in	code.out
6 110101 111111 010101 111111 110101 011101 екПмоо мв го з дурч дсоНмо асанви лв!ыя	Поздравляем участников с Новым годом!

Городская олимпиада

Теоретический тур

1. Записка Бормана (8 баллов)

К Штирлицу попала закодированная записка Бормана:

15, 16, 16, 16, 16, 4, 5, 8, 31, 25, 20, 2, 19, 18.

Штирлиц знал, что Борман пишет по-русски, используя обычную нумерацию букв в русском алфавите от 1 до 33. Пробел между словами он обозначает номером 0. Также он знал, что Борман кодирует свои сообщения, добавляя к номеру каждой буквы число $X = nA + B$, где n — порядковый номер этой буквы в сообщении, а A и B — константы, известные только Борману. Если результат оказывается больше 33, то из него вычитается 34. Кроме того, Борман ни в одном сообщении не обходится без местоимения «Я». Расшифруйте записку.

2. Сортировка вагонов (10 баллов)

Железнодорожная сортировочная станция имеет следующую схему (рис. 5) и состоит из N путей.

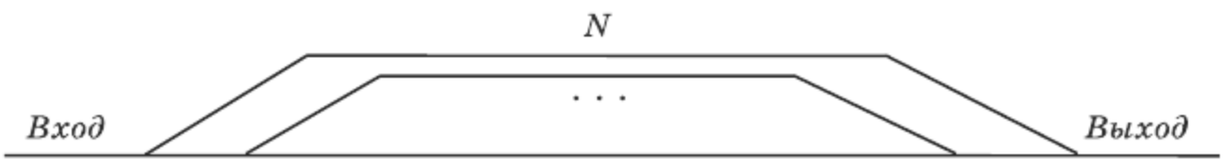


Рис. 5

На вход последовательно подаются вагоны, пронумерованные числами от 1 до M в произвольном порядке. Каждый вагон необходимо направить на один из станционных путей, откуда его затем переводят на выход. На выходе должен образоваться состав с возрастающим порядком номеров вагонов. Опишите алгоритм, который по данным N, M и исходной последовательности номеров вагонов отвечает на вопрос, можно ли выполнить требуемую сортировку.

Замечание. На каждом станционном пути достаточно места для размещения произвольного числа вагонов.

3. Вложенный цикл (6 баллов)

Дан фрагмент программы с параметрами:

```
10 S=0
20 FOR X=A TO B STEP C
30 FOR Y=D TO E STEP F
40 S=S+X+Y
50 NEXT Y
60 NEXT X
```

Какие из следующих шести наборов параметров определяют фрагменты, дающие одинаковое значение S? (K — натуральное число.)

	A	***B***	***C***	***D***	***E***	***F***
1	0	K	1	0	$2(K - X)$	1
2	$2K$	0	-1	$(2K - X)/2$	0	-1
3	K	0	-1	$2(K - X)$	0	-1
4	0	$2K$	1	0	$(2K - X)/2$	1
5	0	$2K$	1	$(2K - X)/2$	0	-1
6	0	K	1	$2(K - X)$	0	-1

4. Корень (10 баллов)

Дано натуральное число n, являющееся квадратом некоторого другого натурального числа. Напишите программу, которая вычисляет $2\sqrt{n}$, используя только операции сложения и вычитания.

5. Четырехугольник и точка (5 баллов)

Заданы декартовы координаты вершин выпуклого четырехугольника в порядке обхода контура и координаты некоторой точки. Напишите программу, которая определяет, лежит ли точка внутри четырехугольника.

6. Фрагмент программы (5 баллов)

Даны два натуральных числа n и b, причем b — нечетное число, большее единицы. Определите, что делает представленный ниже фрагмент программы.

```
нач
  пока n>=b
    нц
     если mod(n,2)=0
      то n:=n/2
      иначе n:=n-b
     все
    кц
   если n=0
    то A:="ДА"
    иначе A:="НЕТ"
   все
кон
```

Практический тур

7. Часы (10 баллов)

Даны два числа ЧЧ и ММ, означающие начальное показание часов (часы и минуты, соответственно: $0 \leqslant$ ЧЧ < 12, $0 \leqslant$ ММ < 60). Определите наименьшее время, округленное до минут, которое должно пройти до того времени, когда часовая и минутная стрелки на циферблате расположатся перпендикулярно друг другу.

Входные данные:

- два целых числа, разделенных пробелом, обозначающие начальное показание часов: первое число — часы, второе — минуты.

Данные вводятся из файла `clocks.in` или с клавиатуры.

Выходные данные:

- одно целое число — результат.

Результат выводится в файл `clocks.out` или на экран.

Пример:

clocks.in	clocks.out
2 50	10

8. Игра (20 баллов)

Двое называют по очереди числа, меньшие N. Начинают с нуля. Каждое новое число должно на 1, 2 или 3 увеличивать одну из цифр предыдущего числа. Проигрывает тот, кто вынужден назвать число $N - 1$. Напишите программу (процедуру одного хода), играющую оптимальным образом в двух случаях:

А) $N = 100$ (5 баллов);
Б) $N = 1000$ (15 баллов).

Требования к оформлению программы.

Заголовок процедуры должен иметь вид:

```
Procedure player<номер игрока>(N,k:integer;
                               var x:integer);
```

где N — ограничение на число сверху, k — текущее число, x — число после вашего хода.

Программы участников будут участвовать в турнире!

Пример:

N, k	x
100 15	18

Областная олимпиада

Теоретический тур

9. Циклы (8 баллов)

При каких целых значениях k следующие фрагменты программ вычисляют одно и то же значение y?

1)
```
y:=0
нц
для x от (k+6)/2 до (2k+16)/2 шаг 1
    y:=y+ABS(x)
кц
```

2)
```
y:=0
нц
для x от (2k+16)/2 до (k+6)/2 шаг -1
    y:=y+ABS(x)
кц
```

10. Письмо (10 баллов)

Составьте алгоритм, выясняющий, можно ли прямоугольную открытку вложить в конверт, если размеры открытки x, y и конверта a, b заданы.

11. Пересечение отрезков (3 балла)

Даны координаты концов двух отрезков на плоскости. Напишите программу, которая проверяет, пересекаются ли данные отрезки.

12. Скобки (8 баллов)

Дана конечная последовательность левых и правых скобок k типов (круглые, квадратные и др.). Составьте программу, проверяющую, можно ли в эту последовательность добавить цифры и знаки арифметических действий так, чтобы получилось правильное арифметическое выражение.

13. Игра «Даты» (8 баллов)

Играют двое. Первый игрок сообщает какую-нибудь дату января 1991 г. Каждый игрок своим ходом называет более позднюю дату, увеличивая либо календарную дату в месяце, либо месяц, но не то и другое сразу. Опишите выигрышную стратегию, при которой:

А) игрок, назвавший 31 декабря, выигрывает (3 балла);
Б) игрок, назвавший 31 декабря, проигрывает (5 баллов).

14. Алгоритм (4 балла)

Что делает этот алгоритм?

```
Алг УПР(нат a,b,c)
   арг a,b рез c
нач нат k, e
   k:=a*b
   e:=M(a,b)
   c:=0
   нц пока k>=e
    k:=k-e
    c:=c+1
   кц
кон
```

```
Алг нат М(нат x,y)
нач
 нц пока x<>y
     если x>y
      то x:=x-y
      иначе y:=y-x
     все
 кц
 знач:=x
кон
```

15. Последовательность (10 баллов)

В последовательности цифр a_1, a_2, a_3, ... каждый член, начиная с четвертого, равен последней цифре суммы трех предыдущих. Опишите алгоритм, который по заданным a_1, a_2 и a_3 определяет:

А) a_{1991} (3 балла);

Б) $a_{1000000000}$ (7 баллов).

Алгоритм должен иметь возможно меньшую трудоемкость.

16. Многочлен (10 баллов)

В ЭВМ вводятся последовательно число x и коэффициенты многочлена, начиная со старшего, причем порядок многочлена заранее неизвестен. Напишите программу, не использующую таблиц, которая вычисляет:

А) значение многочлена в точке x (3 балла);

Б) значение производной многочлена в этой же точке (7 баллов).

Практический тур

17. Игра «7» (20 баллов)

На доске 3×3 произвольным образом расставлены фишки, пронумерованные от 1 до 7. За один ход можно передвинуть фишку на соседнее свободное поле по горизонтали или по

вертикали. Напишите программу, приводящую фишки в конечную позицию:

1	2	3
4	5	6
7		

Входные данные:

- три строки по три числа в каждой, разделенные пробелом — исходное расположение фишек на доске; число «0» обозначает отсутствие фишки в данной ячейке доски.

Данные вводятся из файла `seven.in` или с клавиатуры.

Выходные данные:

- результатом является последовательность ходов, отображаемая следующим образом: в i-й строке содержится четыре числа $(x1, y1)$ — $(x2, y2)$, попарно заключенные в скобки и означающие координаты положения фишки до и после выполнения i-го хода. Числа в скобках разделены пробелом, а скобки — дефисом.

Результат выводится в файл `seven.out` или на экран.

Пример:

seven.in	**seven.out**
1 2 3 4 5 7 6 0 0	(3 1)-(3 2) (2 1)-(3 1) (2 2)-(2 1) (3 2)-(2 2) (2 3)-(3 3) (2 2)-(2 3) (2 1)-(2 2) (3 1)-(2 1) (3 3)-(3 2) (3 2)-(3 1)

18. Имена (12 баллов)

Напишите программу, которая читает имя и печатает уменьшительно-ласкательный его вариант, — например, «Ваня» — «Ванюша» (или «Ванечка») и т. п.

Входные данные:

- строка символов, содержащая имя.

Данные вводятся из файла `names.in` или с клавиатуры.

Выходные данные:

- строка символов, содержащая измененное имя.

Результат выводится в файл `names.out` или на экран.

Пример:

names.in	**names.out**
Ваня	Ванюша

Городская олимпиада

Теоретический тур

1. Бильярд (10 баллов)

Дано бильярдное поле размером $a \times b$ с единственной лузой, расположенной в левом нижнем углу, совпадающем с началом координат. (X, Y) — координаты бильярдного шара на поле. Опишите алгоритм нахождения направлений удара по шару, обеспечивающих попадание в лузу при:

А) одном отражении от борта (2 балла);

Б) двух отражениях от бортов (4 балла);

В) n отражениях от бортов (4 балла).

Считается, что отскок шара от борта происходит по закону «угол падения равен углу отражения».

2. Роман (12 баллов)

В романе N глав. В i-й главе a_i страниц. Требуется издать роман в K томах так, чтобы объем самого толстого тома был минимален. Опишите алгоритм, отвечающий на вопрос, каким будет толщина самого толстого тома. Делить главы нельзя.

3. Польский калькулятор (7 баллов)

Вместо обычной скобочной записи арифметических выражений можно использовать бесскобочную (польскую), предложенную польским математиком Яном Лукасевичем. В ней знаки операций ставятся сразу за своими операндами и вычисление идет слева направо, например:

Обычная запись	Польская запись
`3*(11-1)`	`3 11 1 - *`
`2+10`	`2 10 +`
`3*11-1`	`3 11 * 1 -`
`((2+2)-555)*(9/99)`	`2 2 + 555 - 9 99 / *`

Разделителем операндов служит пробел. Напишите программу, вычисляющую значения заданного бесскобочного выражения. Операнды — целые положительные числа. Операции: +, –, *, /. Исходное выражение вводится в виде строки.

4. Список (6 баллов)

Имеется список учеников в классе. Перед каждой фамилией стоит порядковый номер. После фамилии стоят два числа: порядковый номер того ученика, который непосредственно предшествует данному ученику в алфавитном списке класса (у первого по алфавиту — 0), и порядковый номер ученика, следующего за ним в алфавитном списке (у последнего по алфавиту — 0).

Пример:

```
1. Иванов 0 3
2. Сидоров 3 0
3. Петров 1 2
```

А) Как в такой список включить новичка (3 балла)?

Б) Как исключить из списка k-го по алфавиту (3 балла)?

5. Билеты (5 баллов)

У кассира имеется план зрительного зала, состоящего из k рядов. Первый ряд имеет m мест, а каждый последующий — на два места больше предыдущего. Проданные места в плане вычеркиваются. Напишите программу поиска трех свободных мест, идущих подряд в одном ряду.

6. Алгоритм (4 балла)

Дан алгоритм:

```
Алг вещ P(вещ a,b)
нач вещ c
c:=(b-a)/3 + a
если ABS(b-a)<0.0001
   то знач:=c
    иначе
      если sin(a)*sin(c)<0
       то знач:=P(a,c)
       иначе знач:=P(c,b)
      все
все
кон
```

Чему будут равны $x1$ и $x2$, если:

А) $x1=\boldsymbol{P}(1,2)$ (2 балла);
Б) $x2=\boldsymbol{P}(3,4)$ (2 балла).

Практический тур

7. Умножение «столбиком» (9 баллов)

Напишите программу, которая выводит картинку, изображающую умножение «*столбиком*» двух заданных чисел.

Входные данные:

- два вещественных числа, расположеных в двух строках входного файла (если ввод производится не с клавиатуры).

Данные вводятся из файла `multiplication.in` или с клавиатуры.

Выходные данные.

Результат выводится в файл `multiplication.out` или на экран.

Пример:

multiplication.in	multiplication.out
398.56 85.03	398.56 85.03 -------- 11.9568 1992.80 31884.8 ---------- 33889.5568

8. Кроссворд (15 баллов)

Даны четыре слова. Напишите программу, проверяющую, можно ли из данных слов составить кроссворд при условии, что каждое слово пересекается с двумя другими и располагается сверху вниз или слева направо. Сетка не обязательно симметрична. Результат выведите в виде кроссворда (в противном случае решение оценивается максимум в 10 баллов).

Входные данные:

- четыре строки символов (не более 20 символов каждая), расположенные в четырех строках входного файла.

Данные вводятся из файла `crossword.in` или с клавиатуры.

Выходные данные.

Изображение кроссворда выводится в файл `crossword.out` (пустым клеткам соответствуют пробелы) или на экран. Если нужного кроссворда не существует, то вывести −1.

Пример:

crossword.in	crossword.out
математика информатика физика химия	и н ф о р м а х т ф и з и к а м к м а т е м а т и к а я

Областная олимпиада

Теоретический тур

9. Порядок (4 балла)

Пусть S — множество всех рациональных чисел, представимых в виде несократимых дробей вида m/n, где $m + n \leqslant 100$, $n > 0$, $m > 0$. Эти дроби можно упорядочить по возрастанию или убыванию, но можно ввести и другой порядок. Будем считать, что $m1/n1$ предшествует $m2/n2$, если $n1 + m1 < n2 + m2$,

или $n1 + m1 = n2 + m2$ и $m1 < m2$. Тогда элементы множества S расположатся так:

1/1, 1/2, 2/1, 1/3, 3/1, 1/4, 2/3, 3/2, ..., 1/99, 3/97, ..., 99/1.

Напишите программу, порождающую последовательно в соответствии с данным порядком все дроби множества S.

10. Бензовоз (5 баллов)

Бензовоз должен доставить горючее из пункта A в пункт B, расстояние между которыми равно d миль. Запас горючего в пункте A равен V литров. Бензовоз расходует 1 литр на одну милю, а его бак вместе с цистерной вмещает 1000 литров.

Разработайте алгоритм, который отвечает на вопрос, какое максимальное количество бензина можно перевезти из A в B, если:

А) промежуточных пунктов для хранения бензина между A и B нет (2 балла);

Б) в пункте C на расстоянии x по пути от A до B стоит пустая цистерна, куда предварительно можно завезти некоторый запас бензина, а затем его использовать (3 балла).

11. Слово (5 баллов)

Задается некоторое слово (например «ФАРА»). Из всех его букв составляются различные другие слова, возможно, бессмысленные («АРФА», «ФААР» и т. д.). Опишите алгоритм, который по заданному слову из этого набора строит следующее за ним по алфавиту слово из этого же набора.

12. Схема (6 баллов)

Схема, преобразующая входные сигналы в выходной, может состоять из последовательных и параллельных соединений элементов двух типов (рис. 6 *а*, *б*):

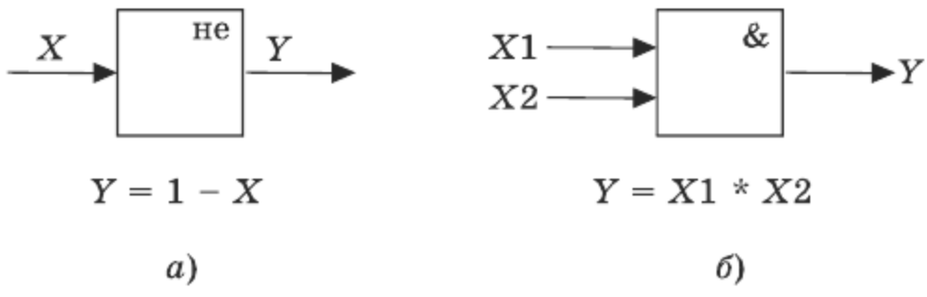


Рис. 6

Входные сигналы X, $X1$ и $X2$ (а значит, и выходной Y) принимают значения 0 или 1.

Выходной сигнал меняется при изменении входного сигнала не сразу, а через время, равное 1 секунде. Составьте из таких элементов схему с одним входом x и одним выходом y, которая после изменения x с 0 на 1 через некоторое время в течение 3 секунд выдавала бы сигнал $y = 1$ (рис. 7).

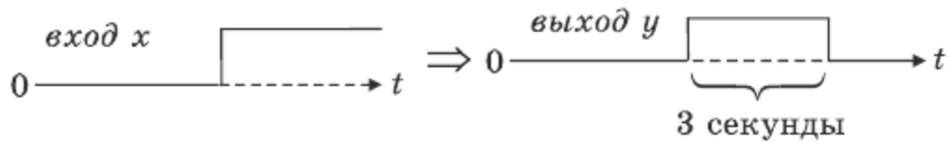


Рис. 7

13. Польская запись (6 баллов)

Вместо обычной скобочной записи арифметических выражений можно использовать бесскобочную (обратную польскую). В ней знаки операций ставятся сразу за своими операндами, а вычисление идет слева направо, например:

Обычная запись	Польская запись
2+10	2 10 +
3*11-1	3 11 * 1 -
1+2*3-4/5	1 2 3 * + 4 5/-

Опишите алгоритм преобразования обычной записи арифметического выражения с операциями +, –, *, / в польскую. Операнды — положительные целые числа. Разделителем операндов и операций в польской записи служит пробел.

14. Алгоритм (8 баллов)

Задана таблица чисел. Алгоритм на каждом шаге выбирает в этой таблице строку (столбец), сумма элементов которой (которого) меньше нуля, и умножает все эти элементы на –1. Докажите, что такой алгоритм не может работать бесконечно долго.

15. Спички (10 баллов)

На столе в нескольких кучках лежат 45 спичек. За один ход разрешается взять по одной спичке из каждой кучки и из них образовать новую кучку. Как с помощью ЭВМ можно доказать или опровергнуть следующее утверждение: в какой-то момент на столе окажется 9 кучек с числом спичек 1, 2, 3, 4, 5, 6, 7, 8 и 9?

Практический тур

16. Числительные (16 баллов)

Напишите программу, переводящую заданную сумму в рублях (до 30 000 включительно) в ее полное название по правилам русского языка. Сокращать слова нельзя.

Входные данные:

- одно число вещественного типа, содержащее не более двух знаков после запятой.

Данные вводятся из файла `numeral.in` или с клавиатуры.

Выходные данные:

- строка символов — ответ на вопрос задачи.

Результат выводится в файл `numeral.out` или на экран.

Пример:

numeral.in	numeral.out
29320.25	двадцать девять тысяч триста двадцать рублей двадцать пять копеек

17. Степень двойки (18 баллов)

Рассматривается десятичная запись натуральных чисел n, $n + 1$ и 2^n. Составьте программу, находящую в течение заданного времени тестирования как можно больше таких n, для каждого из которых выполняется условие: десятичная запись числа n или числа $n + 1$ совпадает с концом десятичной записи числа 2^n. Например, для $n = 7$ конец записи числа $2^7 = 128$ совпадает с $n + 1 = 8$.

Выходные данные.

Результат выводится на экран или в файл `degree.out`; каждое число — в новой строке.

18. Цепочка (23 балла)

В двузначном числе за один ход разрешается заменить любую цифру на сумму его цифр по модулю 10. Заданы два двузначных числа a и b. Напишите программу, которая определяет:

А) можно ли построить цепочку ходов, которая переводит a в b?

Б) минимальную такую цепочку.

В двузначном числе старшая цифра может быть и нулем.

Входные данные:

- a, b — два целых числа, разделенных пробелом.

Данные вводятся из файла `chain.in` или с клавиатуры.

Выходные данные:

- ответ на вопрос задачи в формате «ДА»/«НЕТ».
- последовательность двузначных чисел через пробел, задающих цепочку, переводящую a в b.

Результат выводится на экран или в файл `chain.out`.

chain.in	chain.out
15 37	ДА 15 65 61 67 37

ОЛИМПИАДЫ 1992/93 УЧЕБНОГО ГОДА

Городская олимпиада

Теоретический тур

1. Лидеры (8 баллов)

N спортсменов уходят со старта в моменты времени $t_1 < t_2 < \ldots$ $\ldots < t_n$ и движутся по дистанции с постоянными скоростями v_1, v_2, ..., v_n. Опишите алгоритм, определяющий номера спортсменов, которые к моменту времени t успеют побывать в роли единоличных лидеров.

2. Циклы (7 баллов)

Программа на Бейсике имеет вид:

```
1 GOTO L1
2 GOTO L2
. . . .
N GOTO LN
```

Все L1, L2, ... , LN различны и лежат в интервале от 1 до N. Напишите алгоритм, который по заданным L1, L2, ... , LN определяет число непересекающихся замкнутых циклов в программе.

Примеры:

1)
```
1 GOTO 3
2 GOTO 2
3 GOTO 1
```

Ответ: 2 цикла.

2)
```
1 GOTO 3
2 GOTO 1
3 GOTO 2
```

Ответ: 1 цикл.

3. Фестиваль (8 баллов)

В клубе любителей кино состоят N членов. Ежегодно на Международный кинофестиваль в Канны посылается k членов киноклуба. Одна делегация не должна ездить дважды, т. е. каждый год состав делегации должен хотя бы на одного человека отличаться от состава любой предыдущей делегации. Предложите алгоритм, порождающий списки всех возможных делегаций.

4. Программа (4 балла)

Что делает эта программа?

```
Алг X(вещ a,y, цел n)
 арг a,n
 рез y
нач цел k,вещ t,b
 b:=1
 t:=a
 k:=n
 нц пока k<>0
     если mod(k,2)=0
       то t:=t*t
          k:=k/2
       иначе b:=b*t
              k:=k-1
     все
  кц
   y:=b
кон
```

5. Сумма квадратов (6 баллов)

Дано натуральное число n. Найдите шестикратную сумму квадратов чисел от 1 до n за возможно меньшее количество арифметических операций.

6. Треугольник и точки (5 баллов)

Даны координаты вершин треугольника на плоскости. Опишите алгоритм, определяющий количество точек с целочисленными координатами, попавших внутрь треугольника.

7. Робот (5 баллов)

Робот находится в клетке поля, по которому он может перемещаться в любом из четырех направлений, каждый раз переходя в соседнюю клетку. В одной из клеток поля находится обломок ядерного реактора, который роботу нужно найти и обезвредить. После каждого шага он может определить по уровню радиации, ближе или дальше он от цели поиска, чем был на предыдущем шаге. Опишите алгоритм поведения робота, обеспечивающий возможно более короткий путь к цели.

Практический тур

8. Минимум (15 баллов)

Даны координаты трех точек на плоскости. Найдите координаты точки плоскости, такой, что:

А) сумма расстояний от нее до данных точек минимальна (10 баллов);

Б) сумма квадратов расстояний от нее до данных точек минимальна (5 баллов).

Входные данные:

- в трех строках по два числа через пробел — координаты точек.

Данные вводятся из файла `minimum.in` или с клавиатуры.

Выходные данные:

- в первой строке два числа через пробел — координаты точки, удовлетворяющей условию А;
- во второй строке два числа через пробел — координаты точки, удовлетворяющей условию Б.

Результат (с точностью до 4 знаков после запятой) выводится в файл `minimum.out` или на экран.

Пример:

minimum.in	minimum.out
0 0 4 0 2 3	2 1.1547 2 1

9. Путь коня (12 баллов)

На шахматной доске $N \times N$ с несколькими вырезанными клетками заданы две клетки. Постройте, если это возможно, минимальный путь коня из одной заданной клетки в другую.

Входные данные:

- целое число N, задающее размер шахматной доски;
- в следующих N строках указаны по N чисел (нулей или единиц) через пробел: единица означает, что в соответствующей горизонтали клетка вырезана, нуль — не вырезана;
- в следующих двух строках расположены по два числа — координаты начального и конечного положения коня.

Данные вводятся из файла `way.in` или с клавиатуры.

Выходные данные:

- в первой строке ответ на вопрос задачи в формате «ДА»/«НЕТ».
- в следующих строках по два числа — координаты клеток пути.

Результат выводится в файл `way.out` или на экран.

Пример:

way.in	way.out
4 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 4	Да 1 1 2 3 4 2 3 4 2 2 1 4

Областная олимпиада

Теоретический тур

10. Алгоритм (4 балла)

Литерная строка *a* составлена только из нулей, единиц и двоек. Определите, что делает алгоритм:

```
Алг АЛГОРИТМ(лит a, цел n)
арг a
рез n
нач цел i
    n:=0
    нц для i от 1 до длин(a)
      n:=n*3
      выбор
         при a[i:i] ="1": n:=n+1
         при a[i:i] ="2": n:=n+2
      все
    кц
кон
```

11. Таблица (10 баллов)

Дана таблица $N \times M$, в клетках которой произвольным образом проставлены числа от 1 до NM. «Горизонтальным ходом» называется такая перестановка любого числа элементов таблицы, при которой каждый элемент остается в той строке, в которой он и был. «Вертикальным ходом» называется такая перестановка любого числа элементов таблицы, при которой каждый элемент остается в том столбце, в котором он и был.

Составьте алгоритм, который за возможно меньшее количество ходов расставляет элементы таблицы по порядку, т. е. в первой строке — от 1 до M, во второй строке — от $M + 1$ до $2M$ и т. д.

12. Маршрут (10 баллов)

Клетки таблицы $N \times N$, где $N < 3$, заполнены случайным образом цифрами от 0 до 9. Опишите алгоритм, позволяющий

найти маршрут (последовательность клеток) из клетки (1, 1) в клетку (N, N), удовлетворяющий следующим условиям:

1) любые две последовательные клетки в маршруте имеют общую сторону;
2) количество клеток маршрута минимально;
3) сумма цифр в клетках маршрута максимальна.

13. Триангуляция (6 баллов)

Даны координаты вершин квадрата на плоскости. Внутри квадрата имеется N точек. Рассматриваются эти точки вместе с вершинами квадрата. Известно, что никакие три из них не лежат на одной прямой. Опишите алгоритм разбиения квадрата на непересекающиеся треугольники с вершинами в данных точках.

14. Клавиатура (6 баллов)

Имеется прямоугольная клавиатура 4×8 и два регистра W и R (рис. 8). Если в i-м разряде W стоит 1, то на i-ю строку клавиатуры поступает сигнал. Если в j-м столбце этой строки нажата клавиша, то сигнал проходит в j-й разряд регистра R и устанавливает его в 1. Разряд устанавливается в нуль, когда клавиша отпускается. (Например, если $W = 3$ и нажата клавиша «К», то $R = 4$).

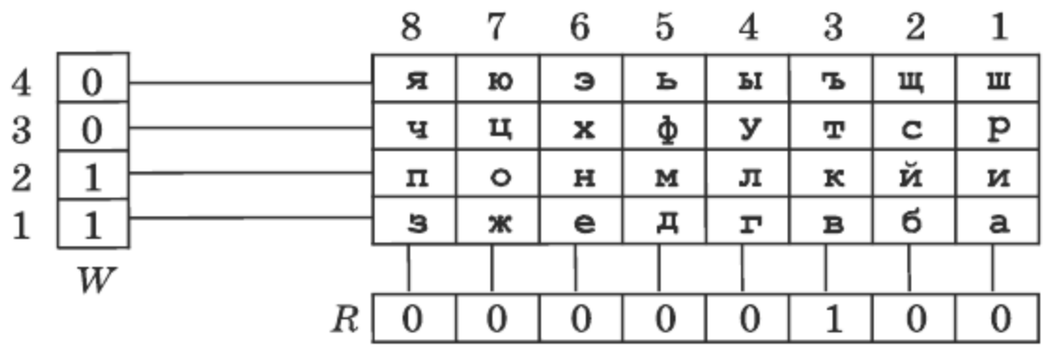


Рис. 8

Напишите программу, которая, записывая числа в регистр W и считывая числа из регистра R, выводит на печать букву нажатой клавиши.

15. Родственники (5 баллов)

На планете Глюк живет группа людей. Про некоторые пары людей известно, что они близкие родственники. Назовем A и B родственниками, если A и B близкие родственники или если найдется третий человек C, который по отдельности является родственником A и родственником B. Опишите алгоритм нахождения всех родственников человека X.

16. Схема (5 баллов)

Переключатель телефонных линий может находиться в одном из двух состояний, представленных на рис. 9.

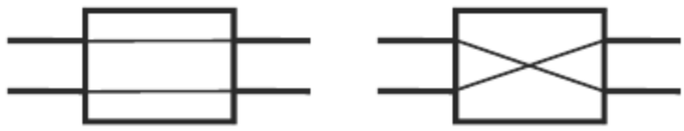


Рис. 9

Нарисуйте схему коммутатора, состоящего из возможно меньшего количества переключателей, обеспечивающего одновременное ведение четырех междугородних разговоров между четырьмя абонентами из одного города и четырьмя абонентами из другого в произвольном сочетании.

Практический тур

17. Родословная (15 баллов)

В Голландии каждому новорожденному теленку присваивается уникальный номер. В паспорте теленка записываются его номер, затем номер отца, а затем номер матери (например: `5 2 8`). Напишите программу, которая вводит информацию из N паспортов и проверяет ее непротиворечивость (например, теленок не может быть своим собственным отцом). Программа должна сообщать, что данные непротиворечивы, или указывать конкретное противоречие.

Входные данные:

- в первой строке — целое число N — количество паспортов ($N \leqslant 100$);
- далее в каждой из N строк через пробел указаны по три числа — номер теленка, номер отца, номер матери.

Данные вводятся из файла `familytree.in` или с клавиатуры.

Выходные данные:

- в первой строке ответ на вопрос задачи в формате «ДА»/«НЕТ»;
- если ответ на вопрос задачи отрицательный, то в следующей строке указывается суть противоречия.

Результат выводится в файл `familytree.out` или на экран.

Пример:

familytree.in	familytree.out
4 1 2 3 2 3 4 3 8 5 4 6 7	Нет Противоречие пола третьего теленка.

18. Параллельные вычисления (15 баллов)

Используя n сумматоров, работающих синхронно, необходимо вычислить 16 частичных сумм S_k чисел a_1, a_2, ... , a_{16}, где $S_k = a_1 + a_2 + \ldots + a_k$, $k = 1, 2, \ldots, 16$. Каждый сумматор складывает два числа за 1 такт. Напишите программу вычисления всех частичных сумм за возможно меньшее количество тактов для: $n = 2$; $n = 4$; $n = 8$; $n = 12$.

Для решения задачи предлагается интерпретатор команд данной параллельной системы. Правила работы с интерпретатором приводятся ниже. Задается имя программы и количество сумматоров.

Список управляющих директив:

- `H` — помощь;
- `N` — начать работу с новой программой;
- `Q` — выход;
- `R` — запуск программы;
- `P` — вывести на экран текст программы;

- `S` — сохранить текст программы под введенным в начале именем;
- `L` — загрузить текст программы;
- `M` — вывести на экран переменные $R1$–$R80$.

Программа набирается в виде следующих команд:

```
I J Z=X+Y
```

где I — номер сумматора, а J — номер такта (от 1 до 16).

Слагаемые X, Y и результат Z должны иметь вид: Ri, где i — целое от 1 до 80, причем $R1$–$R16$ содержат заданные числа, которые запрещено изменять.

Пример:

`2 1 R20=R1+R2` — команда, выполняемая вторым сумматором на первом такте, которая суммирует первые два исходных числа и помещает результат в $R20$.

Городская олимпиада

Теоретический тур

1. Перестановки (6 баллов)

Алгоритм работает следующим образом: в линейной числовой таблице длины N ищется пара соседних элементов, в которой первый элемент больше второго, и если такая пара находится, то элементы меняются местами. Затем поиск повторяется. Алгоритм заканчивается, если искомых пар нет.

А) Обоснуйте конечность такого алгоритма.

Б) Какое максимальное количество перестановок может быть выполнено?

2. Химия (8 баллов)

Задано N веществ и таблица их взаимодействий, т. е. $a[i, j] = 0$, если i-е вещество не взаимодействует с j-м веществом и $a[i, j] = k$, $(1 \leqslant i, k \leqslant N)$, если при их взаимодействии получается k-е вещество. В пробирку одно за другим в заданной последовательности засыпаются вещества. Оказавшись рядом, они могут вступить в реакцию. Вновь образованное вещество, возможно, реагирует с нижележащим и т. д. Известно, что реакция взаимодействия происходит мгновенно и только между двумя соседними слоями. Опишите алгоритм, определяющий, какие вещества останутся в пробирке.

3. Фибоначчи (6 баллов)

Для вычисления N-го члена ряда Фибоначчи $F(N)$ используется следующий рекурсивный алгоритм:

```
алг цел F(нат N)
нач
  если N<3
    то знач:=1
    иначе знач:= F(N-1)+F(N-2)
  все
кон
```

Сколько раз этот алгоритм обратится к самому себе при вычислении $F(20)$?

4. Вычисление (3 балла)

Что вычисляет данный алгоритм?

```
алг ЗАДАЧА(цел n, вещ S)
арг n          |n - большое число
рез S
нач
   S:=0
   нц n раз
      x:=rnd(1)
      y:=rnd(1)
      если x*x+y*y<1
        то S:=S+1
      все
   кц
   S:=S/n
кон
```

5. Дни в месяце (8 баллов)

Составьте арифметическое выражение, которое за возможно меньшее число операций вычисляет по заданному номеру месяца количество дней в нем. Операцией является любая арифметическая операция или встроенная функция. Считать, что в феврале 28 дней.

6. Признак делимости (9 баллов)

Дана строка символов, являющаяся числом в 16-ричной системе. Опишите алгоритм, проверяющий, делится ли данное число на 5.

7. Кольцевая дорога (10 баллов)

Мэрия строит окружную кольцевую дорогу вокруг населенного пункта. Считая дома и другие городские строения точками на плоскости, опишите алгоритм построения дороги минимального радиуса, охватывающей все такие точки.

Практический тур

8. Точки в круге (12 баллов)

Напишите программу, которая по данному числу N выдает координаты центра и радиус круга, содержащего в себе ровно N целочисленных точек, т. е. точек с целыми координатами на плоскости.

Входные данные:

- одно целое число — количество точек в круге.

Данные вводятся из файла `points.in` или с клавиатуры.

Выходные данные:

- в первой строке два числа — координаты центра круга;
- во второй строке одно число — радиус круга.

Результат выводится в файл `points.out` или на экран.

Пример:

points.in	points.out
2	0.5 0 0.6

9. Ребусы (15 баллов)

Решите с помощью компьютера математические ребусы:

```
ДВА*ДВА=ЧЕТЫРЕ
ТРИ*ТРИ=ДЕВЯТЬ
ДВА*ТРИ=ШЕСТЬ
```

В каждом из них разным буквам соответствуют разные цифры, а одинаковым буквам — одинаковые цифры.

Выходные данные.

По каждому ребусу надо дать один из следующих ответов:

- «Решений нет»;
- «Решение не найдено»;
- список найденных решений + «Других решений нет»;
- список найденных решений + «Других решений не найдено».

Результат выводится в файл `output.txt` или на экран.

Областная олимпиада

Теоретический тур

10. Лототрон (7 баллов)

В лототроне имеется n шаров, каждый из которых помечен натуральным числом. Каждую секунду из лототрона вываливаются 2 шара. Если числа на них одинаковые, то один из них удаляется, а второй возвращается назад. Если числа различны, то шар с большим числом помечается разностью данных чисел и оба шара возвращаются в лототрон. Докажите, что через конечное время в лототроне останется только один шар. Какое число окажется на этом шаре?

11. Латинский квадрат (10 баллов)

«Латинским квадратом» порядка n называется таблица $n \times n$, каждая строка и каждый столбец которой содержит все числа от 1 до n. Опишите алгоритм, достраивающий латинский квадрат порядка n по первым его k строкам.

12. Дешифровка (7 баллов)

Задана некоторая перестановка чисел от 1 до n. Она шифруется так: каждое исходное число i заменяется на число, равное количеству чисел, больших i и стоящих в перестановке правее него. Определите условие, при котором данная последовательность n чисел является шифром некоторой перестановки, и опишите алгоритм дешифровки.

13. Кулинария (6 баллов)

Для приготовления любого блюда необходимо выполнить n операций (помыть, посолить и т. д.). Существуют некоторые ограничения на порядок выполнения этих операций. Например, мыть мясо надо раньше, чем варить, а нарезать лук и нарезать морковь можно в любом порядке. Имея эти ограничения, необходимо составить последовательность выполнения операций для приготовления блюда.

А) Разработайте формальную модель задачи.
Б) Исследуйте вопрос существования решения.
В) Опишите алгоритм решения задачи.

14. Алгоритм (3 балла)

Что делает данный алгоритм?

```
Алг цел F(цел a,b)
нач
  выбор
    при b>0: знач:=F(a+1,b-1)
    при b=0: знач:=a
    при b<0: знач:=F(a-1,b+1)
  все
кон
```

15. Джонсон & Джонсон (9 баллов)

Имеется N деталей, каждая из которых должна пройти обработку на двух станках: сначала на токарном, а затем на фрезерном. Для каждой i-й детали задано время обработки на каждом станке — t_i и f_i, соответственно. Опишите алгоритм, строящий последовательность обработки деталей за минимальное общее время.

16. Подстрока (7 баллов)

Составьте рекурсивный алгоритм, определяющий, является ли:

А) заданная строка B — началом заданной строки A (3 балла);

Б) строка B — подстрокой строки A (4 балла).

При построении алгоритма разрешается выполнять следующие операции над строками:

- сравнение;
- присваивание;
- разбиение строки на первый символ и остаток:

```
ПС(<строка>,<первый символ>,<остаток>).
```

Практический тур

17. Игра (16 баллов)

Двое («белые» и «черные») по очереди вписывают в произвольную клетку таблицы 3×3 одно из целых чисел от 1 до 9 (числа не должны повторяться). Начинают «белые». После заполнения таблицы подсчитываются два числа: $S1$ — сумма произведений элементов каждой строки и $S2$ — сумма произведений элементов каждого столбца.

Если $S1 > S2$, считается, что выиграли «белые», а если $S1 < S2$, то «черные». Подсчитывается также количество выигранных очков $|S1 - S2|$.

Напишите программу, делающую очередной ход.

Среди программ участников и жюри будет проведен турнир.

Входные данные:

- три строки, в каждой по три целых числа через пробел — текущее состояние таблицы. Нуль соответствует пустой клетке.

Данные вводятся из файла `game.in` или с клавиатуры.

Выходные данные:

- три строки, в каждой по три целых числа через пробел — состояние таблицы после хода.

Результат выводится в файл `game.out` или на экран.

Пример:

game.in	game.out
1 3 7 4 0 6 0 9 2	1 3 7 4 5 6 0 9 2

18. Сумма простых (16 баллов)

Задаются натуральные числа A и N ($A < 1000$, $N < 10$). Напишите программу, которая подсчитывает количество способов (с учетом порядка), которыми можно представить число A в виде суммы N простых чисел, и приводит пример одного такого разложения.

Входные данные:

- в первой строке — целое число A;
- во второй строке — целое число N.

Данные вводятся из файла `sum.in` или с клавиатуры.

Выходные данные:

- в первой строке целое число — ответ на вопрос задачи;
- во второй строке строка символов — пример одного разложения (если таковое есть).

Результат выводится в файл `sum.out` или на экран.

Пример:

sum.in	sum.out
10 2	3 10=3+7

В этом примере подсчитаны разложения: `10=3+7`, `10=5+5` и `10=7+3`.

ОЛИМПИАДЫ 1994/95 УЧЕБНОГО ГОДА

Городская олимпиада

Теоретический тур

1. Капитал (6 баллов)

Некая фирма ежедневно каждую целую тысячу рублей суммы, имеющуюся в данный момент на счету, увеличивает на k рублей. Напишите программу, определяющую дату ($d2$), когда сумма станет не меньше заданной величины в m рублей. Начальный вклад был сделан в день $d1$ и равнялся n рублей. В день открытия счета сумма не увеличивается. Дата задается в виде `дд.мм.гг` — например `25.03.94`. Программа должна сопровождаться комментариями.

2. Вирус (8 баллов)

На винчестере хранились 12 файлов одинаковой длины с именами `F1`, `F2`, ..., `F12`. После заражения вирусом длина одного из них изменилась, но неизвестно, в какую сторону. Программист может сравнивать длины файлов и образовывать новые слиянием двух или нескольких файлов в один (длина нового файла равна сумме длин сливаемых).

Опишите алгоритм, позволяющий определить испорченный файл всего за три сравнения. Количество слияний файлов может быть произвольным.

3. Библиотека алгоритмов (5 баллов)

Пусть имеется следующий набор вспомогательных алгоритмов для работы с линейной таблицей:

- **Алг** `вещ` ***MAX*** `(цел` n`, вещтаб` X`[1:`n`])`, который принимает значение максимального элемента в таблице X;
- **Алг** `вещ` ***MAXMOD*** `(цел` n`, вещтаб` X`[1:`n`])`, который принимает значение максимального из модулей элементов таблицы;

- **Алг** вещ ***SUM*** (цел *n*, вещтаб *X*[1:*n*]), который принимает значение суммы элементов таблицы;
- **Алг** вещ ***SUMMOD*** (цел *n*, вещтаб *X*[1:*n*]), который принимает значение суммы модулей элементов таблицы;
- **Алг** ***SIGN*** (цел *n*, вещтаб *X*[1:*n*]), который меняет знаки всех элементов таблицы *X* на противоположные.

Как, используя данную библиотеку, написать алгоритм, который:

А) определяет, что в таблице *X* нет положительных элементов;

Б) находит минимум в таблице *X*;

В) определяет, верно ли, что в строке *X* ровно один элемент отличен от нуля.

4. Альфа (3 балла)

Что делает следующий алгоритм?

```
Алг АЛЬФА(нат n,x,y)
арг n
рез x,y
нач нат z
   если n<3
      то x:=1
         y:=1
      иначе
       АЛЬФА(n-1,y,z)
         x:=y+z
  все
кон
```

5. Сейф (10 баллов)

Кодовый замок сейфа представляет собой n независимых рычагов, каждый из которых может находиться в одном из k положений. Таким образом, положение замка может быть описано n-значным числом, каждая цифра которого может принимать значения от 1 до k. За одну секунду можно перевести только один из рычагов в соседнее положение, что соответствует изменению одной из цифр на одну единицу.

Грабители застают замок запертым в некотором положении. Определите, могут ли они перебрать все возможные положения замка, не теряя ни секунды (т. е. каждую секунду получая новую комбинацию), и как это сделать, если:

А) $n = 3$, $k = 3$;
Б) $n = 4$, $k = 4$.

6. Алгоритм Маркова (9 баллов)

Алгоритм преобразования входной строки S описывается конечным упорядоченным набором пар строк, называемых «подстановками». Первая строка в подстановке называется «левой», а вторая — «правой». В S ищется подстрока, совпадающая с одной из левых строк подстановок, и заменяется соответствующей правой строкой. Поиск происходит с начала строки S для первой подстановки, затем для второй, третьей и т. д. После первого выполнения подстановки работа начинается сначала уже с преобразованной строкой. Алгоритм завершается, когда не удается выполнить ни одной подстановки.

Например, алгоритм, описываемый парой подстановок («пи», «ма») и («м», «п»), преобразует входную строку «мама» в строку «папа», а входную строку «мир» — в «пар».

Напишите последовательность подстановок, преобразующую символьную строку, представляющую собой целое положительное десятичное число, в строку, представляющую собой число, большее исходного на 1. В исходной строке число оканчивается символом «*».

Практический тур

7. Арифметика по-русски (20 баллов)

Напишите программу, преобразующую словесное описание арифметического действия в словесное описание его результата.

Входные данные:

- строка символов — арифметический пример. Во входной строке для обозначения арифметического действия используется одно из следующих слов: «плюс», «прибавить»,

«минус», «вычесть», «умножить на». Операнды — целые, положительные, меньшие 100. Использование предлогов, согласование падежей и порядок слов должны соответствовать правилам русского языка.

Данные вводятся из файла `arithmetic.in` или с клавиатуры.

Выходные данные:

- строка символов — результат вычисления.

Результат выводится в файл `arithmetic.out` или на экран.

Пример:

arithmetic.in	arithmetic.out
сорок восемь умножить на три	сто сорок четыре

8. Колесо (17 баллов)

Напишите программу, изображающую на экране катящееся по полу между двумя стенками колесо. Колесо должно иметь равномерно расположенные по кругу спицы, соединяющие центр с ободом. Вертикальные стенки должны располагаться около границ экрана. Диаметр колеса — примерно 4 см. После соприкосновения со стенкой колесо должно менять направление движения на противоположное. Предусмотрите остановку программы по требованию.

А) Количество спиц равно 3.
Б) Количество спиц задается.

Качество рисунка может быть оценено дополнительно (до 3 баллов).

Областная олимпиада

Теоретический тур

9. «ЧЕГОНАДО» (6 баллов)

Всю неделю бутылка ЧЕГОНАДО стоила k рублей, а пустая бутылка стоила l рублей. Компания друзей, собравшихся в понедельник, располагала первоначальным капиталом в n рублей и купила на все эти деньги ЧЕГОНАДО. Употребив все, они на следующий день сдали пустые бутылки, добавили сдачу с предыдущего дня и снова на все деньги купили ЧЕГОНАДО. Данная процедура продолжалась каждый день, пока была возможность.

Предложите алгоритм нахождения минимальной суммы n, при которой в пятницу друзьям будет что употребить. Алгоритм обоснуйте.

10. Список (10 баллов)

Имеется список элементов, каждый из которых содержит адрес следующего элемента, что позволяет последовательно просматривать элементы. Длина списка неизвестна. Последний элемент должен содержать нулевой адрес. Однако список может быть зациклен, т. е. в нем может оказаться элемент, содержащий адрес одного из предшествующих элементов. Предложите алгоритм, определяющий, есть ли в списке цикл.

11. УХТЫ-01 (8 баллов)

Вычислительная машина УХТЫ-01 может выполнять только ввод и вывод данных и одну операцию $x = f(a,b)$, по которой переменной x присваивается значение $1 - \frac{a}{b}$.

Напишите программу для машины УХТЫ-01, которая вводит два исходных числа, отличных от нуля, и выводит их сумму, разность, произведение и частное. В программе можно использовать рабочие переменные, обозначенные латинскими буквами.

12. Лабиринт (10 баллов)

Попав в лабиринт, состоящий из одинаковых квадратных комнат, каждая из которых может иметь от 1 до 4 дверей в соседние комнаты, путник долго блуждал по нему, пока не нашел клад. Во время поиска он составил описание своего маршрута, обозначая каждый переход из комнаты в комнату буквами: С (север), В (восток), Ю (юг), З (запад).

Опишите алгоритм, определяющий по заданной записи возможно более короткий путь назад.

13. Дисперсия (6 баллов)

Средним значением конечного набора чисел $\{a_i\}$ называется число m, равное среднему арифметическому чисел a_i. Среднее арифметическое чисел $(a_i - m)^2$ назовем дисперсией.

Опишите алгоритм, который читает элементы некоторой числовой последовательности и после чтения каждого элемента печатает среднее значение (2 балла) и дисперсию (4 балла) всех прочитанных элементов.

14. Алгоритм (4 балла)

Какое значение примет переменная n после выполнения алгоритма (2 балла):

```
Алг ДУМА(цел n,k)
арг k
рез n
нач цел i
    n:=0
    i:=0
    нц пока i<k
        n:=n+2*i+1
        i:=i+1
    кц
кон
```

Ответ обоснуйте (2 балла).

15. Сапер (7 баллов)

На прямоугольном поле размером $2 \times N$ в нижней строке случайным образом расставлено некоторое количество мин, не видимых игроку (саперу), а в верхней строке в каждой клетке написаны числа от 0 до 3, которые совпадают с количеством заминированных клеток нижней строки, соседних с этой клеткой. Игрок должен определить расположение мин.

А) Предложите алгоритм, который определяет все возможные положения мин (3 балла).

Б) Какое возможно максимальное количество решений (1 балл)?

В) Каким условиям должны удовлетворять начальные данные, чтобы решение было неоднозначным (3 балла)?

Практический тур

16. Уравнение по-польски (20 баллов)

Задано арифметическое выражение $F(x)$, состоящее из целых чисел, арифметических операций: `+`, `-`, `*`, `/` и переменной x, которая входит в выражение не более одного раза. Выражение задается в обратной польской записи, где знак операции ставится не между операндами, а после них. При этом скобки вообще не используются, а операции в выражении выполняются слева направо.

Пример:

Обычная запись	Польская запись
`2+10`	`2 10 +`
`3*(11-x)`	`3 11 x - *`
`3-11*x`	`3 11 x * -`
`((x+2)/555)*(9-99)`	`x 2 + 555 / 9 99 - *`

Разделителем операндов и знаков операций в исходной строке служит пробел. Напишите программу, которая решает уравнение $F(x) = 0$ и печатает все его корни.

Входные данные:

- строка символов — выражение в обратной польской записи.

Данные вводятся из файла `equation.in` или с клавиатуры.

Выходные данные:

- одно число с точностью до двух знаков после запятой — решение уравнения или сообщение «Нет решений».

Примеры:

equation.in	equation.out
3 11 x - *	11.00
3 11 x - /	Нет решений
3 11 x * -	0.27
1 3 11 x - / -	8.00

17. Шахматы (18 баллов)

Задано расположение двух белых ладей и черного короля в шахматной нотации (всего 3 фигуры). Напишите программу, играющую за белых и ставящую мат черному королю.

Входные данные:

- xn, ym, zk — координаты ладей и короля в шахматной нотации, разделенные пробелом: x, y, z — латинские буквы от a до h, обозначающие вертикаль; n, m, k — числа от 1 до 8, обозначающие горизонталь.

Данные вводятся из файла chess.in или с клавиатуры.

Выходные данные:

- единственная строка должна содержать один ход в следующем формате: xn — ym, где xn координаты фигуры до хода, ym — координаты фигуры после хода. Имеется в виду первый ход, гарантирующий выигрыш за минимальное количество ходов.

Результат выводится в файл chess.out или на экран.

Пример:

chess.in	chess.out
a7 b5 h8	b5-b8

Городская олимпиада

Теоретический тур

1. Хорошая таблица (9 баллов)

Пусть имеется целочисленная таблица размером $N \times M$. «Путем» в этой таблице назовем последовательность из $N + M - 1$ клеток, которая начинается из клетки (1,1), заканчивается клеткой (N,M) и любые две идущие подряд клетки в этой последовательности имеют общую сторону. Назовем таблицу «хорошей», если сумма чисел в клетках каждого пути в этой таблице одна и та же. Составьте и обоснуйте алгоритм, который определяет, является ли данная таблица «хорошей».

2. Краевая перфорация (10 баллов)

Для обработки статистической информации использовались перфокарты с двумя горизонтальными рядами отверстий (рис. 10).

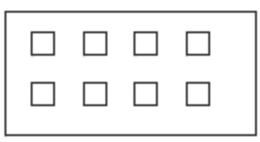


Рис. 10

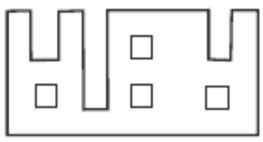


Рис. 11

Рис. 12

Рис. 13

Для кодировки информации на этих перфокартах от одного края пробивались вертикальные прорези, захватывающие либо одно, либо два отверстия (рис. 11). Для отбора карт с нужным признаком стопка перфокарт одевалась на горизонтальные штыри, входящие в отверстия, и отпускалась. При этом часть карт оставалась на штырях, а часть падала вниз.

Пример.

Если перфокарту (см. рис. 11) одеть на штыри (рис. 12), то она провалится, а если на штыри (рис. 13), то останется висеть.

Разработайте способ кодировки десятичных цифр, позволяющих отбирать карты, соответствующие каждой цифре, путем одноразового надевания на какой-то набор штырей (для каждой цифры — свой набор). Кодировка должна использовать возможно меньшее число вертикальных рядов отверстий.

Тот же вопрос — для кодировки 26 букв латинского алфавита.

3. Ковер и моль (8 баллов)

В прямоугольном ковре моль проела несколько маленьких дырок. Опишите алгоритм, который по известным размерам ковра, количеству в нем дырок и их координатам определяет размеры и местоположение прямоугольника максимальной площади, который не содержит дырок и стороны которого параллельны сторонам исходного ковра.

4. Кольцевая дорога (6 баллов)

На кольцевой дороге произвольным образом расположены N бензоколонок с общим количеством бензина, достаточным для того, чтобы проехать на машине один круг.

Докажите, что при любом распределении бензина по бензоколонкам найдется такая, начиная с которой это удастся сделать. Приведите алгоритм нахождения такой бензоколонки.

5. Шайка (6 баллов)

Каждый член шайки, включая главаря, имеет не более двух подручных, знает только их телефонные номера и имеет собственный телефон. Ни один член шайки не может быть подручным сразу у двоих. Сформулируйте алгоритм, позволяющий главарю узнать численность шайки в двух случаях:

А) никто из членов шайки, кроме главаря, не умеет считать;

Б) все члены шайки, кроме главаря, умеют считать.

6. Робот (3 балла)

Исполнитель Робот живет на прямоугольном клетчатом поле и может передвигаться по клеткам, если на них нет препятствий, делая шаги в одном из четырех направлений: север, юг, восток, запад. Препятствия он умеет распознавать с помощью локатора. Кроме того, Робот может закрасить клетку, на которой стоит.

На поле Робота имеется тупик — ряд свободных клеток, идущий с запада на восток, высотой в одну клетку и длиной в несколько клеток. Опишите, что сделает Робот в результате выполнения алгоритма СЕКРЕТ, если к началу выполнения он стоит в первой клетке тупика.

```
Алг СЕКРЕТ
нач
   ХОД
кон

Алг ХОД
нач
  закрасить клетку
  шаг на восток
  если на востоке свободно
    то ХОД
  все
кон
```

7. Кинотеатр (5 баллов)

Опишите информационную модель для подсчета дневной выручки кинотеатра от проката фильма и приведите алгоритм обработки данных для реализации этой задачи.

Примечание: под информационной моделью понимается структурированный набор величин, содержащий всю необходимую информацию об исследуемых объектах или процессах.

Практический тур

8. Как из мухи сделать слона (20 баллов)

Имеется словарь четырехбуквенных слов, записанный в файле.

«Допустимым преобразованием» слова из этого словаря, или «ходом» назовем замену любой одной его буквы на любую другую букву, после которой снова получается слово, имеющееся в данном словаре. Например, «слог» — «стог» или «улов» — «плов».

Напишите программу, которая по данному словарю, исходному слову и конечному слову:

А) определяет, можно ли получить за некоторое количество ходов из начального слова конечное;
Б) приводит пример какой-нибудь искомой цепочки слов (если данное преобразование возможно);
В) находит и печатает искомую цепочку слов минимальной длины.

Входные данные:

- в файле dictionary.txt помещены четырехбуквенные русские слова по одному в строке.
- в двух строках файла fly.in даны начальное и конечное слова.

Выходные данные:

- результат выводится в файл fly.out по одному слову в строке.

Пример:

fly.in	fly.out
море порт	море горе гора пора порт

9. Перестановка книг (22 балла)

В библиотеке на полке стоят N книг ($N < 1000$). На корешке каждой книги написан ее номер от 1 до N.

Библиотекарь отсчитывает i-ю слева книгу и берет с полки k стоящих подряд книг, начиная с i-й. После этого он отсчитывает j-ю книгу из оставшихся и все снятые с полки книги, не меняя их порядка, ставит после j-й. Если $j = 0$, то вставка происходит перед первой книгой. Тройку чисел i, k, j он записывает в тетрадь. Такую операцию он проделывает M раз ($M < 1000$).

А) Напишите программу, которая по записям в тетради находит распределение книг на полке, образовавшееся после всех перестановок.

Б) Напишите программу, которая по исходному расположению книг выводит тройки, определяющие перестановки книг, обеспечивающие сортировку книг по возрастанию номеров. Количество троек M должно быть по возможности минимально.

Входные данные для пункта А:

- N — количество книг ($0 < N < 1000$);
- n_1, n_2, ..., n_N — номера книг в начальной расстановке книг на полке, разделенные пробелом;
- M — количество произведенных операций;
- в следующих M строках через пробел указаны тройки чисел i, k, j, записанные в тетради.

Данные вводятся из файла `biblioA.in`.

Выходные данные для пункта А:

- номера книг в конечной расстановке книг на полке, разделенные пробелом.

Результат выводится в файл `biblioA.out` или на экран по одному слову в строке.

Пример:

biblioA.in	biblioA.out
5 1 2 3 4 5 2 2 3 2 4 2 0	3 4 1 5 2

Входные данные для пункта Б:

- N — количество книг ($0 < N < 1000$);
- n_1, n_2, ..., n_N — номера книг в начальной расстановке книг на полке, разделенные пробелом.

Данные вводятся из файла `biblioB.in`.

Выходные данные для пункта Б:

- M — количество произведенных операций;
- в следующих M строках через пробел указаны тройки чисел i, k, j, определяющие необходимые для сортировки операции.

Результат выводится в файл `biblioB.out`.

Пример:

biblioB.in	biblioB.out
5 3 4 1 5 2	2 1 2 1 2 3 2

Областная олимпиада

Теоретический тур

10. Автобус (10 баллов)

На междугородном автобусном маршруте n остановок, включая конечные. Пусть $p(i, j)$ — количество билетов, проданных пассажирам, едущим от i-й до j-й остановки во время одного из рейсов.

Опишите алгоритм, который по заданным значениям $p(i, j)$ определяет максимальное число пассажиров, одновременно ехавших в автобусе.

11. Сеть (8 баллов)

16 компьютеров связаны в сеть в виде решетки (рис. 14).

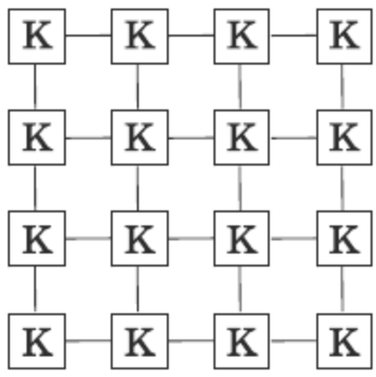


Рис. 14

В памяти каждого имеется своя информация. За один такт компьютер может передать или принять от одного из соседей информацию любого объема.

За сколько тактов и каким образом все компьютеры могут получить всю информацию?

12. Прожекторы (8 баллов)

В некоторых точках плоскости, заданных координатами, расположено N включенных прожекторов. Каждый прожектор освещает прямой угол. Стороны углов сонаправлены положительным направлениям осей координат.

Опишите алгоритм, выключающий максимальное число прожекторов так, чтобы освещенная часть плоскости не уменьшилась.

13. Ранг числа (10 баллов)

Пусть $S(n)$ обозначает сумму цифр целого числа n. Ранг $R(n)$ неотрицательного числа n определим согласно следующим правилам:

$$R(0) = 0;$$
$$R(n) = R(n - S(n)) + 1,\ n > 0.$$

Опишите алгоритм, определяющий минимальное натуральное число, имеющее заданный ранг R.

14. Сломанный калькулятор (4 балла)

У калькулятора сломалась клавиша умножения. При нажатии на нее набранное число возводится в квадрат.

А) Как на таком калькуляторе перемножить два числа?
Б) Как это сделать, если к тому же при нажатии на клавишу деления набранное число сразу делится на два?

15. НОК (4 балла)

Заданы натуральные числа $a_1, a_2, \ldots, a_n$. Опишите алгоритм нахождения их наименьшего общего кратного (НОК).

Практический тур

16. Расшифровка текста (25 баллов)

Некоторый русский текст, хранящийся в файле, зашифрован следующим образом: каждая буква текста циклически сдвигается на одно и то же количество символов, а все остальные символы (знаки препинания и пробелы) остаются неизменными. При шифровке большие буквы переходят в большие, а маленькие в маленькие. Будем для простоты считать, что русский алфавит состоит из 32 букв (за буквой «е» следует буква «ж»).

Требуется написать программу расшифровки текста.

Входные данные:

- зашифрованный текст содержится в файле `text.in`. В некоторых коротких тестах возможен ввод с клавиатуры.

Выходные данные:

- расшифрованный текст должен находиться в файле `text.out`. Возможен вывод на экран. В случае невозможности расшифровки должен быть выведен нуль.

Пример:

text.in	text.out
А сбвпубя, оп а фтубм	Я работаю, но я устал

17. Фигурки из правильных треугольников (20 баллов)

Имеется N одинаковых правильных треугольников. Их последовательно выкладывают на плоскости так, чтобы каждый следующий треугольник имел общую сторону хотя бы с одним из уже выложенных. Таким образом можно получить разные фигуры. Две фигуры считаются одинаковыми, если их можно совместить с помощью параллельных переносов, поворотов и отражений.

Напишите программу, которая по введенному числу N выдает количество разных фигур, которые можно составить, используя все N треугольников, и рисует все эти фигуры на экране.

Входные данные:

- число N вводится с клавиатуры или из файла `gexamino.in`; $0 < N < 20$.

Выходные данные:

- в файл `gexamino.out` или на экран требуется вывести одно число — количество различных фигурок. Изображения фигурок можно вывести на экран в произвольном формате.

Пример:

gexamino.in	gexamino.out
4	3

Далее на экран должно быть выведено изображение трех искомых фигурок (рис. 15).

Рис. 15

ОЛИМПИАДЫ 1996/97 УЧЕБНОГО ГОДА

Городская олимпиада

Теоретический тур

1. Машины (7 баллов)

Из Ярославля в Рыбинск выехало n машин в моменты времени $a_1, a_2, \ldots, a_n$ со скоростями $v_1, v_2, \ldots, v_n$, соответственно. Такое же количество машин выехало из Рыбинска в Ярославль в моменты времени $b_1, b_2, \ldots, b_n$ со скоростями $w_1, w_2, \ldots, w_n$.

Опишите алгоритм, который для любого момента времени t находит участок дороги длиной 1 километр, на котором количество машин, едущих в Рыбинск и в Ярославль, совпадают, или определяет, что такого участка нет.

2. Смежные перестановки (9 баллов)

Перестановкой чисел от 1 до n называется произвольное расположение этих чисел одно за другим. Две перестановки считаются смежными, если они отличаются одной транспозицией, т. е. перестановкой двух рядом стоящих чисел.

Опишите алгоритм, располагающий все перестановки по кругу так, чтобы рядом стояли только смежные.

3. Генеалогическое древо (8 баллов)

Остап Бендер решил восстановить историю своего славного рода турецких подданных. Для этого он внес в список себя, своих родителей, родителей родителей (если их имена были ему известны), и так далее до тех пор, пока окончательно не исчерпал всю информацию. Оказалось, что ни одно имя не встретилось в этом списке дважды.

Опишите алгоритм, который нумерует всех попавших в этот список таким образом, чтобы номер любого был:

А) меньше номера своего отца (если он есть в списке), но больше номера своей матери (если она есть в списке);

Б) меньше номера любого своего предка по отцовской линии, но больше номера любого своего предка по материнской линии.

4. Гири (10 баллов)

Даны гири различных весов a_1, a_2, ..., a_n. Опишите алгоритм определения первых k значений веса (в порядке возрастания), которые можно отвесить с помощью этих гирь. Гири можно класть только на одну чашу весов.

5. Рациональный умножитель (6 баллов)

Дано натуральное число n. Разрешается умножать его на рациональное число $\frac{p}{q} < 1$ (p, q — простые числа или единица), такое, что произведение опять натуральное. С получившимся в результате этого действия числом можно проделать такую же операцию (возможно, с другой дробью).

Опишите алгоритм, который по заданному n строит одну из наиболее длинных последовательностей таких действий.

6. Алгоритм (4 балла)

Что делает алгоритм?

```
Алг УПР(целтаб a[1..10000])
  рез a
нач цел i,j
  нц для i от 1 до 10000
     a[i]:=0
  кц
нц для i от 2 до 100
  нц для j от i+i до 10000 шаг i
     a[i]:=1
  кц
кц
кон
```

7. Три круга (4 балла)

Даны три круга с вершинами в точках $A(x_1, y_1)$, $B(x_2, y_2)$ и $C(x_3, y_3)$ радиусов R_1, R_2 и R_3, соответственно. Опишите алгоритм, определяющий, накрывают ли данные круги треугольник ABC.

Практический тур

8. Поиск (24 баллов)

Из заданного словаря русских слов требуется выбрать все слова, соответствующие шаблону.

Шаблон — это произвольная последовательность строчных букв, вопросительных знаков «?» и звездочек «*». Слово соответствует шаблону, если оно может быть получено из него путем замены каждого знака «?» одной буквой, а знака «*» — некоторой последовательностью букв (возможно пустой).

Входные данные:

- в файле `dictionary.txt` находится словарь русских слов, написанных строчными буквами, разделенных пробелами и/или символами конца строки;
- шаблон — строка, содержащая не более 256 символов, находится в файле `search.in`.

Некоторые короткие тесты могут быть введены с клавиатуры.

Выходные данные:

- найденные слова, через пробел, в том же порядке, в каком они шли в словаре, должны быть выведены в файл `search.out` или на экран.

Пример:

search.in	search.out
?и?ер*а	гипербола

Возможно частичное решение задачи, когда шаблон не содержит символов «*» (9 баллов).

9. Табло (30 баллов)

Имеется табло размером $m \times n$, состоящее из лампочек-кнопок. Лампочка включается и выключается при нажатии на нее, при этом одновременно меняется состояние всех соседних с ней лампочек (горящие выключаются, а не горящие включаются). Соседними считаются лампочки, расположенные

рядом по горизонтали, вертикали или диагонали. У лампочки может быть до восьми соседей.

А) Напишите программу, которая из некоторого заданного начального состояния табло по заданной последовательности нажатий на кнопки выводит конечное состояние (7 баллов).

Б) Напишите программу, которая для заданного начального состояния лампочек находит последовательность переключения кнопок, обеспечивающую полное выключение табло, или выдает на экран сообщение, что этого сделать невозможно (23 балла).

Входные данные для пункта А:

- в первой строке файла `tabloA.in` находятся размеры табло m, n — два натуральных числа, разделенные пробелом, не превышающие 256;
- в последующих m строках находятся последовательности из нулей и единиц длины n, соответствующие начальному состоянию табло. Нуль соответствует выключенной лампочке, а единица — включенной;
- в следующей строке находится одно число k — количество нажатий на кнопки ($0 \leqslant k \leqslant 1000$);
- в последующих k строках через пробел записаны два натуральных числа s, r — номер строки и столбца, указывающие положение кнопки.

Выходные данные для пункта А:

- в m строках файла `tabloA.out` находятся последовательности из нулей и единиц длины n, соответствующие конечному состоянию табло.

Пример:

tabloA.in	tabloA.out
3 4 1000 0000 0000 2 2 2 3 4	0110 1101 1101

Входные данные для пункта Б:

- в первой строке файла `tabloB.in` находятся размеры табло m, n — два натуральных числа, разделенные пробелом, не превышающие 256;
- в последующих m строках находятся последовательности из нулей и единиц длины n, соответствующие состоянию табло.

Выходные данные для пункта Б:

- в первой строке файла `tabloB.out` находится одно число k — количество нажатий на кнопки;
- в последующих k строках через пробел записаны по два натуральных числа s, r — номер строки и столбца, указывающие положение кнопки. В случае отсутствия решения — вывести одно число −1.

Пример:

tabloB.in	tabloB.out
3 4 1110 0010 1100	2 1 2 3 1

Областная олимпиада

10. МПС (60 баллов)

От Москвы до Владивостока n железнодорожных станций: a_0 (*Москва*), a_1, a_2, ... , a_n (*Владивосток*). Стоимость проезда от станции a_{i-1} до станции a_i равнялась s_i рублей. Между любыми двумя станциями цена билета складывалась из стоимостей проезда на отдельных участках. В целях ускорения обслуживания пассажиров Министерство путей сообщения рассчитало цены билетов между каждыми парами станций и округлило полученный результат до целого числа тысяч рублей (500 рублей округлялось в большую сторону). В результате некоторые билеты стали стоить дороже, а некоторые — дешевле.

А) Напишите программу, которая по заданным s_i рассчитывает, до каких станций следует покупать билеты по пути следования из Москвы во Владивосток, чтобы стоимость проезда была минимальной (30 баллов).

Б) Дайте краткое описание и обоснование алгоритма (30 баллов).

Входные данные:

- в первой строке файла `mps.in` указано число n ($0 < n < 5000$);
- в последующих n строках указана s_i — стоимость проезда от станции a_{i-1} до станции a_i — натуральное число, не превышающее 10 000.

Выходные данные:

- в первой строке файла `mps.out` должно содержаться одно число — общая стоимость проезда;
- во второй строке должно быть указано k — число станций, до которых следует покупать очередной билет;
- в последующих k строках должны быть указаны номера этих станций.

Пример:

mps.in	mps.out
3 1800 1600 1400	4000 2 2 3

11. База данных (40 баллов)

В текстовом файле `STATES` содержатся сведения о штатах США на английском языке. В каждой строке содержатся сведения об одном штате. При этом через запятую перечисляются название штата, столица, площадь и население. Площадь и население — целые числа в диапазоне от 0 до 30 000.

Напишите программу, которая по заданному фильтру создает файл, содержащий строки исходного файла, удовлетворяющие условиям фильтра.

Фильтр представляет собой логическое выражение, содержащее одно простое условие или два простых условия, связанных операциями OR или AND. Простое условие состоит из имени поля (State, Capital, Square, Population), одной из операций сравнения (для числовых полей: >, <, =; для текстовых: =) и значения (для числовых полей — число, для текстовых — текст или текст со звездочкой). Простые условия отделяются от слов OR и AND пробелом.

Примеры фильтров:

```
State=Texas
Capital=Ai* OR Population<10000
Square>300 AND Population=20000
```

Примечание: выражение Capital=Ai* означает, что требуется выбрать информацию обо всех штатах, столицы которых начинаются на «Ai».

Городская олимпиада

Теоретический тур

1. Восстановление таблицы (10 баллов)

По двумерной числовой таблице A размером $m \times n$ (m, $n > 2$) построили новую таблицу B того же размера, в которой каждый элемент $B(i, j)$ равен среднему арифметическому элементов i-й строки и j-го столбца таблицы A без учета элемента $A(i,j)$, например:

A:

4	0	6
4	4	0
6	2	4

B:

4	4	2
3.5	1.5	4.5
3.5	3.5	3.5

Опишите алгоритм восстановления исходной таблицы A по заданной таблице B.

2. Программа (4 балла)

Дано натуральное число k. Составьте программу нахождения всех пар натуральных чисел m и n, таких что $m^2 + k = n^2$.

3. Роботы-парашютисты (7 баллов)

На целочисленную числовую прямую одновременно приземлились два одинаковых робота-парашютиста. Каждый из них может выполнять команды: `ШАГ ВЛЕВО`, `ШАГ ВПРАВО`, `ПОЛОЖИТЬ ПАРАШЮТ`, проверять условие `НАЙДЕН ПАРАШЮТ`, `НАЙДЕН РОБОТ`, а также понимает конструкции ветвления и цикла `ПОКА`, в том числе и с составными условиями. При этом на каждое из 5 действий у робота уходит 1 такт. Какую программу должны были написать хозяева роботов для каждого из них, чтобы независимо от начального положения роботы встретились?

4. Удаление элементов массива (10 баллов)

По заданному линейному числовому массиву A определите наименьшее количество элементов, при удалении которых останется возрастающая последовательность.

5. Шифровка (9 баллов)

Вася придумал способ шифровки текстов, допускающий разные уровни сложности, и зашифровал фразу. У него получилось:

Первый уровень —

РР-НРЖОФ, ЯУР ОЗ УВЛВА ХЗ ССРТФББ ИВЕВШВ.

Второй уровень —

СС-ПТЗПЦ, БФС РЙ ФГНДБ ЦЙ УТСФЦВВ КДЖГЪД.

Третий уровень —

ТТ-РФКСЧ, ВХУ ТЛ ХДОЖГ ШК ФУУЦШГГ ЛЖЗЕЫЕ.

Расшифруйте фразу и зашифруйте ее на четвертом уровне сложности.

6. Алгоритм (3 балла)

Что делает данный алгоритм?

```
алг вещ ALFA(цел N,I,J, вещтаб A[1:N])
нач цел k
  если I=J
    то     знач:=A[I]
    иначе k:=цел_часть((I+J)/2)
           знач:=ALFA(N,I,k,A)+ALFA(N,k+1,J,A)
  все
кон
```

Практический тур

7. «Приятное» число (20 баллов)

n-значное число назовем «приятным», если для любого $k \leqslant n$ первые его k цифр образуют число, делящееся на k. Найдите как можно большее «приятное» число.

Пример четырехзначного «приятного» числа: 3216 (3 делится на 1; 32 делится на 2; 321 делится на 3; 3216 делится на 4).

Результат представить на экране или поместить в файл `number.out`.

8. Множества (30 баллов)

Рассмотрим множество P точек экрана с координатами $10 \leqslant x \leqslant 600$ и $10 \leqslant y \leqslant 400$, образующее прямоугольник.

Будем задавать круги, расположенные в этом прямоугольнике, координатами центров и радиусами. Кругов может быть задано не более четырех. Каждый круг обозначается соответственно A, B, C и D.

Назовем объединением кругов $A + B$ множество точек, принадлежащих хотя бы одному из кругов, пересечением $A * B$ — множество точек, принадлежащих обоим кругам, и разностью $A - B$ — множество точек, принадлежащих A и не принадлежащих B.

Напишите программу, которая по заданным кругам и по введенному выражению, связывающему множества с помощью определенных выше операций, рисует на экране контуры множеств P, A, B, ... и закрашивает множество точек, определенное данным выражением.

Входные данные:

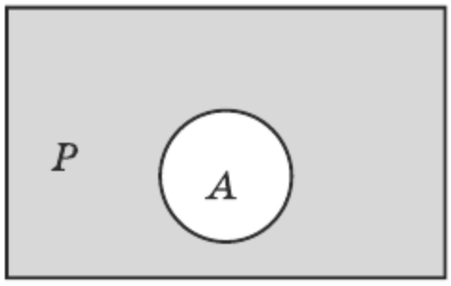
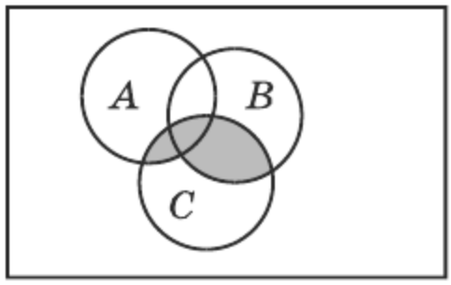
- в первой строке входного файла `Euler.in` находится количество кругов — натуральное число n, не превышающее четырех;
- в следующих n строках находятся по три числа, разделенных пробелом: r, x, y — радиусы и координаты центров кругов A, B, C и D, соответственно;
- в последней строке помещено логическое выражение (возможно, со скобками). Длина строки не превышает 20.

Данные могут быть введены с клавиатуры.

Выходные данные.

Результат выводится на экран.

Примеры:

Euler.in	**Изображение на экране**
1 100 300 100 P-A	
3 50 200 300 50 300 200 45 250 150 (A+B)*C	

Областная олимпиада

Практический тур

9. Прогрессия из простых чисел (40 баллов)

Найдите две возможно более длинные арифметические прогрессии, состоящие из простых чисел, не превосходящих 32 768. Разности найденных прогрессий должны быть различны.

Выходной файл должен быть создан до окончания работы (дополнительного времени для его создания предоставляться не будет). Файл должен содержать две строки — в каждой одна прогрессия. Элементы прогрессии отделяются пробелами.

Пример выходного файла `prime.out`:

```
3 5 7
3 7 11
```

Это — не самые длинные прогрессии!

10. Пробирки (60 баллов)

Имеются три пробирки. Вместимость каждой из них — 100 миллилитров. Две пробирки из трех одинаково размечены. Деления нанесены произвольно и соответствуют целым количествам миллилитров. Изначально одна из пробирок с делениями наполнена 100 миллилитрами кваса, а остальные — пустые.

А) Напишите программу, которая выясняет, можно ли поместить в пробирку без делений один миллилитр кваса, и если да, то находит минимальное число необходимых для этого переливаний. Каждое переливание из одной пробирки в другую можно проводить до тех пор, пока либо первая из них не станет пустой, либо одна из пробирок не окажется заполненной до какого либо деления (40 баллов).

Б) Опишите алгоритм, реализованный в программе, и дайте его теоретическое обоснование (20 баллов).

Входные данные:

- в первой строке входного файла `tube.in` находится число делений на каждой из первых двух пробирок: N — натуральное число, не превышающее 20;
- в следующих N строках в порядке возрастания через пробел или символ конца строки следуют натуральные числа, приписанные делениям: $V_1, \ldots, V_N$ ($1 \leqslant V_i \leqslant 100$). Последнее деление считается сделанным на верхнем крае пробирки ($V_N = 100$).

Выходные данные:

- в первой строке файла `tube.out` должно содержаться слово «`YES`», если добиться результата можно, и «`NO`» — в противном случае;
- в первом случае во второй строке необходимо указать искомое количество переливаний.

Пример:

tube.in	tube.out
4 13 37 71 100	YES 8

Городская олимпиада

Теоретический тур

1. Доска (4 балла)

На доске выписаны 19 единиц и 99 «минус единиц». Что останется на доске после применения следующего алгоритма:

```
нц  пока на доске больше одного числа
  выбрать два произвольных числа
  вычислить их произведение
  записать на доску результат
  стереть с доски выбранные числа
кц
```

Ответ обоснуйте.

2. Алгоритм (4 балла)

Что делает следующая процедура?

```
procedure A (n: integer);
begin
 if n>0 then
   begin
     A(n div 2);
     write(n mod 2)
   end;
end;
```

Примечание: `k div m` — частное от деления; `k mod m` — остаток от деления k на m.

3. Матричный порядок (8 баллов)

Две таблицы размером $n \times m$ (n строк, m столбцов) сравниваются следующим образом: сначала первые элементы первых строк, затем вторые элементы первых строк и т. д. После

сравнения всех элементов первых строк переходят ко вторым строкам таблиц, затем к третьим и т. д. Та таблица считается большей, у которой при таких сравнениях раньше появится больший элемент.

Имеется таблица, все элементы которой различны. Из нее получают все возможные другие таблицы путем перестановок строк и столбцов. Затем все полученные таблицы сортируют по возрастанию.

Опишите алгоритм, позволяющий определить элемент, стоящий на последнем месте последней строки в самой последней таблице.

4. Зеркало (6 баллов)

Имеется зеркальное стекло в виде выпуклого многоугольника. Опишите алгоритм, с помощью которого можно вырезать круглое зеркало так, чтобы общая площадь отходов была минимальна.

5. Теннисный турнир (3 балла)

В теннисном турнире «на вылет» для победы в партии необходимо выиграть три сета. Напишите программу, которая по заданным целым положительным числам m и k определяет, мог ли какой-то из участников выиграть в турнире ровно m сетов, а проиграть — ровно k сетов.

6. Максимальное произведение (10 баллов)

Опишите алгоритм нахождения таких различных натуральных чисел, сумма которых равна заданному числу n, а произведение максимально.

7. Максимальная сумма (10 баллов)

Дана последовательность чисел a_1, a_2, ..., a_n. Опишите эффективный алгоритм нахождения в этой последовательности нескольких идущих подряд элементов, имеющих наибольшую возможную сумму.

Практический тур

8. «Пестрое» число (30 баллов)

Натуральное число, все цифры которого различны, назовем «пестрым». Напишите программу нахождения возможно большего «пестрого» числа, кратного заданному натуральному числу n.

Входные данные:

- во входном файле `pied.in` находится одно натуральное число n, не превышающее 32 000. Разрешается ввод с клавиатуры.

Выходные данные:

- результат должен быть выведен в файл `pied.out` или на экран.

Пример:

pied.in	pied.out
9	9876543210

9. Электронное письмо (20 баллов)

Вами получено электронное письмо от друга, проживающего в Урюпинске. К сожалению, текст письма не читается, так как во время длительного пути в Интернете сбилась кодировка русских букв. Но не все потеряно!

У вас имеется информация о возможных кодировках символов. Остается только написать программу, которая расшифрует письмо. Выполнить задание надо в двух вариантах.

А) Вместо символов в тексте письма — последовательность кодов, соответствующих символам (15 баллов).

Б) Текст письма — результат «неудачного» раскодирования письма (письмо было закодировано в одной кодировке, а раскодировано с помощью другой) (10 баллов).

Входные данные:

- в файле `code.txt` содержится информация о возможных кодировках символов в следующем формате. В каждой строке первым идет кодируемый символ (пробел, русская прописная или строчная буква), а далее через пробел — коды (числа от 0 до 255), соответствующие разным кодировкам данного символа. Количество кодируемых символов определяется числом строк в файле (не более 67), а количество кодировок определяется числом кодов в одной строке (не более 100). Каждый описанный символ имеет коды во всех кодировках.

Входные данные для задания А:

- во входном файле `messageA.in` содержится текст письма — последовательность кодов (не более 1000), разделенных пробелами и/или символами конца строки.

Входные данные для задания Б:

- во входном файле `messageB.in` содержится текст письма — последовательность символов (не более 1000), возможно разделенных символами конца строки.

Выходные данные:

- все возможные тексты исходного письма должны быть выведены в файл `message.out`. В случае невозможности раскодирования надо вывести число 0.

Пример для задания А:

messageA.in	messageA.out
118 113 106 102 104 115	Привет емуСПк

Пример для задания Б:

messageB.in	messageB.out
е м у С П к Р у ю а м з н у п л т У	Привет из Урюпинска

Пример файла `code.txt`:

```
а 100 122
б 101 121
в 102 120
  103 119
е 104 118
З 105 117
и 106 116
к 107 115
л 108 114
м 109 113
н 110 112
о 111 111
п 112 110
р 113 109
с 114 108
т 115 107
у 116 106
ю 117 105
П 118 104
Р 119 103
С 120 102
Т 121 101
У 122 100
```

Областная олимпиада

Теоретический тур

10. Слово (9 баллов)

Задано слово, в котором каждая буква встречается дважды. Указатель находится перед первой буквой слова. Каждый нечетный ход его передвигают на одну позицию вправо, а каждый четный — на букву, парную той, на которую он указывает. Движение указателя заканчивается, когда он оказывается правее последней буквы.

Можно ли утверждать, что алгоритм заканчивается для любого слова? Если да, то докажите это; если нет, то приведите пример, когда это не так.

11. Алгоритм (5 баллов)

Алгоритм X вычисляет значение некоторой функции, зависящей от двух целых неотрицательных чисел ($0 \leqslant m \leqslant n$):

```
Алг цел X(цел m,n)
  нач
    если m=n или m=0
       то знач:=1
       иначе знач:=X(m-1,n-1)+X(m,n-1)
     все
  кон
```

А) Что вычисляет данная функция? (2 балла).

Б) Как связано значение функции и количество рекурсивных вызовов при ее вычислении? (3 балла)

12. Прогрессия (9 баллов)

Задан файл натуральных чисел, каждое из которых не превышает 2 000 000 000. Количество чисел не превышает 200 000. Необходимо определить, можно ли выстроить все эти числа в арифметическую прогрессию.

Опишите алгоритм, который отвечает на данный вопрос и использует возможно меньшее число обращений к исходному файлу. Объем памяти для хранения данных не должен превышать 64 кб.

13. Точки в круге (8 баллов)

На плоскости заданы N точек своими координатами. Опишите и обоснуйте алгоритм нахождения на оси X центра круга наименьшего радиуса, содержащего все данные точки.

14. Невозможная сумма (10 баллов)

Для заданного множества M натуральных чисел N_1, N_2, ..., N_k требуется найти наименьшее натуральное число P, которое не представимо в виде суммы элементов никакого подмножества M. Опишите и обоснуйте алгоритм нахождения числа P.

15. Пути коня (7 баллов)

Задано натуральное число k. Опишите алгоритм нахождения количества различных путей коня с поля `a1` на поле `h8`, содержащих ровно k шагов.

16. Перевозки (4 балла)

В авиакомпании «Небесный тихоход» учет рейсов производится по следующей форме:

№ рейса; Дата и время вылета; Дата и время прилета; Количество пассажиров.

Опишите алгоритм, определяющий по учетным записям период времени, когда в воздухе находилось наибольшее количество пассажиров.

Практический тур

17. Ноты (30 баллов)

Нотная запись состоит из последовательности нот, каждой из которых соответствует один звук. Каждая нота имеет высоту и длительность. Возможны следующие длительности нот: 1 — целая (базовая); 1/2, 1/4, 1/8, 1/16, 1/32, 1/64 — являются соответствующими долями базовой длительности. В этой задаче мы ограничимся длительностями от 1/8 до 1 включительно.

Высоту ноты принято обозначать латинскими буквами (`a` — ля, `b` — си, `c` — до, `d` — ре, `e` — ми, `f` — фа, `g` — соль). Если после буквы стоит знак «`+`» или «`-`», то это означает повышение или понижение звука на полтона, при этом до-диез и ре-бемоль (`c+` и `d-`) означают один и тот же звук. В одной октаве всего 12 полутонов, которые расположены в следующем порядке: `c`, `c+`, `d`, `d+`, `e`, `f`, `f+`, `g`, `g+`, `a`, `a+`, `b` (обратите внимание, что между звуками «ми» и «фа» не целый тон, а половина). В пределах одной октавы мелодию можно записать в виде строки, где подряд указываются ноты и их длительности (мы будем указывать у длительности только знаменатель), например: `c8d8e8f+4g2a-1b16f32`.

При графическом изображении ноты рисуются на нотном стане (из пяти линий) одна за другой. При этом высота ноты изображается высотой ее положения на нотном стане, а длительность определяется значком, изображающим ноту. Пример расположения нот по высоте и длительности приведен на рис. 16.

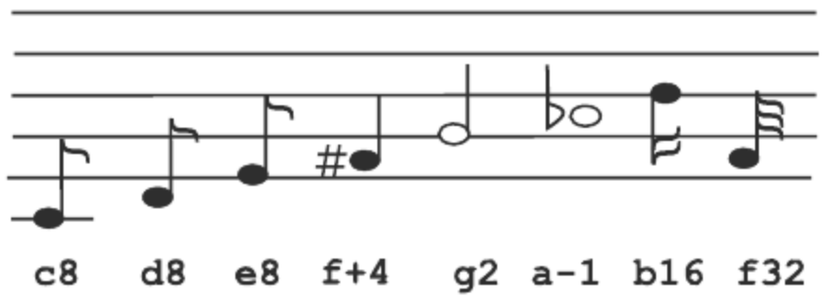


Рис. 16

В нотной записи принято разделять последовательность нот на такты. Продолжительность такта определяется его размером. Например, размер 4/4 означает, что такт содержит последовательность нот, суммарная длительность которых равна четырем четвертям. Допустимы следующие размеры такта: 2/4, 3/4, 4/4, 3/8, 5/8, 7/8. Существуют и другие размеры, но в этой задаче они не используются.

Если нота не умещается в такте, то остаток размещается в следующем такте, а на нотном стане рисуется дуга, соединяющая эти две ноты, которая и означает, что они составляют единое целое. В случае, когда при разбиении получается нота нестандартной длительности, нужно разбить ее на ноты стандартной длительности в порядке убывания и всех их объединить дугой. При разбиении на такты последний такт может оказаться неполным.

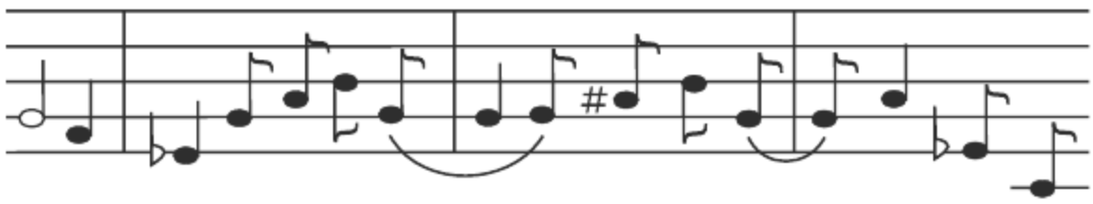
Требуется написать программу, которая по размеру такта (первая строка входного файла) и последовательности нот (вторая строка файла, пробелы отсутствуют) решает одну из задач (на выбор).

А) Рисует эту последовательность на нотном стане по описанным выше правилам, разбитую на такты в соответствии с указанным размером (30 баллов).

Пример входного файла:

```
3/4
g2f4e-4g8a8b8g2a+8b8g4a4e-8c8
```

Результат на экране:



Б) Выводит на экран в буквенной нотации разбитую на такты нотную запись, где такты разделяются между собой пробелами (15 баллов).

Для приведенного выше входного файла ответ будет выглядеть так:

```
g2f4  e-4g8a8b8g8  g4g8a+8b8g8  g8f4e-8c8
```

18. Колье (25 баллов)

Ювелир создает колье из золотых колец, при этом каждое новое кольцо он зацепляет с одним из уже имеющихся в колье колец. Заказчик решил купить вместо колье цепочку. Ювелир хочет, отсоединив от колье часть колец, получить цепочку наибольшей возможной длины.

Напишите программу, которая по информации о порядке соединения колец в колье определяет длину наибольшей цепочки и номера первого и последнего колец в этой цепочке.

Входные данные:

- в первой строке входного файла `necklace.in` находится одно натуральное число n — количество колец в колье, не превышающее 1000;
- следующие строки содержат информацию, к какому кольцу присоединяется очередное кольцо. Так, вторая строка содержит число 1 (второе кольцо может быть присоединено только к первому); третья строка — число 1 или 2; i-я строка содержит одно из натуральных чисел от 1 до $i - 1$.

Выходные данные:

- три натуральных числа через пробел или символ конца строки:
  - d — количество колец в цепочке,
  - b, e — номер первого и последнего кольца одной из таких цепочек.

Результат должен быть выведен в файл `necklace.out`. Он соответствует ситуации, показанной на рис. 17: второе, третье и четвертое кольца сцеплены с первым, а последнее (пятое) — с четвертым.

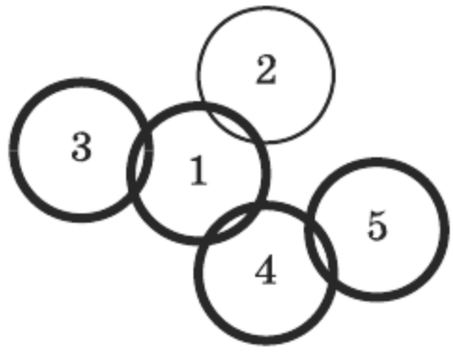


Рис. 17

Пример:

necklace.in	necklace.out
5 1 1 1 4	4 3 5

Городская олимпиада

Теоретический тур

1. Алгоритм (4 балла)

Что будет выведено в результате обращения $X(20, A, 0, 0)$ к следующему алгоритму:

```
алг X(цел N, : вещтаб A[1..N], вещ B,C)
нач
  если N>0
    то X(N-1,A,C,B+A[N])
    иначе вывод(B,C)
  все
кон
```

2. Шестеренки (8 баллов)

На плоскости собрали некий механизм, состоящий из шестеренок с одинаковым шагом (расстоянием между соседними зубьями). Все шестеренки пронумерованы и для каждой из них известны номера шестеренок, с которыми она зацеплена.

Опишите алгоритм, определяющий, можно ли привести весь механизм в движение, вращая какую-нибудь из шестеренок.

3. Делимость (6 баллов)

Даны две строки (не длиннее 255 символов), представляющие собой запись натуральных чисел S и N в некоторой системе счисления с основанием K ($K \leqslant 16$). При $K > 10$ старшие цифры обозначаются латинскими буквами A, B, C, ... Опишите алгоритм, который определяет, делится ли S на N.

4. Система тестов (5 баллов)

Имеется программа, которая по трем заданным целым числам проверяет, можно ли построить равнобедренный треугольник со сторонами, длины которых равны этим числам.

Программа выдает в качестве ответа одно из следующих сообщений: «треугольник не существует», «треугольник не равнобедренный» или «треугольник равнобедренный».

Приведите систему тестов, которые позволили бы в полной мере проверить работоспособность этой программы. Каждый тест состоит из трех чисел и соответствующего ответа программы.

5. Фляга (10 баллов)

На велотреке одновременно стартуют в одном направлении n велосипедистов с номерами от 1 до n. Известны их скорости, выраженные в количестве кругов, которые каждый проезжает за час. Все скорости выражаются натуральными числами. У k-го велосипедиста в момент старта находится фляга. При обгоне если у одного из велосипедистов есть фляга, то она обязательно передается другому. Известно, что скорости таковы, что тройных обгонов не существует.

Опишите алгоритм, определяющий, сколько кругов проедет фляга за один час.

6. Перестановки (6 баллов)

Имеется одномерный массив A длины N, с которым можно производить только одну операцию — перестановку двух любых элементов `SWAP(A[i],A[j])`.

Опишите алгоритм, который за возможно меньшее количество операций:

А) производит «обращение» части массива (элементов с номерами от k до l), т. е. меняет в ней порядок следования элементов на обратный. Например, в символьном массиве `И, Н, Ф, О, Р, М, А, Т, И, К, А` при $k = 3$, $l = 6$ должен получиться массив `И, Н, М, Р, О, Ф, А, Т, И, К, А` (2 балла);

Б) производит циклический сдвиг элементов массива на K элементов. Например из числового массива 10, 9, 8, 7, 6, 5, 4, 3 после циклической перестановки на 3 элемента получается массив 7, 6, 5, 4, 3, 10, 9, 8 (4 балла).

7. Стадион (4 балла)

На стадионе в десятом ряду n мест, расположенных по кругу по всему стадиону. На всех болельщиках «Спартака» надеты бело-красные шарфы, а на болельщиках «Динамо» — синие.

Опишите алгоритм, который позволяет, просматривая одного за другим сидящих в ряду людей, определить длину максимальной последовательности подряд сидящих болельщиков одной команды.

Практический тур

8. Частотный словарь (20 баллов)

Напишите программу, определяющую, какие слова встречаются в заданном тексте наибольшее количество раз.

Входные данные.

Текст на русском языке находится в файле `text.in`. Словом считается любая непустая последовательность русских букв (строчные и прописные буквы различаются), отделенная от других слов любыми другими символами, включая символ конца строки и/или конца файла. Перед первым словом может не быть никакого символа. Каждая строка в файле содержит не более 256 символов. Размер файла не более 100 килобайт.

Выходные данные:

- в первой строке файла должно быть указано одно число — количество вхождений в текст наиболее часто встречающегося слова;
- во второй строке через пробел должны следовать все такие слова.

Результат должен быть выведен в файл `text.out`.

Пример:

text.in	text.out
Я дышу, и, значит, я люблю, Я люблю, и, значит, я живу.	2 Я и значит я люблю

9. Электронная таблица (30 баллов)

Имеется прямоугольная таблица, содержащая M строк ($M \leqslant 26$) и N столбцов ($N \leqslant 10$). Строки обозначаются строчными латинскими буквами от `a` до `z`, а столбцы — цифрами от 0 до 9. Каждая ячейка имеет двухсимвольный адрес, например, `b7` или `z0`. Ячейка содержит или целочисленную константу, или формулу из одной операции (`+`, `-`, `*`), где операндами являются только адреса ячеек, например `k6-g5`. Таблица содержит цикл, если существует последовательность формул, каждая из которых содержит адрес следующей в этой последовательности формулы, а последняя формула ссылается на первую.

Составьте программу, которая проверяет заданную таблицу на наличие циклов, и если их нет, то просчитывает таблицу, т. е. заменяет все стоящие в ячейках формулы на их значения. Известно, что константы и вычисляемые значения формул не превосходят по модулю 32 000. Исходные данные корректны.

Входные данные.

Во входном файле `etable.in` находится таблица. Каждая строка файла соответствует одной строке этой таблицы. В строке перечисляются данные ячеек (константы или формулы), разделяемые одним или несколькими пробелами. Разделители внутри константы или формулы не допускаются.

Выходные данные.

В файл `etable.out` должно быть выведено сообщение «цикл есть» и адрес любой ячейки, входящей в цикл, либо просчитанная таблица в формате, аналогичном формату входного файла. Допускается вывод выровненной по столбцам таблицы на экран.

Время работы программы для каждого теста не должно превышать 10 с.

Пример 1:

etable.in	etable.out
11 a0-b0 b1*b1 a0+a1	цикл есть b1

Пример 2:

etable.in	etable.out
11 a0-b0 b1*b1 5	11 -14 25 5

Областная олимпиада

Практический тур (день первый)

10. Упорядочивание файла (20 баллов)

Дан файл целых чисел. Составьте программу, которая создает новый файл, содержащий те же числа в порядке неубывания.

Входные данные.

Входной текстовый файл `sort.in` содержит не более 160 300 положительных чисел типа `integer`, разделенных пробелом.

Выходные данные.

Имя выходного файла — `sort.out`. Формат — такой же, как у входного файла.

Время тестирования: 30 с на каждый тест.

11. Архипелаг в океане (40 баллов)

В океане расположен архипелаг из N островов, каждый из которых имеет форму выпуклого многоугольника. Острова не соприкасаются и не пересекаются. Эти острова необходимо соединить между собой мостами так, чтобы от любого острова архипелага можно было добраться до любого друго-

го. Каждый мост должен соединять пару островов, при этом суммарная длина мостов должна быть минимальной.

Входные данные:

- в первой строке входного файла `archi.in` указано число N — количество островов ($0 < N < 50$);
- в последующих N строках идет описание островов в следующем формате: K, x_1, y_1, x_2, y_2, ..., x_k, y_k — количество вершин и координаты вершин в порядке обхода по контуру, разделенные пробелом ($K \leqslant 20$, координаты вершин — числа типа `integer`).

Выходные данные.

В выходной файл `archi.out` по одному в строке должны быть выведены два числа — количество мостов и их суммарная длина.

Время тестирования: 10 с на каждый тест.

Пример 1:

archi.in	archi.out
2 4 -2 -2 -2 2 2 2 2 -2 3 3 3 5 3 4 4	1 1.4142

Пример 2:

archi.in	archi.out
3 4 -2 -2 -2 2 2 2 2 -2 3 -3 0 -5 -1 -5 1 3 6 -5 8 -4 8 -5	2 6

Примечание: полное решение задачи для случая двух островов оценивается в 20 баллов.

Практический тур (день второй)

12. Атомный реактор (50 баллов)

Пластины урана загружаются в атомный реактор стопками по N пластин. Чтобы эффективно контролировать процесс реакции, между пластинами вставляют датчики. Не допускается, чтобы более K подряд идущих пластин в стопке шли без датчиков между ними. Требуется определить, сколько различных стопок пластин можно составить, соблюдая эти условия, если в каждой стопке должно быть ровно M датчиков.

Все датчики должны размещаться только между пластинами по одному.

Входные данные.

В одной строке входного файла `atomi.in` указаны через пробел три числа: N, M, K ($0 < N < 100$, $1 \leqslant M < N$, $1 \leqslant K \leqslant N$).

Выходные данные.

В выходном файле `atomi.out` должно быть указано единственное число — ответ. Результат может быть выведен на экран.

Время тестирования: 10 с на каждый тест.

Пример:

atomi.in	atomi.out
5 2 2	3

13. Грязные предвыборные технологии (50 баллов)

В Анчурии предстоят президентские выборы. На эту должность претендуют K кандидатов. Им присвоены номера от 1 до K. Всего в Анчурии N избирателей.

Группа поддержки кандидата с порядковым номером R провела опрос всех жителей Анчурии и получила N опросных листов, в каждом из которых номера претендентов перечислены в порядке убывания симпатий. На основании

этих сведений разрабатывается план приведения к власти нужного кандидата. Для этого используются два метода. Первый — дискредитация любого кандидата, что требует D условных денежных единиц; второй — подкуп любого избирателя, что требует P условных единиц. Если какой-то кандидат дискредитирован, то он снимает свою кандидатуру. Если какой-то избиратель подкуплен, то он голосует за кандидата R. Побеждает кандидат, набравший большинство голосов (хотя бы на один голос больше, чем любой другой).

Напишите программу, которая вычисляет наименьшую возможную сумму, необходимую для успешного проведения предвыборной кампании.

Входные данные:

- в первой строке входного файла `election.in` содержатся пять целых чисел, разделенных пробелом: N, K, R, D, P ($1 \leqslant K \leqslant 10$, $K \leqslant N \leqslant 5000$, $1 \leqslant R \leqslant K$, $0 \leqslant D \leqslant 30\,000$, $0 \leqslant P \leqslant 30\,000$);
- в последующих N строках записаны опросные листы, каждый из которых представляет собой K чисел, разделенных пробелами (перестановка чисел от 1 до K).

Выходные данные.

В выходном файле `election.out` должно быть указано единственное число — ответ. Результат может быть выведен на экран.

Время тестирования: 15 с на каждый тест.

Пример:

election.in	election.out
55 3 1 15 10 2 1 3 3 1 2 2 1 3 2 3 1 3 2 1	25

ОЛИМПИАДЫ 2000/01 УЧЕБНОГО ГОДА

Городская олимпиада

Теоретический тур

1. Последовательность (7 баллов)

С имеющейся последовательностью из N нулей и единиц разрешается проделывать следующую операцию: заменить несколько (может быть, и ни одного) идущих подряд первых слева нулей на единицы, а следующую за ними цифру инвертировать. Например из 0010 можно получить 1010, либо 1110, либо 1100.

Как за наименьшее количество таких операций превратить N единиц в N нулей? Ответ обоснуйте.

2. Центральная симметрия (4 балла)

На плоскости заданы своими координатами N точек. Опишите эффективный алгоритм, определяющий, является ли данное множество точек центрально-симметричным.

3. Юстас и Алекс (8 баллов)

Юстас и Алекс договорились о четырех сигналах A, $Б$, $В$ и $Г$, передача которых занимает 1, 2, 2 и 3 секунды, соответственно. За один сеанс связи передается одно сообщение, состоящее из произвольной комбинации данных сигналов. Помогите разведке определить минимальный промежуток времени t, такой, что по крайней мере 1000 различных сообщений имеют время передачи не более t.

4. Меткий стрелок (8 баллов)

За квадратным фанерным щитом размером 10×10 метров закреплен воздушный шарик диаметром 10 сантиметров. Ковбой стреляет сквозь щит, стараясь поразить шарик. После каждого выстрела ему сообщают, ближе или дальше от шарика прошла его последняя пуля по отношению к предпоследней. Опишите алгоритм, позволяющий ковбою заведомо поразить шарик за наименьшее возможное количество выстрелов. Размерами пули пренебречь.

5. Гигантомания (7 баллов)

Для встречи третьего тысячелетия была изготовлена гирлянда из 2000 лампочек, висящих на одном проводе, причем каждая лампочка (кроме крайних) при подвешивании гирлянды оказывается висящей на 1 мм ниже, чем среднее арифметическое высоты подвеса ее соседей справа и слева. Опишите алгоритм, который по заданным расстояниям до земли первой и последней лампочек определяет, коснется ли гирлянда земли.

6. Алгоритм (3 балла)

Что делает следующий алгоритм?

```
Алг цел X (нат a,b)
нач
  если a=b
    то знач:=b
    иначе
      если    a < b
        то    знач:=X(b,a)
        иначе    знач:=X(a-b,b)
      все
  все
кон
```

7. Буквы (4 балла)

Опишите алгоритм, определяющий, можно ли из букв первого заданного слова составить другое заданное слово. Количество вхождений каждой буквы во второе слово не должно превышать количества ее вхождений в первое слово.

Практический тур

8. Робот (25 баллов)

В начале координат стоит робот, смотрящий в положительном направлении оси Y. Робот может выполнять команды: «повернуть на 72° вправо» и «проехать один километр вперед». Проехать расстояние, меньшее одного километра, робот не может. На преодоление одного километра требуется один

литр топлива. На поворот топливо не расходуется. Требуется подвести робота как можно ближе к заданной точке. Полностью расходовать топливо необязательно.

Напишите программу, которая определяет минимальное расстояние, которое может остаться между роботом и точкой после выполнения последней команды, если известны координаты точки и начальное количество топлива.

Входные данные:

- в первой строке входного файла `robot.in` указано начальное количество топлива — натуральное число, не превышающее 300;
- во второй строке через пробел указаны координаты точки — действительные числа, не превышающие 100 по модулю.

Допускается ввод с клавиатуры.

Выходные данные.

В выходном файле `robot.out` должно быть одно число — результат с точностью 0.00001. Допускается вывод на экран.

Время работы программы: 10 с на тест.

Пример:

robot.in	robot.out
2 0.30902 2	0.04894

9. Интерпретатор (25 баллов)

Исходный текстовый файл содержит программу, состоящую из команд, каждая из которых находится в отдельной строке и имеет вид:

```
<операция><результат><операнд1><операнд2>
```

Операнд — это переменная или целая константа. Одна переменная может несколько раз участвовать в одной операции. Второй операнд или сразу оба могут отсутствовать,

в зависимости от формата команды. Допускаются следующие форматы команд:

<**арифметическая операция**><результат><операнд1><операнд2>
<**операция сравнения**><*N*строки><операнд1><операнд2>
go<*N*строки>
<**операция присваивания**><результат><операнд1>

В качестве арифметической операции могут быть использованы знаки +, –, *. Константы и вычисляемые значения не превосходят по модулю 32 000.

В качестве операции сравнения могут быть использованы знаки ==, <>, >, <, >=, <=. При этом если результат операции является истинным, то происходит переход на команду, указанную номером строки (*Nстроки*); в противном случае обрабатывается следующая строка файла.

Операция go является безусловным переходом.

Строки файла нумеруются с 1; переход возможен только вперед.

Для обозначения операции присваивания используется знак =.

Составьте программу, которая проверяет программу на отсутствие ошибок и если их нет, то интерпретирует (выполняет) команды, выдает на экран имя и значение той переменной, которая в процессе интерпретации программы была использована последней в поле результата. Интерпретация осуществляется до первой ошибки.

Входные данные.

Текст программы находится во входном файле inter.in. Между полями команды допускаются один или несколько пробелов. Имена переменных содержат не более шести символов — строчных латинских букв и цифр (первая — буква). Условный или безусловный переход может быть только вперед на одну из последующих команд. Число команд в программе не превышает 1000.

Выходные данные.

Результат — одно число выводится в файл inter.out. При обнаружении ошибки выводится слово «error». При этом на экран должно быть выведено сообщение, например:

```
В команде N 3 переменная a2bc не определена.
```

Время работы программы: 10 с на тест.

Пример:

inter.in	Комментарий	inter.out
= a1 5 > 4 a1 -1 - a1 0 a1 = a2 a1	a1:=5 если a1>-1 то переход на 4 a1:=0-a1 a2:=a1	5

Областная олимпиада

10. Раскраска ладей (50 баллов)

На доске размером $N \times N$ стоят несколько ладей. Две ладьи бьют друг друга, если они стоят на одной вертикали или горизонтали и между ними нет других ладей.

Требуется написать программу, определяющую наименьшее количество цветов, в которые можно раскрасить всех ладей так, чтобы одноцветные ладьи не били друг друга.

Входные данные:

- в первой строке входного файла `color.in` указано целое число N ($1 \leqslant N \leqslant 100$);
- каждая из последующих N строк длиной N символов, состоящая из нулей и единиц, служит для описания расположения ладей на шахматной доске по горизонталям. Клетка с ладьей обозначается «1»; без ладьи — «0».

Выходные данные.
В выходном файле `color.out` должно быть одно число — искомое количество цветов.

Время работы программы: 10 с на тест.

Пример:

color.in	color.out
3 111 010 100	2

Городская олимпиада

Теоретический тур

1. Разделяющий элемент (4 балла)

Имеется массив $A[N]$ из положительных чисел. Назовем элемент массива $A[i]$ разделяющим, если сумма элементов от $A[1]$ до $A[i-1]$ меньше половины всей суммы элементов массива, а сумма элементов от $A[1]$ до $A[i]$ — не меньше. Опишите алгоритм поиска разделяющего элемента за возможно меньшее число обращений к памяти.

2. Вычисление суммы (7 баллов)

В языке программирования ARSEQ, кроме обычных паскалевских операций, есть операция ***S***(*b*, *e*, *s*), вычисляющая сумму арифметической прогрессии от b до e с шагом s. Напишите программу на языке ARSEQ, вычисляющую за возможно меньшее количество операций сумму всех натуральных чисел от N до M, которые не делятся ни на 2, ни на 3.

3. Вычеркивание (8 баллов)

А) Сколько различных трехзначных чисел можно получить из заданного натурального N, вычеркивая цифры из его десятичной записи? (4 балла).

Б) Сколько различных k-значных чисел можно получить из заданного натурального N, вычеркивая цифры из его десятичной записи? (4 балла).

4. Клетки таблицы (5 баллов)

Клетки прямоугольной таблицы размером $M \times N$ пронумерованы от 1 до $M \times N$ по спирали так, как показано на рис. 18. Опишите алгоритм, определяющий местонахождение клетки, в которую записано последнее число.

1	2	3	4
10	11	12	5
9	8	7	6

Рис. 18

5. Преобразование текста (10 баллов)

Рассмотрим всевозможные слова на алфавите из 3 букв (a, b, c). За один шаг в слове разрешается производить одну из следующих замен:

aa ↔ пустое слово;
bb ↔ пустое слово;
cc ↔ пустое слово;
ab ↔ c; ba ↔ c;
bc ↔ a; cb ↔ a;
ac ↔ b.

Все замены — двусторонние, например можно вставить в любое место слова любую пару одинаковых букв.

Два слова называются эквивалентными, если от одного из них можно перейти к другому за какое-то число шагов.

Приведите список всех попарно неэквивалентных слов. Ответ обоснуйте.

6. Алгоритм (8 баллов)

[illegible]

```
    a:=b; c:=0
    нц пока a>0
        e:=a mod 2
        a:=a div 2
        c:=c+d*e
        d:=d*2
    кц
кон
```

7. Рождественские подарки (10 баллов)

Имеются шары k различных цветов, из которых составляются подарочные наборы из n шаров. При этом для каждого цвета i указано допустимое количество a_i шаров этого цвета в наборе.

Опишите алгоритм подсчета количества различных возможных наборов.

8. Ломаная (8 баллов)

Ломаная на плоскости задана координатами своих вершин вдоль ее обхода. Известно, что если звенья ломаной пересекаются, то не более чем в одной точке. Опишите алгоритм, определяющий, на какое количество областей ломаная делит плоскость.

Практический тур

9. Ограда (30 баллов)

Для ограждения игровой площадки в детском саду были привезены несколько секций забора. Ими требуется огородить площадку, имеющую форму выпуклого многоугольника. Напишите программу, определяющую, какую площадь может иметь этот многоугольник.

Входные данные:

- в первой строке входного файла `fence.in` находится число N — количество секций (N может равняться только 4 или 5);
- в последующих N строках по одному расположены вещественные числа — длины секций.

Допускается ввод с клавиатуры. При чтении исходных данных с клавиатуры оценка может быть снижена.

Выходные данные.

В выходном файле `fence.out` должно быть два числа — наибольшее и наименьшее значение площади многоугольника с точностью 0.0001.
Если из данных отрезков невозможно сложить выпуклую фигуру, то следует вывести число −1.
Допускается вывод на экран.

Время работы программы: 5 с на тест.

Пример:

fence.in	fence.out
3 3 4 5	6 6

10. Кубики (20 баллов)

На некоторых клетках шахматной доски расположены столбики кубиков. На доску посмотрели спереди и записали высоту видимых столбиков слева направо (всего 8 чисел). Затем доску повернули на 90° и процедуру повторили. Составьте программу, по получившимся 16 числам определяющую наибольшее и наименьшее количество кубиков, которое могло пойти на строительство всего сооружения. В случае некорректности входных данных выдать соответствующее сообщение.

Входные данные:

- в первой строке входного файла `blocks.in` записаны 8 целых чисел в диапазоне от 0 до 50 000, разделенных пробелами (одним или несколькими);
- во второй строке входного файла записаны еще 8 целых чисел в том же диапазоне, разделенных пробелами.

Выходные данные.

В файл `blocks.out` следует вывести два числа, разделенные пробелом — наибольшее и наименьшее количество кубиков или −1, если указанное сооружение сложить невозможно.

Время работы программы: 10 с на тест.

Пример:

blocks.in	blocks.out
2 2 0 0 0 0 0 0 2 1 0 0 0 0 0 0	6 5

Областная олимпиада

11. Таблица (30 баллов)

В каждой клетке таблицы $N \times N$ ($1 \leqslant N \leqslant 1\,000\,000$) записаны целые неотрицательные числа так, что число, стоящее на пересечениии i-й строки и j-го столбца, является наименьшим из тех, которых нет в i-й строке на местах с 1 по (j−1) и в j-м столбце на местах с 1 по (i−1). Фрагмент начала таблицы приведен на рис. 19.

0	1	2	3	4	
1	0	3	2	5	
2	3	0	1	6	
3	2	1	0	7	

Рис. 19

Напишите программу, которая по заданным i и j определяет число, стоящее на пересечении i-й строки и j-го столбца.

Входные данные.

В первой строке входного файла `table.in` указаны через пробел два числа, не превышающих 300, — номер строки и столбца.

Выходные данные.

В выходном файле `table.out` должно быть одно число — результат.

Время работы программы: 5 с на тест.

Пример:

table.in	table.out
3 4	1

12. Умножение дроби (30 баллов)

Задана некоторая правильная периодическая дробь Q и натуральное число N. При изображении дроби Q периодическая часть заключается в круглые скобки.

Напишите программу, которая определяет результат умножения Q на N, т. е. непериодическую часть и минимальный период числа $Q \times N$. В случае получения результата умножения в виде конечной дроби скобки опускаются.

Входные данные:

- в первой строке входного файла `fract.in` указана дробь в вышеописанном формате. Количество цифр, используемых для описания дроби, не превосходит 100;
- во второй строке — натуральное число N, не превышающее 1000.

Выходные данные.

В выходном файле `fract.out` должно быть одно число — результат.

Допускается вывод на экран.

Время работы программы: 5 с на тест.

Пример:

fract.in	fract.out
0.1(6) 3	0.5

13. Система дорог (50 баллов)

В государстве N городов. Требуется соединить их системой дорог наименьшей общей протяженности. Дороги могут иметь пересечения и развилки. Напишите программу, которая по заданным координатам городов рассчитывает суммарную длину искомой сети дорог.

N может принимать только значения 3 (20 баллов) или 4 (30 баллов).

Входные данные:

- в первой строке входного файла `roads.in` задается число N;
- в последующих N строках указаны пары целых чисел — координаты городов. Числа не превышают по модулю 10 000 и разделены пробелом.

Выходные данные.

Результат (одно число с точностью 0.0001) должен быть выведен в файл `roads.out`.

Время работы программы: 5 с на тест.

Пример:

roads.in	roads.out
3 1 1 3 1 2 4	5.196

Городская олимпиада

Теоретический тур

1. Циклический состав (6 баллов)

Представьте, что вы находитесь в поезде, в каждом вагоне которого есть доска, тряпка и мел. Вы можете писать на доске, что хотите, и стирать с доски. Кроме того, вы можете переходить из вагона в вагон в любом направлении. До вас в поезде уже кто-то побывал и оставил на некоторых досках какие-то надписи. Опишите алгоритм, позволяющий вам определить, зациклен ли состав, т. е. попадете ли вы в исходный вагон при движении в одном направлении, или нет.

2. Тасовка (10 баллов)

Под тасовкой колоды карт мы понимаем следующее: несколько карт берутся с верха колоды и вставляются без изменения порядка, но не обязательно подряд, в оставшуюся часть колоды.

А) Докажите, что в колоде из 2^n карт можно n тасовками изменить порядок карт на противоположный.

Б) Докажите, что в колоде из 2^n карт можно n тасовками изменить порядок карт на произвольный.

3. Шахматный турнир (6 баллов)

В шахматном турнире каждый из n участников сыграл с каждым одну партию. Докажите, что участников можно выстроить в ряд так, чтобы ни один из них не проиграл непосредственно стоящему за ним. Опишите алгоритм такой расстановки.

4. Барабан (7 баллов)

Приз лежит за дверью, перед которой стоит барабан с вертикальной осью вращения и четырьмя отверстиями по бокам. Внутри каждого отверстия стоит переключатель, имеющий

два положения: «верх» и «низ». Игрок может засунуть руки одновременно в любые два отверстия, пощупать, как стоят переключатели, и произвольно переключить один или оба сразу (а можно их и не переключать). Затем барабан быстро вращается и после его остановки уже нельзя определить, в каких отверстиях переключатели трогали в предыдущий раз. Дверь открывается, когда все переключатели встанут в одно положение.

Опишите алгоритм, пользуясь которым, игрок обязательно получит приз.

5. Функция (5 баллов)

Функция натурального аргумента вычисляется по следующему алгоритму:

```
function F(n: integer): integer;
begin
  var k: integer;
  k:=n div 2;    {k — частное от деления n на 2}
  if n<3 then f:=0
         else if n=3 then F:=1
                     else f:= F(k)+ F(n-k);
end.
```

Найдите точки минимума и максимума, а также значения функции в этих точках. Точка называется «точкой максимума», если значения функции в двух соседних точках (слева и справа) строго меньше, чем в ней. Точка является точкой минимума, если значения функции в двух соседних точках (слева и справа) строго больше, чем в ней.

6. Слова (7 баллов)

Имеется словарь, состоящий из n слов, а также символьная строка длины m. Некоторые идущие подряд символы в строке могут образовывать имеющееся в словаре слово. Опишите алгоритм разделения строки (если это возможно) на максимальное количество частей, каждая из которых является словом.

7. Векторы (8 баллов)

Пусть переменные $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{e}$, $\boldsymbol{f}$, $\boldsymbol{g}$, $\boldsymbol{h}$ обозначают векторы на плоскости, а x, y, z, t — вещественные числа. Над векторами можно выполнять следующие операции:

- вычислять сумму $\boldsymbol{c}$ векторов $\boldsymbol{a}$ и $\boldsymbol{b}$ с помощью процедуры `СУММА(a,b,c);`
- вычислять произведение $\boldsymbol{b}$ вектора $\boldsymbol{a}$ на число x с помощью процедуры `ПРОИЗВЕДЕНИЕ(a,x,b);`
- вычислять длину x вектора $\boldsymbol{a}$ с помощью процедуры `МОДУЛЬ(a,x);`
- раскладывать вектор $\boldsymbol{c}$ по базису из векторов $\boldsymbol{a}$ и $\boldsymbol{b}$ (т. е. находить такие числа x и y, что $\boldsymbol{c} = x\boldsymbol{a} + y\boldsymbol{b}$) с помощью процедуры `РЕШЕНИЕ_СИСТЕМЫ(a,b,c,x,y);`

Пусть некоторый треугольник ABC задан векторами $\boldsymbol{a} = AB$ и $\boldsymbol{b} = AC$. Тогда, например, вектор $\boldsymbol{c}$ медианы AM, где M — середина стороны BC, можно найти с помощью следующей программы:

```
СУММА(a,b,c);
ПРОИЗВЕДЕНИЕ(c,0.5,c);
```

Составьте программу:

А) нахождения какого-нибудь вектора, направленного вдоль биссектрисы $\angle BAC$;

Б) нахождения вектора AO, где O — центр вписанной в треугольник ABC окружности.

Практический тур

8. Перекраска (20 баллов)

В N пронумерованных ячейках лежат фишки трех цветов. Фокусник каждую минуту, не глядя на фишки, называет два числа от 1 до N. Если фишки, лежащие в ячейках с названными номерами, одинаковы по цвету, то их не трогают, иначе их заменяют фишками третьего (не совпадающего с ними) цвета. Через некоторое время фокусник прекращает называть номера и демонстрирует зрителям результат. Фокус считается удавшимся, если все фишки оказались одно-

цветными. Вам надо написать программу, которая по заданному N выдает последовательность пар номеров, которые должен говорить фокусник для достижения удачи независимо от начальных цветов, или констатирует, что такой последовательности не существует.

M не должно превышать 100 000.

Входные данные.

Число N ($1 < N < 10\,000$) находится в файле `focus.in`.

Допускается ввод с клавиатуры.

Выходные данные:

- в первой строке выходного файла `focus.out` должно быть число M — количество пар номеров, называемых фокусником, или -1, если гарантировать успех фокуса при указанных входных данных и ограничениях нельзя;
- в последующих M строках должны быть через пробел выведены соответствующие номера.

Время работы программы: 5 с на тест.

Пример 1:

focus.in	focus.out
2	1 1 2

Пример 2:

focus.in	focus.out
3	-1

9. Поисковая система (20 баллов)

Есть информационная система, содержащая данные о группе людей. Время от времени в этой системе требуется найти информацию об отдельных лицах или группе лиц. Вам необходимо создать удобную поисковую программу, позволяющую задать значение одного или нескольких имеющихся

в системе признаков и формирующую на выходе нумерованный список людей, удовлетворяющих этим признакам.

Имеющаяся информационная система состоит из двух файлов: `personal.dat` и `region.dat` (оба в кодировке DOS).

Текстовый файл `personal.dat` содержит таблицу с данными по людям. Запись, относящаяся к одному человеку, находится в одной строке файла и имеет следующий формат:

- фамилия (20 символов);
- имя (10 символов);
- пол (1 символ: «м» или «ж»);
- год рождения (4 символа, например `1975`);
- код региона проживания (2 символа — целое число).

Перечисленные поля идут подряд от начала строки и всегда заполнены. Если реальные фамилия или имя короче, то они дополняются справа пробелами. Таким образом, строки содержат по 37 символов. Порядок людей в файле — произвольный.

Текстовый файл `region.dat` содержит таблицу регионов. Запись, относящаяся к одному региону, находится в одной строке файла и имеет следующий формат:

- код региона проживания (2 символа — целое число);
- наименование региона (одно слово длиной до 40 символов).

Однозначный код региона всегда дополняется слева нулем (например `04`). Наименование начинается с 3-й позиции и пробелами справа не дополняется. Список регионов упорядочен по возрастанию кода.

Ограничения:

- количество записей в файле `personal.dat` не превышает 10 000;
- количество записей в файле `region.dat` не превышает 100.

Входные данные.

Программа должна запрашивать значения перечисленных выше признаков для поиска (отбора) записей с заданными значениями признаков. Год рождения может задаваться одним числом или двумя числами — границами годов рождения (границы включаются в диапазон). Любой из признаков может быть не задан. В этом случае записи отбираются только по заданным признакам. Например, если задано только значение признака «пол», то выбираются все записи, в которых значения пола совпадают с заданным. При выборке фамилия и/или имя должны точно совпадать с заданными.

Если в программе реализованы не все признаки (например, позволяется ввести только фамилию и/или пол), то отбор записей идет только по значениям реализованных признаков.

Выходные данные.

Список людей, соответствующих заданию, должен быть выведен в файл `poisk.out` по одной записи в строке. На экране должно отображаться количество записей выборки.

Оцениваются правильность работы программы, количество реализованных признаков, удобство интерфейса программы.

Время работы программы: 5 с на тест.

Пример 1

Файл `personal.dat`:

```
Белков              Александр м197707
Белкова             Александраж197907
Проскуряков         Петя      м200376
Белков              Михаил    м198707
```

Файл `region.dat`:

```
07Кабардино-Балкарская
76Ярославская
```

При вводе шаблона поиска:

Запрос на экране	Ответ с клавиатуры
Фамилия:	Белков
Имя:	
Пол:	м
Год:	1970-2000
Регион:	

На экране: 2

В файле `poisk.out`:

```
Белков            Александр м197707
Белков            Михаил    м198707
```

Областная олимпиада

10. Разделение пограничных городов (20 баллов)

На границе двух государств, только что обретших свою независимость друг от друга, расположены два города, представляющих собой на карте два выпуклых непересекающихся многоугольника. По соглашению между президентами государств эти города отходят к разным государствам, и между ними должна быть проведена прямолинейная граница. Напишите программу, которая рассчитывает, как следует провести искомую границу.

Входные данные:

- в первой строке входного текстового файла `bound.in` находится число N — количество вершин многоугольника, соответствующего первому городу ($3 \leqslant N \leqslant 100$);
- в последующих N строках находятся пары чисел x_i, y_i типа `real`, разделенных пробелами, — координаты вершин первого многоугольника, перечисленные вдоль обхода контура;
- в следующей строке K — количество вершин многоугольника, соответствующего второму городу ($3 \leqslant K \leqslant 100$);
- в последующих K строках находятся пары чисел x_i, y_i, разделенных пробелами, — координаты вершин второго многоугольника, перечисленные вдоль обхода контура.

Выходные данные:

- в первой строке выходного текстового файла `bound.out` должна быть выведена пара чисел, разделенных пробелом, — координаты произвольной точки, лежащей на прямой — искомой границы между городами;
- во второй строке — пара чисел, разделенных пробелом, — координаты вектора, направленного вдоль искомой границы.

Время работы программы: 2 с на тест, не считая времени на ввод-вывод.

Пример:

bound.in	bound.out
4 0 0 1 0 1 1 0 1 3 0 2 1 2 0.5 3	1.5 2 -1 -0.5

11. Кубик на доске (30 баллов)

Кубик находится на некоторой клетке обычной шахматной доски. При этом он закрывает ровно одну клетку доски. На каждой грани кубика записано какое-то целое число из диапазона от 0 до 1000.

Кубик может перемещаться по доске путем перекатывания через ребро на одну из соседних клеток. Каждому возможному пути перемещения (последовательности перекатываний) поставим в соответствие сумму чисел, которые оказывались на нижней грани кубика при всех его перекатываниях. Каждое число суммируется столько раз, сколько раз оно оказывалось снизу. Числа, соответствующие начальному и конечному положениям кубика, также суммируются.

Требуется написать программу, которая находит путь с минимальным значением суммы до заданной клетки доски.

Входные данные:

- в первой строке входного файла `cube.in` через пробел указано начальное и конечное местоположение кубика в шахматной нотации;
- во второй строке находятся шесть чисел, расположенных на гранях кубика в исходном положении, в следующем порядке: сверху, снизу, спереди, сзади, слева, справа. Числа разделены пробелами. Считается, что передняя грань кубика повернута в сторону первой горизонтали, задняя — в сторону последней горизонтали, левая — в сторону первой вертикали, правая — в сторону последней вертикали.

Выходные данные:

- в первой строке выходного файла `cube.out` должна быть выведена искомая сумма;
- во второй строке — последовательность клеток вдоль оптимального пути в шахматной нотации, разделенных пробелами.

Время работы программы: 5 с на тест.

Пример:

cube.in	cube.out
e2 e3 0 1 1 8 1 2	5 e2 d2 d1 e1 e2 e3

12. Экзамен в Африке (30 баллов)

Профессор одного африканского университета принимает экзамен у группы из k студентов, имеющих одинаковый уровень умственных способностей. Профессор очень торопится и хочет за минимальное время оценить умственные способности своих студентов.

Он подготовил n задач, сложность которых оценил от 1 до n баллов. Все задачи имеют разную сложность. Умствен-

ные способности студента оцениваются в r баллов, если он может решить задачу сложности r, но не может решить задачу сложности $r + 1$. Предполагается, что если студент не решил какую-то задачу, то он не может решить никакую задачу более высокой сложности, а если студент решил некоторую задачу, то он может решить все более простые задачи. Если он не может решить задачу сложности 1, то его способности оцениваются в 0 баллов. Если же студент решил задачу сложности n, то его способности равны n.

Экзамен проходит по следующей схеме: профессор выбирает какую-либо задачу и дает ее студенту. Если тот справляется с задачей за одну минуту, то профессор дает ему еще одну задачу; если нет, то студент покидает аудиторию.

Напишите программу, отвечающую на вопрос: за какое минимальное время профессор может гарантированно определить умственные способности своих студентов?

Входные данные.

В первой строке файла `exam.in` через пробел записаны числа k и n ($1 \leqslant k \leqslant 10$; $1 \leqslant n \leqslant 1000$).

Допускается ввод с клавиатуры.

Выходные данные.

Результат должен быть выведен в файл `exam.out` или на экран.

Время работы программы: 3 с на тест.

Пример:

exam.in	exam.out
2 3	2

Комментарий. Первому студенту предлагается вопрос сложности 2. Если он не справляется с заданием, то второму студенту предлагается вопрос сложности 1; иначе первому студенту предлагается вопрос сложности 3.

Городская олимпиада

Теоретический тур

1. Ряд из нулей и единиц (8 баллов)

Имеется ряд из 2003 символов — нулей и единиц. За один ход разрешается поменять на противоположные значения (нуль на единицу, а единицу на нуль) два идущих один за другим символа. Известно, что существует алгоритм, состоящий не более чем из n ходов, который переводит имеющийся ряд в ряд, состоящий из одних нулей, кроме одной единицы, стоящей на первом месте. Также известно, что существует алгоритм, состоящий не более чем из n ходов, который переводит имеющийся ряд в ряд, состоящий из одних нулей, кроме одной единицы, стоящей на последнем месте. Докажите, что существует алгоритм, состоящий не более чем из $2n$ ходов, который переводит имеющийся ряд в ряд, состоящий из одних нулей, кроме одной единицы, стоящей ровно посередине.

2. От края до края (7 баллов)

Каждая клетка шахматной доски покрашена в один из k цветов. Опишите алгоритм поиска пути, проходящего по клеткам одного цвета, идущего от левого края доски до правого. Путь должен состоять из последовательности шагов, каждый из которых переходит от клетки к клетке через их общую сторону.

3. Словарь племени Мумбо-Юмбо (10 баллов)

В алфавите племени Мумбо-Юмбо всего две буквы — `А` и `У`. Словом является любая последовательность n букв, в которой нет двух рядом стоящих букв `У`. В словаре племени слова отсортированы по алфавиту (в лексикографическом порядке). Так, первым при $n = 5$ указано слово `ААААА`, а последним — `УАУАУ`. Опишите алгоритм, который по заданному n определяет:

А) сколько всего слов длины n имеется в словаре племени Мумбо-Юмбо (5 баллов);

Б) слово, которое стоит в словаре на k-м месте (5 баллов).

Пример.

Пусть $n = 4$, $k = 3$. Тогда словарь племени такой: `АААА`, `ААУ`, `ААУА`, `АУАА`, `АУАУ`, `УАА`, `УААУ`, `УАУА`. На третьем месте стоит слово `ААУА`.

4. 3 из 5 (8 баллов)

А) Составьте программу, которая в заданном числовом интервале $[a, b]$ подсчитывает количество чисел, делящихся ровно на три из пяти заданных простых чисел (4 балла).

Б) Сделайте то же самое, используя из арифметических операций только операцию сложения (4 балла).

5. Ломаная (8 баллов)

На плоскости заданы координаты n точек (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), причем $x_1 < x_2 < \ldots < x_n$. Опишите алгоритм построения функции, график которой проходит через все эти точки. Функция должна быть равна алгебраической сумме линейных функций и модулей линейных функций. На интервале $(-\infty, x_1)$ функция должна равняться y_1, а на интервале $(x_n, +\infty)$ функция должна равняться y_n.

Пример.

Если $n = 2$ и координаты точек: (0, 0), (1, 2), то искомая функция: $|x| + x - |x-1| - x + 1$. (Возможны и другие ответы.)

6. Память (4 балла)

У специализированного компьютера память объема 1 Гигабайт состоит из 256 32-разрядных регистров и 511 Кб кэш-памяти, а остальное — оперативная память. Найдите, сколько мегабайт она составляет.

7. Нумерация по диагонали (6 баллов)

В клетки бесконечной вправо и вниз таблицы вписаны по порядку по диагоналям (рис. 20) натуральные числа.

1	2	4	7	11	...
3	5	8	12	...	
6	9	13	...		
10	14	...			
15	...				
...					

Рис. 20

А) Опишите алгоритм определения по заданным координатам клетки (номерам строки и столбца), какое натуральное число туда записано (2 балла);

Б) Опишите алгоритм определения по заданному натуральному числу координаты клетки (номер строки и столбца), в которую это число записано (4 балла).

Нумерация строк и столбцов начинается с единицы.

8. Фрагменты программ (6 баллов)

Какие из представленных ниже фрагментов программ эквивалентны между собой? Что они делают?

1)
```
d:=a[1]>0
нц для i от 2 до n
  d:=d и (a[i]>0)
кц
Ответ:=d
```

2)
```
d:=a[1]<=0
нц для i от 2 до n
  d:=d или (a[i]<=0)
кц
Ответ:=не d
```

3)
```
d:=a[1]>0
нц для i от 2 до n
  d:=d или (a[i]>0)
кц
Ответ:=d
```

4)
```
d:=истина
нц для i от 1 до n
  если a[i]<=0
    то d:=ложь
  все
кц
Ответ:=d
```

5)
```
d:=ложь
нц для i от 1 до n
  если a[i]>0
    то d:=истина
  все
кц
Ответ:=d
```

Практический тур

9. Банк (30 баллов)

Сотрудникам многих российских банков разрешается заводить специальный счет на кредитных карточках, на котором можно хранить деньги в трех валютах — рублях, долларах или евро. При этом их привилегия заключается в том, что перевод любой суммы из одной валюты в другую происходит по курсам валют Центробанка России на данный момент без взимания процентов.

Центральный банк России в середине текущего дня обнародует курс валют на следующий день, что позволяет играть на колебаниях курсов валют. Составьте программу, которая по известным курсам валют за определенный период времени позволяет определить максимальную величину прибыли, которую можно получить за счет грамотного использования колебаний курсов двух данных валют.

Ограничения.

В начальный момент времени на счете хранится ровно 1000 рублей. Курсы доллара и евро даются в рублях с точностью до 4 знаков после запятой. Все расчеты производятся в рублях. Суммы всех операций даются с точностью до двух знаков после запятой (центов или копеек). Суммы округляются всегда в пользу банка. В конце периода вся сумма обязательно должна быть переведена в рубли.

Входные данные.

Во входном текстовом файле bank.in в первой строке содержится одно натуральное число — количество последующих строк (или банковских дней), а в каждой последующей строке даются два вещественных числа с четырьмя знаками после запятой, разделенных одним или несколькими пробелами. Первое означает курс доллара, а второе — курс евро на текущий банковский день. Количество строк файла (дней) не превосходит 250. Дни, соответствующие строкам, идут в хронологическом порядке.

Первый вариант

Выходные данные.

В результате работы программы в последней строке выходного текстового файла bank.out должна быть записана сумма счета в рублях по истечении указанного периода времени (с точностью до двух знаков после запятой). В первых строках должны быть записаны проведенные операции в следующем формате:

```
<Номер банковского дня><код операции><сумма
операции в рублях>
```

Возможные коды операций: 12 (покупка долларов), 13 (покупка евро), 21 (продажа долларов), 31 (продажа евро).

Операции должны идти в хронологическом порядке. Если в какой-то день операций было несколько, то они должны идти в том порядке, в котором выполнялись в течение дня. При этом номер дня повторяется в каждой строке.

Пример:

bank.in	bank.out
4 21.0000 30.0000 20.0000 29.8765 25.0000 32.0000 23.1234 31.5000	2 12 1000.00 3 21 1250.00 1250.00

Второй вариант (*оценивается не более 20 баллов*)

Выходные данные.

В результате работы программы на экран должно выводиться одно число — сумма счета в рублях по истечении указанного периода времени (с точностью до двух знаков после запятой).

Пример:

bank.in	Результат на экране
4 21.0000 30.0000 20.0000 29.8765 25.0000 32.0000 23.1234 31.5000	1250.00

10. Сумма (30 баллов)

Задана последовательность целых чисел $\{a_i\}$. Требуется определить, можно ли расставить знаки сложения и вычитания перед каждым из них так, чтобы сумма делилась на заданное целое число k.

Входные данные.

В первой строке файла `sum.in` находятся целые числа n (количество чисел в последовательности) и k. Затем в n строках записаны числа a_i, по одному числу в строке ($1 < k \leqslant 1000$, $0 \leqslant a_i < k$, $1 \leqslant n \leqslant 2000$).

Выходные данные.

В случае, если знаки + и − можно расставить требуемым образом, в первую строку файла `sum.out` нужно записать слово `YES`. В следующей строке следует записать ровно n знаков «+» и «−» (без пробелов) в искомом порядке.

В случае, если знаки нельзя расставить нужным образом, в единственной строке выходного файла должно быть записано слово `NO`.

Время работы программы: 5 с на тест.

Пример:

sum.in	sum.out
3 5 4 4 3	YES ++-

Пояснение к примеру. В этом примере можно расставить знаки «+» и «−» следующим образом:

+ 4 + 4 − 3 = 5 — делится на 5.

Областная олимпиада

11. Калькулятор (50 баллов)

Дано выражение вида:

<число><операция><число>=

где <операция> — это одна из операций +, −, *, / (деление понимается целочисленное, т. е. целая часть от деления), а <число> — любое натуральное число.

Требуется вычислить значение данного выражения при следующих ограничениях:

Подзадача 1. В выражении оба числа — от 1 до 9.

Подзадача 2. В выражении оба числа не превышают 150.

Подзадача 3. В выражении числа не превышают 10100 и допускаются только операции + и −.

Подзадача 4. В выражении числа не превышают 10100 и допускается только операция *.

Подзадача 5. В выражении первое число не превышает 10100, второе число не превышает 150 и допускается только операция /.

Подзадача 6. В выражении числа не превышают 10100 и допускается только операция /.

Подзадачи 1–2 оцениваются по 5 баллов, подзадачи 3–6 — по 10 баллов.

Входные данные.

Входной файл `calc.in` содержит одну строку вида `<число><операция><число>=` без пробелов.

Выходные данные.

Выходной файл `calc.out` должен содержать одно целое число — значение выражения, записанное без лишних ведущих нулей.

Ограничение по времени тестирования: 1 с на один тест.

Примеры:

	calc.in	calc.out
Подзадача 1	3+4=	7
Подзадача 1	9*9=	81
Подзадача 1	8/2=	4
Подзадача 2	17/3=	5
Подзадача 2	25-124=	-99
Подзадача 2	150*150=	22500
Подзадача 3	1234567891234567890-1=	1234567891234567889
Подзадача 4	10000000000*10000000000=	100000000000000000000
Подзадача 5	325/100=	3
Подзадача 6	1/127361836136137183=	0

12. Кладоискатель (50 баллов)

Некий лабиринт представляет собой матрицу $M \times N$, некоторые ячейки которой — пустые, а остальные заполнены камнями. В одной из пустых ячеек находится клад, а в другой — кладоискатель. Кладоискатель и клад не могут находиться в ячейках, заполненных камнями, за пределами лабиринта, а также одновременно в одной и той же ячейке. Кладоискатель может передвигаться в соседнюю ячейку (соседними считаются ячейки, граничащие по стороне), а также передвигать клад следующим образом: встать в соседнюю к кладу ячейку и толкнуть его. При этом клад передвинется на сосед-

нюю ячейку в направлении, заданном толчком, а кладоискатель переместится в ячейку, где только что находился клад.

Требуется написать программу, которая определяет последовательность толчков и передвижений кладоискателя, обеспечивающую перемещение клада к выходу (выход находится в заданной пустой ячейке, по которой, в частности, может передвигаться кладоискатель). Так как клад очень тяжелый, количество толчков должно быть минимальным.

При наличии нескольких решений программа должна выводить то из них, которое требует наименьшего количества перемещений.

Входные данные:

- первая строка входного файла `klad.in` содержит числа M и N ($1 \leqslant M, N \leqslant 30$);
- последующие M строк содержат описание лабиринта. Каждая строка состоит из N символов, описывающих ячейки лабиринта: заполненная камнями ячейка обозначается заглавной латинской буквой «X», пустая ячейка обозначается символом «.» (точка), начальная позиция кладоискателя — буквой «B», позиция клада — латинской буквой «K», выход — латинской буквой «E».

Выходные данные.

Если решения не существует, то файл `klad.out` должен содержать слово `NO`.

Иначе в первой строке выходного файла `klad.out` должно содержаться слово `YES`, а во второй строке — последовательность символов, определяющая действия кладоискателя. В частности, символы «`w`», «`e`», «`n`», «`s`» обозначают передвижения кладоискателя на запад, восток, север и юг, соответственно, а символы «`W`», «`E`», «`N`», «`S`» обозначают толчки кладоискателя в соответствующих направлениях.

Ограничение по времени тестирования: 5 с на один тест.

Пример:

klad.in	klad.out
3 3 ..B .K. EXX	YES sWnwS

Городская олимпиада

Теоретический тур

1. Выборы (7 баллов)

В Анчурии в президентских выборах участвуют два кандидата: Ю и Я. На голосовании они получают n и m бюллетеней, соответственно. Председатель ЦИК имеет возможность признать одновременно недействительными одинаковое количество бюллетеней, поданных за каждого из кандидатов. Кроме того, он может удвоить количество бюллетеней, поданных за одного из кандидатов. Эти операции он может повторять в любой последовательности сколько угодно раз.

А) Может ли председатель одновременно обнулить количество голосов, поданных за каждого кандидата? Если да, то приведите алгоритм (3 балла).

Б) Может ли он добиться заранее заданного результата, при котором за кандидата Ю будет подано k, а за Я — r голосов? Если да, то приведите алгоритм (4 балла).

2. Игра в монотонность (8 баллов)

Двое по очереди вписывают по одному числу в свободную клетку полоски 1×100 клеток. Игра заканчивается, когда заполнены все клетки. После этого ищется наиболее длин-

7. Алгоритм (4 балла)

Что делает данный алгоритм?

```
Алг УПР (целтаб A[1:10000])
     рез A
нач цел i,j
    нц для i от 1 до 10000
     A[i]:=0
    кц
    нц для i от 2 до 100
        нц для j от i+i до 10000 шаг i
            A[j]:=1
        кц
    кц
    j:=0
    нц для i от 2 до 10000
        j:=j+1-A[i]
    кц
кон
```

Практический тур

8. Путь через дорогу (30 баллов)

По пути на работу после сильного дождя математик дошел до автомобильной дороги, залитой водой почти полностью, и остановился на самом краю, раздумывая, как же ему переправиться на ту сторону дороги. Над водой на дороге остались только несколько круглых островков — крышек водопроводных люков. Математик ввел систему координат, в начале которой он находился; ось X он направил вдоль края дороги. Затем он записал координаты и радиусы кругов.

Напишите программу, определяющую, можно ли переправиться через дорогу посуху, если известны ширина дороги (ее стороны параллельны) и максимальная величина шага математика.

Входные данные:

- во входном текстовом файле `road.in` в первой строке через пробел указаны три числа: w, s, n — ширина дороги, максимальная величина шага математика и количество люков;
- в каждой из n следующих строк записано по три числа: x_i, y_i, r_i — координаты и радиус одного из кругов;
- w, s, x_i, y_i, r_i — числа типа `real`; $n \leqslant 50$.

Выходные данные.

В выходном текстовом файле road.out должно быть записано одно целое число — минимальное количество островков, через которое проходит путь на ту сторону дороги, или число −1, если такого пути нет.

Время работы программы: 3 с на тест.

Пример:

road.in	road.out
15 5 3 -3 6 2.3 0 10 1.5 7 8 0.5	2

9. Поиск вируса (20 баллов)

Задан текстовый файл doc.txt. Файл считается зараженным, если в нем найдется последовательность подряд идущих символов (сигнатура), характерная для конкретного вируса. Символы перевода строки при поиске игнорируются. Сигнатура вируса задана символьной строкой длиной не более 300 символов в файле vir.in.

Напишите программу, которая определяет, заражен ли исходный файл. В результате работы программы в выходном текстовом файле vir.out должно быть записано одно целое число — номер символа, с которого начинается сигнатура в исходном текстовом файле, если он заражен вирусом, и −1, если не заражен.

Время работы программы: 1 с на тест.

Пример.

Файл doc.txt:

```
1234567890123456789012345678901234567890123456789
0123456789012
ОднаХизХсамых серьезных проблем сегодняшнего дня в
кмптрн облст связана с ви
русами, которые сильно портят нам жизнь
```

Файл vir.in	Файл vir.out
вирус	134

Городская олимпиада

Теоретический тур

1. Тестирование микросхем (8 баллов)

Имеется N микросхем, из которых более половины исправны. В прибор-тестер можно одновременно вставлять две микросхемы. При этом на исправной микросхеме лампочка загорается, только если вместе с ней в тестере находится неисправная. Лампочка на неисправной микросхеме может загораться когда угодно, независимо от исправности второй микросхемы.

Опишите алгоритм отбора всех неисправных микросхем.

2. Набор гирь (10 баллов)

Имеется восемь не различимых по внешнему виду гирь весом 5, 6, 7, 8, 9, 10, 11 и 12 грамм. Как с помощью чашечных весов определить две самые тяжелые и две самые легкие гири за возможно меньшее количество взвешиваний?

3. Королевский расчет (8 баллов)

Сколькими различными кратчайшими путями шахматный король может пройти с поля b5 на поле h2?

4. Алгоритм (5 баллов)

Что делает следующий алгоритм?

1) *Вариант на Паскале*:

```
program X;
var i,j,n:integer;
     A:array[1..10] of real;
begin {ввод данных}
for j:=1 to n div 2 do
   for i:=1 to n-1 do
     begin
       A[i+1]:=A[i]+A[i+1];
       A[i]:=A[i+1]-A[i];
       A[i+1]:=A[i+1]-A[i]
     end;
end.
```

2) *Вариант на школьном алгоритмическом языке*:

```
алг X(цел n, вещтаб A)
  арг n, A
  рез A
нач цел i,j
  нц для j от 1 до div(n,2)
    нц для i от 1 до n-1
        A[i+1]:=A[i]+A[i+1];
        A[i]:=A[i+1]-A[i];
        A[i+1]:=A[i+1]-A[i]
    кц
  кц
кон
```

5. Множество отрезков (5 баллов)

Даны координаты $2n$ точек на плоскости. Опишите алгоритм построения n непересекающихся отрезков с концами в этих точках.

6. Копирование файлов (6 баллов)

При копировании файлов в системе Windows система сообщает ориентировочное время копирования оставшейся информации исходя из ее объема и скорости копирования. Однако при затруднениях с копированием система уменьшает скорость копирования, а затем после копирования проблемного участка может снова увеличивать скорость копирования. Время оставшегося копирования соответственно при этом скачет то вверх, то вниз. Программисту Васе надоело просто сидеть и ждать, и он стал записывать прогнозы компьютера каждые 5 секунд.

Опишите алгоритм, который по известным прогнозам времени копирования и общему объему скопированных файлов помогает Васе вычислить максимальную и минимальную скорость копирования, а также расположение части информации, которая труднее всего читается с диска.

7. Разнобуквенные слова (7 баллов)

Дан текст, состоящий из английских слов, разделенных пробелами, заканчивающийся точкой. В написании слов используются только строчные английские буквы. Составьте программу, которая выводит на экран все слова текста, не содержащие одинаковых букв.

Практический тур

8. Автоморфные числа (20 баллов)

Натуральное число n называется «автоморфным», если с ним совпадают последние цифры числа n^2. Например, 5 и 25 — автоморфные числа, так как $5^2 = 25$ и $25^2 = 625$.

Напишите программу, которая для заданного натурального $n \leqslant 100$ находит одно из n-значных автоморфных чисел.

Входные данные.

В первой и единственной строке входного текстового файла `auto.in` задается число n — количество цифр искомого числа.

Выходные данные.

В единственной строке выходного файла `auto.out` выводится искомое n-значное автоморфное число. Если же таковых не обнаружено, то в выходной файл выводится число -1.

Время работы программы: 5 с на тест.

Пример:

auto.in	auto.out
2	25

9. Короткий набор SMS (20 баллов)

В памяти мобильных телефонов имеется словарь записанных только строчными латинскими буквами слов. С клавиатуры телефона вводится некая последовательность цифр. Каждой цифре может соответствовать одна из букв группы, написанной на этой клавише телефона. Буквы на кнопках

расположены в алфавитном порядке. Найдите список слов из словаря, которым соответствует данная последовательность цифр.

Входные данные.

Словарь расположен во входном текстовом файле `dict.txt` по одному слову в строке без пробелов. Количество строк — не более 10 000. Распределение букв по кнопкам телефона задано в файле `letters.txt` в следующем формате: каждая строка файла содержит цифру и последовательность букв, отделенную от цифры пробелом. При этом строк — не более десяти, цифры в строках идут в порядке возрастания, а буквы — в алфавитном порядке. Во входном файле `sms.in` в единственной строке без пробелов записана последовательность цифр, длина которой не превосходит 20.

Выходные данные.

В первой строке выходного текстового файла `sms.out` должно быть записано количество возможных слов; далее следует их список в алфавитном порядке по одному в строке без пробелов.

Время работы программы: 5 с на тест.

Пример:

dict.txt	letters.txt	sms.in	sms.out
dear friends good luck	3 abcdefghijklmn opqr 7 stuvwxyz	3333	2 dear good

Городская олимпиада

Теоретический тур

1. Лифт (9 баллов)

Вы находитесь в кабине бесконечного в обе стороны лифта. В лифте имеются две кнопки. При нажатии на одну лифт всегда опускается на один этаж вниз, а при нажатии на другую — всегда поднимается на N этажей вверх. Вам неизвестно N и какая кнопка что делает. Вы также не можете определить, вверх или вниз пошел лифт. Перемещение происходит мгновенно. После остановки двери лифта открываются, и вы можете сделать любую пометку на доске, установленной на этаже, где вы оказались. Опишите алгоритм определения числа N за возможно меньшее число нажатий на кнопки.

2. Алгоритм (4 балла)

Ниже приведен текст алгоритма *X* на школьном алгоритмическом языке. Что делает этот алгоритм?

```
лит Алг X(цел К, вещ М, вещтаб А[1:1000])
нач
  если К=0
    то знач:="нет"
    иначе если А[К]=М
              то знач:="да"
              иначе знач:=X(К-1, М, А)
           все
  все
кон
```

3. Когда функции равны? (4 балла)

Почти во всех языках программирования есть функция A mod N, которая вычисляет остаток от деления числа A на N. Для заданных натуральных чисел A и B составьте алгоритм вычисления наибольшего возможного натурального N, при котором A mod N = B mod N.

4. Порядок в кавалерии (6 баллов)

В каждой клетке шахматной доски стоит 0. Разрешается выбрать любые две клетки, соединенные ходом коня, и увеличить на 1 стоящие в них числа. Можно ли добиться того, чтобы в клетках оказались записаны все натуральные числа от 1 до 64 (в произвольном порядке)?

5. Среднее число (6 баллов)

Числа от 1 до 55 разбили произвольным образом на одиннадцать пятерок и в каждой пятерке выбрали среднее число. Какое наибольшее и наименьшее значение может принимать среднее из выбранных чисел? Под средним числом из набора, содержащего N чисел, называется число X из этого набора, для которого половина оставшихся чисел из этого набора меньше X, а половина — больше X.

6. Прямоугольники на экране (6 баллов)

На белом экране (прямоугольном клетчатом поле) размерами $M \times N$ некоторые прямоугольные области, границы которых проходят по линиям сетки, окрашены в синий цвет так, что они не перекрываются и не касаются друг друга. Опишите алгоритм подсчета количества таких синих прямоугольников.

7. Цепочка домино (7 баллов)

Есть набор костяшек домино. Каждая костяшка задана парой натуральных чисел от 0 до N (в стандартном наборе домино $N = 6$). Некоторые костяшки из полного набора могут отсутствовать. Опишите алгоритм, определяющий, можно ли по правилам игры домино выстроить цепочку из имеющегося набора.

8. Электронное письмо (5 баллов)

Электронное письмо было послано из пункта $A1$ и последовательно проходило через пункты $A1$, $A2$, ..., An. Известно, что длина посланного письма оказалась не равной длине последнего полученного письма. Опишите алгоритм определения участка (хотя бы одного), где произошла ошибка при передаче. Разрешается звонить на любой пункт и узнавать длину полученного в нем письма.

Практический тур

9. Зарплата (15 баллов)

По данным о сотрудниках фирмы требуется определить среднюю зарплату низкооплачиваемых и среднюю зарплату высокооплачиваемых сотрудников.

Под средней понимается среднее арифметическое зарплат. Низкооплачиваемым сотрудником считается тот, у кого зарплата ниже средней зарплаты по фирме, а высокооплачиваемым тот, у кого зарплата выше средней зарплаты по фирме.

Если в какой-то из групп сотрудников нет, то их средняя зарплата считается равной средней по фирме.

Входные данные.

- в первой строке входного текстового файла `pay.in` задается число N — количество сотрудников фирмы ($N \leqslant 1000$);
- в N следующих строках содержится информация о каждом сотруднике фирмы: фамилия сотрудника и его зарплата в рублях, разделенные пробелом. Фамилия — это строка символов, зарплата — целое число.

Выходные данные.

В единственной строке выходного текстового файла `pay.out` должно быть два числа, разделенных пробелом: средняя зарплата низкооплачиваемых и средняя зарплата высокооплачиваемых сотрудников, округленные до копеек.

Время работы программы: 1 с на тест.

Пример:

pay.in	pay.out
3 Иванов 15000 Петров 16000 Сидоров 2000	2000.00 15500.00

10. Вычеркивание цифр (25 баллов)

Все натуральные числа от 1 до N ($N \leqslant 1000$) записали подряд слева направо: 123456789101112 ... N. Сколько существует способов вычеркнуть все цифры полученного числа, кроме четырех, чтобы оставшиеся цифры образовали (без перестановок) число 2006?

Например, для $N = 27$ из числа

123456789101112131415161718192021222324252627

можно получить 2006 только одним способом:

~~1~~2~~345678910~~1~~1121314151617181920~~21222324252~~6~~27~~

Входные данные.

Во входном текстовом файле `deletion.in` указано одно натуральное число — значение N.

Выходные данные.

Требуется вывести результат в текстовый файл `deletion.out`.

Время работы программы: 10 с на тест.

Примеры:

deletion.in	deletion.out
20	0
30	1
40	5

Областная олимпиада

11. Судоку (100 баллов)

Судоку играется на клетчатом поле 9×9, дополнительно разделенном на 9 малых квадратов 3×3.

Цель игры судоку — заполнить все клетки игрового поля цифрами от 1 до 9 так, чтобы в каждой строке и в каждом столбце, а также внутри каждого малого квадрата цифры не повторялись. В начальном положении некоторые клетки игрового поля уже заполнены цифрами.

Пример игры приведен на рис. 21.

Исходное расположение

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

Одно из конечных расположений

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

Рис. 21

Требуется написать программу, играющую в судоку.

Входные данные.

Файл `sudoku.in` содержит начальное положение игры в виде 9 строк по 9 цифр, где некоторые клетки игрового поля заполнены цифрами, а пустые клетки обозначены нулями.

Выходные данные.

Выходные данные должны быть представлены в текстовом файле `sudoku.out` в таком же виде, но без нулей. Если искомого заполнения не существует, то в выходной файл следует вывести число –1.

Время работы программы: 3 с на тест.

Пример:

sudoku.in	sudoku.out
007000603 005736000 306000271 004189306 000304007 823000109 108207900 059860030 700093010	987412653 215736894 346958271 574189326 691324587 823675149 138247965 459861732 762593418

12. Лифт (100 баллов)

В подъезде многоэтажного дома на каждом этаже расположены по две кнопки вызова лифта — вверх и вниз. Дом обслуживается лифтом неограниченной грузоподъемности. Движение лифта подчиняется следующим правилам:

- пассажир, зайдя в лифт, нажимает кнопку с номером этажа, куда он хочет попасть;
- на преодоление одного этажа у лифта, независимо от направления его движения, уходит 1 секунда, а на остановку — 3 секунды;
- если лифт проезжает мимо этажа, где есть необслуженный вызов в направлении его движения, то он останавливается и забирает пассажиров;
- если лифт не пустой, то он не меняет направления своего движения до момента выполнения заказов находящихся в лифте пассажиров;
- если лифт опустел, то он едет к ближайшему этажу, где есть необслуженный вызов, а если ближайших вызовов несколько, то он продолжает движение в том же направлении, что и до этого;
- в начальный момент лифт при наличии выбора (т. е. при нескольких ближайших вызовах) выбирает направление движения так, чтобы время обслуживания всех вызовов в соответствии с данными правилами было наименьшим.

По известному начальному положению лифта и известным вызовам на каждом этаже требуется определить, сколько времени лифт будет обслуживать имеющиеся вызовы. Во время обслуживания новые вызовы не поступают. Время стоянки лифта перед началом движения и после высадки последнего пассажира не учитывается.

Входные данные:

- в первой строке входного файла `lift.in` указано количество этажей в доме K ($1 \leqslant K \leqslant 1000$) и номер этажа, на котором находится лифт;
- в последующих K строках содержится информация о вызовах с каждого этажа. При этом первым числом в строке указано, сколько вызовов сделано с данного этажа. Это

может быть 0, если вызовов с данного этажа нет; 1, если вызов один и 2, если вызовов два. В двух последних случаях через пробелы указаны одно число или два числа — номера этажей, куда хотят попасть люди с данного этажа. Если заданы два числа, то одно из них всегда больше номера данного этажа, а другое — всегда меньше.

Выходные данные.

Суммарное время обслуживания всех вызовов должно быть выведено в файл `lift.out`.

Время работы программы: 1 с на тест.

Пример:

lift.in	lift.out
5 1 1 5 0 2 5 1 1 1 1 1	20

Городская олимпиада

Теоретический тур

1. Интересное число (6 баллов)

Четырехзначное число 1256 обладает следующим свойством: любая пара соседних цифр образует двузначное число, равное произведению двух однозначных. Найдите возможно большее число с неповторяющимися цифрами, не содержащее в своей записи цифры 0 и обладающее таким же свойством.

2. Алгоритм (4 балла)

Что делает следующий алгоритм?

```
Function X(n: integer): integer;
  begin if n<10
      then X:=n
      else X:=X(n div 10)+(n mod 10)
  end.
```

3. Ядерный чемоданчик (10 баллов)

На передней панели ядерного чемоданчика находятся 12 кнопок. Каждая кнопка управляет своим переключателем: ее нажатие переключает его из положения ВКЛ в положение ВЫКЛ и обратно. Начальное положение выключателей неизвестно. Ядерный чемоданчик подает сигнал тревоги на неслышимой (ультразвуковой) частоте, когда не менее восьми выключателей оказываются в положении ВКЛ. Опишите алгоритм, использующий возможно меньшее число нажатий на кнопки для того, чтобы чемоданчик гарантированно подал сигнал тревоги.

4. Определение победителя (8 баллов)

В ряд стоят N бойцов ($1 < N < 100$). Один тур битв проводится следующим образом: первая битва тура проводится между первым и вторым бойцом в ряду, вторая — между третьим и четвертым и т. д. Если у последнего бойца нет пары, то

он считается победителем и переходит в следующий этап. После тура в ряду остаются только победители, стоящие в прежнем порядке. Затем проводится второй тур, третий и т. д. пока не останется один человек — чемпион. Опишите алгоритм, который по результатам всех битв определит порядковый номер чемпиона в начальном строю.

Результаты задаются в виде строки, где подряд без всяких разделителей записаны результаты всех битв в виде знаков «+» (если выиграл боец с меньшим номером) или «−» (если выиграл боец с большим номером). Результаты пишутся в порядке прохождения битв. Например, если строка состоит из одних знаков «+», то победитель имеет номер 1, а если из одних знаков «−», то победитель имеет номер N.

5. Блоха на кубе (10 баллов)

Блоха находится в вершине куба. За один прыжок она может переместиться в любую соседнюю по ребру вершину. Сколькими различными способами она ровно за семь прыжков может оказаться в противоположной вершине?

6. ABBA (7 баллов)

В алфавите племени Мумбо-Юмбо имеется только две буквы — `A` и `B`, а в каждом слове равное количество обеих букв. Например, `ABBA` — слово длины 4 из словаря племени.

Опишите алгоритм, определяющий для данного слова длины $2k$ следующее за ним в лексикографическом порядке слово такой же длины или указывающий, что данное слово — последнее.

7. Q Ку (7 баллов)

Опишите алгоритм, определяющий, пересекается ли данный отрезок с заданной окружностью. Отрезок задан координатами концов, окружность задана координатами центра и радиусом.

Практический тур

8. Узкое шоссе (25 баллов)

На шоссе X–Y запрещены обгоны. Про каждую из N машин, которые будут ехать сегодня по шоссе в направлении от X к Y, известно время T_i начала ее движения, координаты то-

чек X_i и Y_i (в км), где она выезжает на шоссе и сворачивает с него, и ее собственная скорость V_i (в км/ч), с которой она едет, если ей никто не мешает спереди. Определите время, когда машина с номером 1 свернет с шоссе.

Входные данные:

- в первой строке входного текстового файла `road.in` задается число N — количество машин $(1 \leqslant N \leqslant 50)$ и число L — длина дороги в километрах $(1 \leqslant L \leqslant 1000)$;
- в N следующих строках содержится информация о движении каждой машины в следующем порядке: T — время ее появления на шоссе (в часах); X_i — координата точки появления машины на шоссе; Y_i — координата конечной точки маршрута данной машины и V_i — собственная скорость машины, разделенные пробелами $(0 \leqslant X_i < Y_i \leqslant L;$ $0 < V_i \leqslant 200)$.

Выходные данные.

В единственной строке выходного текстового файла `road.out` должно быть одно число — время окончания движения по шоссе машины с номером 1 в часах с точностью до 2 знаков после запятой.

Время работы программы: 1 с на тест.

Пример:

road.in	road.out
2 100 0 10 100 100 0.5 70 90 40	1.10

9. Юбиляр (20 баллов)

В организации каждый месяц чествуют самого старшего из юбиляров, т. е. самого старшего из сотрудников, которым в этом месяце исполняется круглая дата (20, 30 и т. д. лет).

Напишите программу, которая по заданному месяцу и году находит в файле с данными о сотрудниках самого старшего юбиляра (если таковой есть) и выдает его фамилию.

Входные данные.

Данные о сотрудниках находятся в файле `data.txt`. Информация о каждом сотруднике занимает одну строку, в которой сначала следует дата его рождения и фамилия в формате `ЧЧ.ММ.ГГГГ,фамилия`. Информация по сотрудникам никак не упорядочена. Количество сотрудников не превышает 10000.

В файле `anniversary.in` через запятую задаются месяц (два символа) и год (четыре символа).

Выходные данные.

В файл `anniversary.out` выводится слово `NO`, если для заданного месяца и года в организации не нашлось юбиляра, либо фамилия самого старшего юбиляра. Если самых старших юбиляров окажется несколько, то их фамилии выводятся в последующих строках файла.

Время работы программы: 3 с на тест.

Пример.

Файл `data.txt`:

```
15.03.1930,Алферов
01.10.1972,Петрова
11.12.1918,Солженицин
```

Файл `anniversary.in`:

```
11, 2008
```

Файл `anniversary.out`:

```
Солженицин
```

Городская олимпиада

Теоретический тур

1. Поиск числа (7 баллов)

Отсортированный по возрастанию массив из n различных чисел изменили следующим образом: отделили некоторое количество первых элементов и переставили их в конец, сохранив порядок. Опишите эффективный алгоритм определения, входит ли заданное число x в этот массив.

2. Покупка (6 баллов)

4. Алгоритм (4 балла)

Что делает следующий алгоритм?

1) *Вариант на Паскале*:

```
Function d(x:real; n:integer):real;
  begin
    if n=0
      then d:=1
      else
        if n<0
          then d:= 1/d(x,-n)
          else d:=d(x,n-1)*x
  end.
```

2) *Вариант на школьном алгоритмическом языке*:

```
алг вещ d(арг вещ x, арг цел n)
  нач
    если n=0
      то знач :=1
      иначе
        если n<0
          то знач :=1/d(x,-n)
          иначе знач := d(x,n-1)*x
        все
    все
  кон
```

5. Игральный автомат (7 баллов)

На игральном автомате расположено N закрытых ячеек с кнопками. В одной из ячеек находится приз. При нажатии на кнопку ячейка на короткое время открывается, и приз можно взять, если он там. После каждого неудачного нажатия приз незаметно перемещается в одну из соседних ячеек. Опишите алгоритм, который обеспечивает гарантированное получение приза, если:

А) за один ход можно одновременно нажимать на две кнопки (2 балла);

Б) за один ход можно нажимать только на одну кнопку (5 баллов).

6. Словарь (10 баллов)

Имеется словарь слов, состоящих из символов латинского алфавита. Длина каждого слова не превышает 100 символов. Придумайте способ представления информации, хранящейся в словаре, обеспечивающий быстрое определение, входит ли в словарь заданное слово. Предложенные решения будут оцениваться как с точки зрения скорости работы, так и с точки зрения объема используемой памяти.

7. Уравнение (8 баллов)

В этой задаче вы получите по одному баллу за каждое из пяти найденных решений уравнения в натуральных числах $x^2 + xy - y^2 = 1$ или восемь баллов за обоснованный алгоритм получения бесконечного ряда решений.

8. Инвестиции (9 баллов)

Имеется K предприятий и M денежных единиц, которые можно инвестировать в развитие предприятий. Известно, какую прибыль даст каждое предприятие, если в него вложить 0, 1, ..., M денежных единиц. Опишите алгоритм, который вычисляет максимально возможную суммарную прибыль и дает один из возможных вариантов распределения денег между предприятиями, при котором эта прибыль будет достигнута.

Например, для $K = 5$, $M = 5$ и следующей таблицы прибылей:

	Предпр. 1	**Предпр. 2**	**Предпр. 3**	**Предпр. 4**	**Предпр. 5**
0 ден. ед.	5	**4**	8	2	0
1 ден. ед.	**14**	11	**17**	0	12
2 ден. ед.	15	18	17	**19**	15
3 ден. ед.	20	22	21	19	**28**
4 ден. ед.	−5	18	−7	−1	0
5 ден. ед.	28	29	31	27	30

максимально возможная суммарная прибыль равна 82 и будет достигаться при следующем распределении денег (соответствующие значения прибыли выделены в таблице под-

черкиванием): 1-му предприятию дадим 1, 2-му — дадим 0, 3-му — дадим 1, 4-му — дадим 2 и 5-му — дадим 3 денежных единицы.

Практический тур

9. Покраска забора (20 баллов)

Однажды Том решил покрасить свой забор. Так как Том очень занятой человек, то он решил нанять рабочих, чтобы они выполнили работу за него. Для этого он подал объявление в газету и в ответ получил набор пар натуральных чисел (a_i, b_i), где a_i — номер первой, b_i — номер последней досок того участка, который готов покрасить i-й рабочий (i от 1 до M).

Требуется определить, может ли Том покрасить весь забор с помощью данных подрядчиков?

Входные данные:

- первая строка файла `fence.in` содержит число N — количество досок в заборе ($1 \leqslant N \leqslant 2\,000\,000\,000$);
- вторая строка файла содержит число M — количество заявок от рабочих ($1 \leqslant M \leqslant 10\,000$);
- следующие M строк содержат пары чисел через пробел: a_i, b_i — номера первой и последней досок участка забора, который готов покрасить i-й рабочий ($1 \leqslant a_i \leqslant b_i \leqslant N$).

Доски в заборе нумеруются, начиная с 1.

Выходные данные.

Единственная строка выходного файла `fence.out` должна содержать слово `YES`, если рабочие смогут покрасить забор, или `NO`, если не смогут.

Время работы программы: 1 с на тест.

Пример:

fence.in	fence.out
10 3 1 3 3 7 8 10	YES

10. Система распознавания автомобильных номеров (15 баллов)

На дорогах России появляются видеосистемы с автоматическим распознаванием номеров автомобилей. Каждый считанный видеокамерой номер может быть найден в базе данных. При считывании возможны ошибки (отклонение), при которых некоторые символы заменяются другими. Требуется написать программу, которая по считанному номеру, величине отклонения и заданной базе номеров выдает все автомобильные номера из базы, соответствующие данному номеру.

Входные данные.

База данных содержится в файле `bd.txt` по одному номеру в строке.

Каждый автомобильный номер имеет следующий формат:

```
<буква><цифра><цифра><цифра><буква><буква>
<цифра><цифра>
```

Буквы — строчные латинские. Цифры — от 0 до 9. И буквы, и цифры могут повторяться. Количество номеров в базе не превышает 10 000.

В первой строке файла `auto.in` находится считанный автомобильный номер. Во второй — число от 0 до 8 — величина отклонения, т. е. количество символов, которые считаны с ошибкой: 0, если номера совпадают; 1, если

3. Квадратики

Числа из множества {0, 1, 5, 7} обладают тем свойством, что их сумма равна 13, но никакие другие четыре числа, равные элементам этого множества, не дают такую же сумму. Таким образом, для решения задачи достаточно проверить сумму четырех чисел в клетках, образующих все квадратики 2×2 и подсчитать количество квадратиков, сумма в которых равна 13.

4. Последовательность

Прежде всего отметим, что все элементы множества получаются умножением единицы на какое-то количество сомножителей 2, 3 и 5. Следовательно, все элементы данного множества имеют вид $2^{\alpha}3^{\beta}5^{\gamma}$ и наоборот, все числа указанного вида принадлежат описанному в условии задачи множеству.

Один из вариантов дальнейшего решения задачи заключается в написании функции, которая по числу X определяет, является ли оно числом указанного вида, а затем — в последовательном переборе всех натуральных чисел, проверке условия, выводе на экран и подсчете количества найденных элементов последовательности. Функция проверки устроена достаточно просто — число делится на 2, пока это возможно, а затем — на 3 и на 5. Если после всех делений получилась единица, то результат проверки — «истина», а если получилось любое другое число, то значение функции — «ложь».

```
function check(x:longint):boolean;
begin
  while x mod 2 = 0 do x:=x div 2;
  while x mod 3 = 0 do x:=x div 3;
  while x mod 5 = 0 do x:=x div 5;
  check:=x=1
end;
```

Если про число n заранее известно, что оно может быть очень велико, то можно решать задачу другим способом. Для этого организуем линейный список или массив и в качестве первого элемента возьмем 1. Обрабатывая очередной элемент, мы умножаем его на 2, 3 и 5, а полученные при этом новые числа вставляем в список (с сохранением его упорядоченности) и переходим к следующему элементу. Так продолжается, пока мы не дойдем до n-го элемента. Возможны и другие варианты эффективного построения заданной последовательности.

5. Точка в треугольнике

Возможны следующие способы решения этой задачи.

Во-первых, рассмотрим три прямые, содержащие стороны треугольника. Если для каждой из них данная точка находится по ту же сторону, что и третья вершина, то точка M расположена внутри треугольника.

Во-вторых, если сумма площадей трех треугольников, образованных точкой M и каждой парой вершин треугольника, равна площади исходного треугольника, то заданная точка находится внутри треугольника.

Третий способ основан на том, что сумма углов при вершине M, образованной лучами, идущими в вершины треугольника, равна 360° только в том случае, если точка находится внутри треугольника.

Во всех случаях вычисления надо проводить в пределах некоторой заданной точности.

6. Соседние клетки

С первого взгляда кажется, что в задаче необходимо предварительно исследовать, сколько клеток соседствует с данной клеткой x. Оказывается это не так. Нужный фрагмент программы имеет вид:

```
Procedure kletki(x:integer);
begin
  if x+8<=64 then writeln(x+8);
  if x-8>=1 then writeln(x-8);
  if (x-1)/8<>int((x-1)/8) then writeln(x-1);
  if x/8<>int(x/8) then writeln(x+1);
end;
```

Областная олимпиада

Теоретический тур

7. Цикловоз

В этой задаче требовалось найти как можно более короткое решение; по этому критерию отдельно поощрялся победитель (самое короткое решение, полученное одним из участников, содержало 16 команд).

Прежде чем кодировать алгоритм на предложенном языке, построим общую стратегию перевода состава с пути $P1$ на путь $P2$. Если дважды перегонять вагоны на следующий путь по два, а потом перегнать оставшийся вагон, то на втором пути состав будет находиться в следующем порядке следования: 4, 5, 2, 3, 1. Таким образом, мы научились перегонять состав с одного пути на другой и при этом совершать некоторую перестановку вагонов. Легко заметить, что применяя эту перестановку пять раз, мы получим исходную перестановку, причем на втором пути (что нам и требуется).

Итак, одно из возможных решений состоит в пятикратном применении указанных действий.

№	Команда	Комментарий
1	влево (Л)	Эта группа команд перегоняет вагоны 4 и 5 на второй путь. При этом стрелка возвращается в исходное положение
2	прицепить вагон (+)	
3	прицепить вагон (+)	
4	вправо (П)	
5	перевести стрелку (С)	
6	влево (Л)	
7	отцепить вагон (–)	
8	отцепить вагон (–)	
9	вправо (П)	
10	перевести стрелку (С)	
11	выполнить команды с номера 1 по 10 (Ц 1 10)	Перегон вагонов 2 и 3
12	влево (Л)	Перегон вагона 1. Перевод стрелки не делаем, так как следующий перегон мы будем осуществлять со второго пути на первый
13	прицепить вагон (+)	
14	вправо (П)	
15	перевести стрелку (С)	
16	влево (Л)	
17	отцепить вагон (–)	
18	вправо (П)	
19	Ц 1 18	Второй перегон
20	Ц 1 19	Еще два перегона
21	Ц 1 18	Пятый перегон

Легко заметить, что многие группы команд здесь повторяются, а значит их количество можно уменьшить, используя команду «ц». В частности, первые две команды позже встречаются в решении, и их тоже можно заменить командой ц. Реализовав все подобные ухищрения, мы получим следующую программу, решающую задачу за 14 команд.

№	Команда	№	Команда
1	ц 7,8	8	+
2	ц 3,8	9	ц 5,7
3	ц 8,10	10	–
4	–	11	п
5	п	12	ц 1,11
6	с	13	ц 1,12
7	л	14	ц 1,11

Вопрос о том, можно ли решить задачу за меньшее число команд, предоставляем решить самим читателям.

8. Алгоритм

Данный алгоритм реализует перестановку букв в слове x в обратном порядке.

9. Период дроби

При решении этой задачи следует иметь в виду, что простое деление M на N не приведет к желаемому результату, так как период может превышать стандартное количество разрядов ЭВМ. Поэтому нужно организовать деление «в столбик». При этом следует учесть, что период дроби может начинаться и не сразу после запятой. Можно считать период выделенным, как только при делении на каком-то шаге встретится остаток 0 (тогда мы считаем, что и период равен 0) или повторится остаток, уже встречавшийся ранее.

При написании программы мы резервируем два массива длины N: один — для цифр дробной части M/N, а второй — для того, чтобы отмечать остатки, которые уже встречались.

Массив остатков мы сначала инициализируем нулями, а из данной в условии дроби выделяем целую часть. На каждом шаге деления «в столбик» мы умножаем предыдущий остаток на 10, делим полученное число на N, частное записываем в массив цифр, а в соответствующий элемент массива остатков (если там пока еще находится нуль) записываем номер шага цикла. Не позже, чем через N шагов у нас встретится остаток 0 или повторится один из встречавшихся ранее остатков. Зная номер K шага цикла, когда данный остаток встретился в первый раз, и номер последнего шага, мы выдаем в ответе цифры дробной части от K-й до предпоследней — это и будет искомый период дроби.

10. Печать по спирали

Для распечатки элементов таблицы в указанном порядке необходимо организовать последовательный просмотр всех элементов. Для этого заведем вспомогательную переменную k, хранящую номер очередного витка спирали. Пусть i, j — текущие номера строки и столбца очередного элемента таблицы. Изначально $k = 1$. Движение по таблице состоит из четырех направлений:

- вправо — j меняется от k до $n + 1 - k$;
- вниз — i меняется от k до $m + 1 - k$;
- влево — j меняется от $n - k$ до k шагом -1;
- вверх — i меняется от $m - k$ до $k + 1$ шагом -1.

При этом после каждого «витка» спирали необходимо k увеличить на 1.

Выполняем эти движения до тех пор, пока количество просмотренных элементов не станет равно $m \cdot n$.

11. Обход конем

В терминах теории графов данная задача является «*задачей поиска гамильтонова пути в графе*», т. е. такого пути, который проходит через каждую вершину ровно один раз. Пока, кроме полного перебора всех вариантов путей, не известно других алгоритмов решения этой задачи в общем случае. Поскольку такой перебор является очень трудоемким, то необходимо позаботиться о способах его возможного сокращения.

Наша задача может быть решена с помощью алгоритма, именуемого «*перебором с возвратом*», или «*backtracking*». Рекурсивная схема этого алгоритма представлена ниже.

```
алг backtrack(очередной ход);
нач
  если решение найдено
    то вывод ответа
    иначе
      пока есть возможные ходы
      нц выбор следующего хода из списка
         возможных ходов;
      backtrack(выбранный следующий ход)
    кц
  все
кон
```

В зависимости от типа задачи возможен поиск какого-нибудь одного решения, всех решений или оптимального в каком-то смысле решения. Для задач первого типа в рекурсивной процедуре предусматривают завершение работы при нахождении первого решения, а для задач третьего типа при переборе вариантов очередного хода стараются оптимизировать порядок выбора ходов, чтобы отсечь перебор заведомо невыгодных вариантов.

В нашей задаче первое решение найдется заметно быстрее, если сначала пробовать ходы, ведущие в клетки, имеющие наименьшее количество продолжений.

Для программной реализации необходимо завести рабочий массив из 64 элементов, где будут отмечаться ходы коня (при заполнении этого массива целиком программа завершает работу); полезно также завести массив размерами 64×8, где содержится список возможных ходов из каждой клетки, и дополнительный массив из 64 элементов, где в каждой клетке будет отмечаться количество возможных ходов из нее. В начале работы программы мы обнуляем рабочий массив и заполняем дополнительные массивы. При совершении очередного хода его необходимо фиксировать в рабочем массиве, а элементы вспомогательного массива, соответствующие клетке, в которую можно попасть за один ход из клетки очередного хода, надо уменьшить на 1. При возврате же из рекурсивно

вызванной процедуры соответствующие значения надо вернуть в исходное состояние. Порядок выбора варианта следующего хода при этом зависит от количества возможных после этого продолжений, т. е. мы в первую очередь будем пробо-

тов одночленов, полученных при произведении слагаемых со степенями $x^0 \cdot x^i$, $x^1 \cdot x^{i-1}$, $x^2 \cdot x^{i-2}$, ... , $x^i \cdot x^0$. Тогда формула для коэффициентов произведения двух многочленов имеет вид:

$$c_i = \sum_{k=0}^{i} a_k b_{i-k} \quad \text{для} \quad i = 0,\ 1,\ 2,\ \ldots,\ n + m,$$

где m и n — степени данных многочленов, а их коэффициенты задаются соответственно массивами a и b с нумерацией в порядке возрастания степеней от 0 до максимальной степени (остальные коэффициенты считаются равными нулю).

Практический тур

9. Кубическое уравнение

Известно, что у кубического уравнения всегда существует хотя бы один корень. Интервал, на котором его можно искать, например, может быть таким: $[-2M,\ 2M]$, где M — максимум модуля коэффициентов, деленных на a. Действительно, на концах этого интервала первый член уравнения по модулю превосходит сумму остальных членов и имеет противоположные знаки. Значит, левая часть уравнения имеет разные знаки, и из непрерывности функции следует, что на указанном интервале есть корень x_1. Найти его с любой степенью точности можно *методом дихотомии* (деления интервала пополам). По *теореме Безу* левая часть уравнения делится на $(x - x_1)$. Тогда уравнение можно представить в виде: $(x - x_1)(ax^2 + b_1x + c_1) = 0$. Новые коэффициенты легко вычислить. Для нахождения остальных корней требуется дополнительно решить квадратное уравнение.

Конечно, при вычислениях возможны погрешности, которые могут радикально повлиять на ответ (например, дискриминант квадратного уравнения может превратиться из положительного или нулевого в отрицательный и наоборот). Поэтому, получив корни, рекомендуется их проверить (к сожалению, тоже с какой-то погрешностью).

Возможны и другие способы решения описываемой задачи, — например, применяя дифференцирование, нетрудно

установить интервалы возрастания и убывания левой части уравнения, после чего исследовать каждый из этих интервалов на наличие корней. Далее остается найти каждый корень с нужной точностью описанным выше методом дихотомии.

10. Дешифровка

В задаче необходимо лишь аккуратно запрограммировать указанный в условии задачи алгоритм. Заметим, что программа будет очень простой, если заранее построить перекодировочную таблицу в виде кольца, т.е. обеспечить переход после буквы «я» к «а», приписав каждой букве ее новый код.

Областная олимпиада

Теоретический тур

11. Что получится?

Данный алгоритм выполняет поиск максимального элемента.

12. Путь по клеткам

Оптимальным способом решения данной задачи является так называемый *«волновой» алгоритм*. Он состоит из прямого и обратного хода.

Прямой ход. Помечаем первую клетку цифрой 1. Все возможные для следующего хода клетки помечаем цифрой 2 и таким же образом распространяем «волну» далее. Если путь существует, то «волна захлестнет» правый верхний угол. (Чтобы каждый раз не искать клетки, помеченные тем или иным числом, рекомендуется образовать *очередь* из вновь помечаемых клеток и пометки производить от текущей клетки согласно этой очереди.)

Обратный ход. Начинаем двигаться из правого верхнего угла. Пусть в эту клетку записано число K. Ищем клетку с номером $K - 1$, из которой можно попасть в текущую; затем — клетку с номером $K - 2$ и т. д. до клетки 1. Построенный таким образом путь имеет длину K. Чтобы каждый раз не искать клетку с нужным номером, рекомендуется во время прямого хода при пометке очередной клетки запоминать, от какой клетки мы сделали шаг к данной, — при обратном ходе достаточно двигаться в направлении пометок.

13. Конечность алгоритма

Докажем утверждение из условия задачи по индукции.

Пару соседних символов 01 назовем *блоком*. (Отметим, что допустимые преобразования могут применяться только к блокам.) Если в последовательности имеется только один блок, то каждое преобразование уменьшает на единицу количество нулей слева от «блоковой» единицы. Следовательно, через определенное количество шагов блок исчезнет, так как «бывшая блоковая единица» либо станет самой левой в последовательности, либо присоединится к единицам, с которых начинается последовательность. Таким образом, основание для индукции по количеству блоков имеется.

Если же количество блоков больше единицы, то, рассматривая часть последовательности справа от первого блока, можно отметить, что эта часть содержит на один блок меньше, чем вся последовательность. Поэтому, согласно предположению индукции, к этой части можно применить лишь конечное число преобразований. После этого либо коли-

нули, а целая часть равна нулю. Затем число печатается в виде, приведенном в примерах, т. е. сначала «0.» с предшествующим знаком «–», если число отрицательное; затем — все десятичные цифры исходного числа, начиная с первой ненулевой, кроме идущих подряд последних нулей, если они есть. Потом печатается буква «E» и вычисленный порядок числа (количество разрядов от первого ненулевого разряда числа до десятичной точки). Необходимо также учесть, что при отсутствии десятичной точки она подразумевается сразу после числа.

16. Вычисление выражения

Грамматика выражения совпадает с той, что была описана в задаче 88/89(4), — только вместо букв операндами здесь являются цифры.

Приведем простой алгоритм для вычисления выражения.

Назовем операции умножения и деления «операциями первого ранга», а сложение и вычитание — «операциями второго ранга».

1. Просматривая выражение слева направо, составим набор последовательностей операций первого ранга с примыкающими к ним операндами (будем называть такие последовательности «подвыражениями»).
2. Произведем вычисление полученных подвыражений, производя операции слева направо.
3. Полученные числа вставим в исходное выражение вместо извлеченных из него наборов («подвыражений»), в результате получится последовательность чисел, соединенных знаками операций второго ранга. (Заметим, что эти числа могут оказаться и неоднозначными.)
4. Вычислим полученное выражение, производя операции слева направо.

17. Упорядочение

Заметим, что для решения этой задачи годится любой алгоритм сортировки. Необходимо только научиться сравнивать два слова между собой. Это можно сделать, задав в программе список букв русского языка в алфавитном порядке, а сравнение двух слов производя посимвольно, сверяясь с этим списком.

Практический тур

18. Римская арифметика

Самый простой способ — перевести исходные числа в десятичную систему, выполнить требуемые действия, а затем выполнить обратный перевод.

19. Арбузы

Для решения этой задачи можно применять разные методы. Самыми известными среди них являются методы «*прогонки*» и «*стрельбы*». Кратко рассмотрим первый из них, а со вторым читатели могут ознакомиться самостоятельно, прочитав решение похожей задачи 00/01(5).

Обозначим веса арбузов как неизвестные x_i. По условию, $x_i = \frac{x_{i-1} + x_{i+1}}{2} - 0{,}1$ (в килограммах). Тогда начиная с x_3 можно выразить неизвестные x_i через предыдущие.

Заметим, что поскольку исходные уравнения системы являются линейными, то и выражение всех переменных через первые две тоже будет линейным. Значение первой переменной задано, поэтому все переменные мы можем выразить линейно через x_2 в виде $x_i = \alpha_i x_2 + \beta_i$. При этом первые коэффициенты равны, по условию: $\alpha_1 = 0$, $\beta_1 = a$, $\alpha_2 = 1$, $\beta_2 = 0$, а остальные можно получить последовательно при $i = 2, 3, \ldots n-1$ (собственно прогонка) с помощью рекуррентных соотношений:

$$\begin{cases} \alpha_{i+1} = 2\alpha_i - \alpha_{i-1}, \\ \beta_{i+1} = 2\beta_i - \beta_{i-1} + 0{,}2, \end{cases}$$

так как уравнения имеют вид:

$$x_{i+1} = 2x_i - x_{i-1} + 2{,}0$$

или

$$\alpha_{i+1}x_2 + \beta_{i+1} = 2(\alpha_i x_2 + \beta_i) - (\alpha_{i-1}x_2 + \beta_{i-1}) + 2{,}0.$$

Значение $x_n = b$, поэтому x_2 легко вычислить. Однако в этой задаче есть «подводный камень». Может оказаться так, что значение какого-то неизвестного окажется меньше нуля, что физически означает отрицательный вес арбуза. Эту ситуацию можно отследить, вычислив, кроме нужного веса x_2, также и значения всех остальных неизвестных.

20. Площадь многоугольника

Если многоугольник — выпуклый, то его можно разбить на треугольники диагоналями, исходящими из одной вершины, и сложить площади всех полученных треугольников, найдя последние, например, с помощью *формулы Герона*.

Опишем более эффективный способ вычисления площади многоугольника. Поскольку площади фигур не меняются при параллельных переносах, то для удобства сдвинем многоугольник вверх и вправо так, чтобы он целиком находился в первой четверти системы координат. Это не ограничивает общности задачи.

Проведем из каждой вершины многоугольника перпендикуляры на ось Ox. При этом получится n прямоугольных трапеций с основаниями Y_i и Y_{i+1} и высотами $(X_{i+1} - X_i)$. Для удобства будем считать, что $(n+1)$-я вершина совпадает с первой. Заметим, что при обходе многоугольника значения высот будут менять знак, поэтому при движении слева направо площади трапеций будут браться со знаком «+», а при движении справа налево — со знаком «−». В результате суммирования все площади вне нашего многоугольника сократятся и останется только внутренняя площадь (возможно, со знаком «−», что зависит от направления обхода). Формула же для нахождения площади примет вид:

$$S = \frac{1}{2}\sum_{i=1}^{n}(X_{i+1} - X_i)\cdot(Y_{i+1} + Y_i).$$

Проиллюстрируем данное утверждение на примере (рис. 22):

$$S_{ABCDE} = S_{H1ABH2} + S_{H2BCH4} + S_{H4CDH5} - S_{H5DEH3} - S_{H3EAH1}.$$

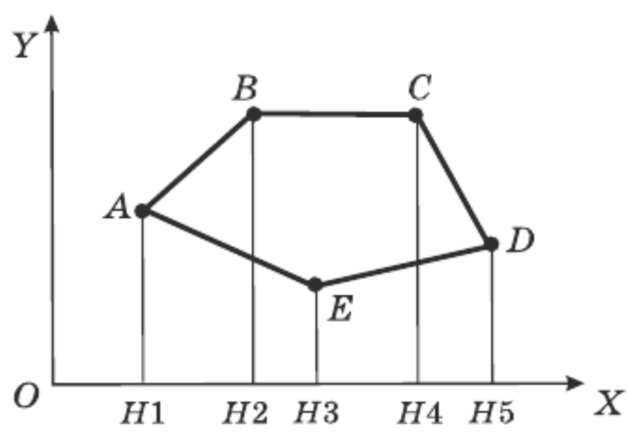


Рис. 22

Олимпиады 1989/90 учебного года

Городская олимпиада

Теоретический тур

1. Множество

Ключевое соображение для решения данной задачи вытекает из очевидной формулы: $(x+1)(y+1) = xy + x + y + 1$. Отсюда следует, что если в исходном множестве M увеличить элементы на 1 и получить множество $M1$, то описать структуру нового множества и правило его построения намного проще — его новые элементы получаются из двух уже найденных с помощью операции умножения. Действительно, если обозначить $\tilde{x} = x + 1$, $\tilde{y} = y + 1$, $\tilde{z} = z + 1$, где $z = xy + x + y$, то пара (x,y) порождает в множестве M элемент z, а соответствующая пара $(\tilde{x}, \tilde{y})$ порождает в множестве $M1$ элемент $\tilde{z} = \tilde{x}\tilde{y}$. Поэтому при известных двух первых элементах $a + 1$ и $b + 1$, порождающих множество $M1$, все остальные элементы нового множества представляются в виде $(a+1)^k(b+1)^l$, и наоборот, все числа данного вида являются элементами множества $M1$.

Приведенные выше соображения делают проверку вхождения числа z в исходное множество (или числа $z + 1$ в множество $M1$) достаточно простой. Надо делить число $z + 1$ на число $x + 1$, пока это возможно, а затем на число $y + 1$, тоже пока это возможно. Если в итоге получится единица, то ответ положительный, а если любое другое число — то отрицательный. Реализация аналогична функции в задаче 87/88(4). Кроме того, так же, как и в задаче 87/88(4), возможна реализация процесса построения множества по определению, хотя она заметно сложнее предыдущего решения.

2. Минимальное кодирование

Однозначность расшифровки может быть обеспечена тем, что ни один из кодов букв не является началом кода другой буквы.

Оптимальным является код, который получается следующим способом. Из двух наиболее редко встречающихся букв образуется одна новая буква с суммарной частотой их

появления. При этом код, который будет построен для новой буквы, с дописанной единицей присваивается одной из исходных букв, а с дописанным нулем — другой букве. Таким образом, задача сводится к задаче кодирования алфавита, содержащего на одну букву меньше. Эту задачу можно продолжать решать рекурсивно, пока в алфавите не останется одна буква. Ей присваивается пустой код (не содержащий ни одного символа).

Действуя указанным образом, мы получим следующий код, который называется *кодом Хаффмана*.

а – 1111	б – 011011	в – 10101	г – 11010	д – 00101
е,ё – 1101	ж – 1100011	з – 010111	и – 1110	й – 1011011
к – 01010	л – 11001	м – 01100	н – 111	о – 00
п – 101100	р – 10111	с – 0011	т – 0100	у – 01001
ф – 101101011	х – 1100010	ц – 00100011	ч – 01001	ш – 10110100
щ – 101101010	ъ,ь – 010110	ы – 110000	э – 00100010	ю – 0010000
я – 101000	Пробел – 100			

Логика построения кода понятна. Редко встречающиеся буквы в нем получают более длинные коды, а часто встречающиеся — более короткие. Это позволяет получать некоторую экономию. Скажем, для сообщения длиной 1000 символов, в котором буквы встречаются с указанной частотой, при использовании кодов, содержащих ровно по пять битов 0 или 1, объем занимаемой памяти оказывается равным 5000 бит, а при использовании кода Хаффмана — 4395 бит.

3. Двоичное умножение

Данную задачу можно решать несколькими способами.

Первый способ заключается в переводе заданных двоичных чисел в десятичную систему счисления. После этого над преобразованными числами выполняется умножение, результат переводится в двоичную систему счисления и цифры записываются в результирующую таблицу.

Второй способ моделирует умножение двоичных чисел «в столбик». При этом, как и при умножении десятичных чисел, надо организовать многократное поразрядное сложение с учетом переноса единиц в старший разряд, а также сдвиги второго сомножителя на количество нулей первого.

Нам же кажется наиболее предпочтительным третий способ, основанный на арифметике многочленов. В задаче 88/89(8) была описана процедура умножения двух многочленов. Согласно определению двоичного числа, его можно воспринимать как значение многочлена с коэффициентами, равными цифрам этого числа, и $x = 2$, так как:

$$(a_k a_{k-1} \ldots a_1 a_0)_2 = a_k 2^k + a_{k-1} 2^{k-1} + \ldots + a_1 2^1 + a_0 2^0.$$

Произведению чисел соответствует значение произведения соответствующих многочленов при $x = 2$. Формулы для вычисления коэффициентов произведения были приведены в решении задачи 88/89(8), надо только учесть, что цифры в результирующем числе могут принимать только значения 0 и 1. Поэтому процедура умножения модифицируется следующим образом: коэффициенты $c_i = \sum_{k=0}^{i} a_k b_{i-k}$ мы находим последовательно для $i = 0, 1, 2, \ldots, n + m$ по приведенным формулам, затем оставляем на i-м месте остаток от деления полученного числа на 2, а частное от его деления на 2 запоминаем «в уме» и добавляем к следующему разряду. Если в конце вычислений у нас «в уме» что-то осталось, то переводим это число в двоичную систему и записываем в старшие разряды результата. Преимущество данного способа состоит в том, что таким образом мы можем эффективно умножать сколь угодно большие числа, а программа при этом имеет достаточно простой вид.

4. Клетки в круге

Без ограничения общности можно считать, что центр окружности совпадает с началом координат. Тогда достаточно подсчитать количество клеток в первой четверти круга и затем умножить полученное значение на 4.

Пусть X, Y — координаты правого верхнего угла клетки. Данная клетка находится в рассматриваемой первой четверти круга тогда и только тогда, когда выполняются условия:

$$\begin{cases} X \geqslant 1; \\ Y \geqslant 1; \\ X^2 + Y^2 \leqslant R^2. \end{cases}$$

Простейшее решение представляет собой простой перебор, где X и Y «пробегают» все значения от 1 до $R - 1$. Некоторое сокращение времени счета можно получить, если проверять последнее неравенство не для каждой клетки. Действительно, если оно справедливо для некоторых X и Y, то оно справедливо и для всех меньших их значений. Поэтому достаточно для каждого X от 1 до $R - 1$ найти максимальное целое значение Y, удовлетворяющее неравенству (именно столько клеток в данном вертикальном ряду целиком лежат внутри четверти круга), и просуммировать все эти значения Y.

5. Робот

Задача предполагает составление алгоритма на школьном алгоритмическом языке.

Наиболее простой алгоритм заключается в обычном последовательном обходе поля роботом слева направо с одновременным исследованием состояния в соседних клетках (соседними считаются клетки, имеющие с данными общую сторону). Очевидно, что граничные клетки не со всех сторон имеют соседей, — это также надо учитывать.

6. Новобранцы

Приведем два решения задачи.

Первое решение основано на применении *принципа крайнего*. Предположим, что повороты никогда не закончатся. Рассмотрим крайнего левого солдата, который не закончит вертеться. Что его заставляет повернуться после очередного поворота налево? Только то, что он смотрит в лицо стоящего слева солдата. Но в этом случае и стоящий слева тоже должен совершить разворот. Значит, левее выбранного солдата

стоит солдат, который тоже никогда не закончит вертеться. Таким образом, самого левого солдата, который не закончит вертеться, просто не существует! Имеем противоречие.

Второе решение. Разворот двух солдат, смотрящих друг другу в лицо, эквивалентен перемене мест этих солдат. Конечно, реально солдаты никуда не переходят, но если считать солдат одинаковыми, то результаты двух указанных операций эквивалентны. Это означает, что каждый «разворот» приближает солдата, повернувшегося вначале налево, к левому флангу, а солдата, повернувшегося направо, — к правому флангу. Процесс завершится, когда все солдаты, повернувшиеся сначала налево, окажутся с одной стороны, а все остальные — с другой стороны шеренги. Заметим, что это второе решение позволяет дополнительно вычислить и общее количество разворотов, которые произойдут до окончания процесса. Это общее количество пар солдат, смотрящих после первого поворота друг на друга, независимо от того, стоят они рядом или нет.

7. Фрагмент программы

Алгоритм выполняет перестановку двух чисел, не используя дополнительной памяти.

Практический тур

8. Точка в тетраэдре

Подобно задаче о треугольнике и точке — 87/88(5), — данная задача ставит ту же проблему, только в трехмерном пространстве.

Уравнение плоскости, приведенное в указании к условию задачи, является ключом к ее решению. Подобно тому, как прямая делит плоскость, — плоскость, содержащая грань пирамиды, делит пространство на две части. Условием того, что заданная точка находится по ту же сторону от грани, что и вершина, является совпадение знаков чисел, которые получаются, если в уравнение плоскости вместо X, Y, Z подставить координаты четвертой вершины пирамиды и координаты заданной точки. Если это условие выполняется для каждой из четырех граней, то точка находится внутри пирамиды.

9. Телеграфный перевод

Если S_0 — сумма перевода при зарплате Z_0, то $S_1 = S_0 + 100$ — сумма перевода при зарплате $Z_1 = Z_0 + 102$ (за перевод S_1 нужно заплатить ровно на 2 коп. больше, чем за S_0, так как S_1 ровно на 1 руб. больше S_0; следовательно, $Z_1 = Z_0 + 100 + 2$). Обозначим через Z зарплату в копейках, а через S — сумму перевода, тогда искомый фрагмент программы будет выглядеть следующим образом:

```
A=int(Z/102);
S=A*100
C=Z-A*102
if C>2 then S=S+C-2
writeln('Сумма перевода =',S,'коп.')
```

10. Перестановка

Для решения этой задачи достаточно в качестве вспомогательной процедуры трижды использовать фрагмент программы из условия задачи 89/90(7).

Областная олимпиада

Теоретический тур

11. Мультипроцессорная система

Самая короткая программа, известная нам, состоит из шести тактов. Доказательство того, что пяти тактов недостаточно, мы оставляем читателю. Перед дальнейшим чтением советуем также вам самим попробовать уложиться в 6 тактов. Гарантируем удовольствие от головоломки!

Начальное распределение памяти по процессорам, начиная с первых ячеек, следующее:

$П1$: x; $П2$: x, a, b; $П3$: x, c; $П4$: x, d, e (рис. 23).

Последние ячейки сначала не используются.

Программы для каждого процессора:

	П1	*П2*	*П3*	*П4*
1	M,1,1,2	M,1,1,2	M,1,1,3	M,1,1,2
2	M,1,1,2	A,2,3,1	L,3,4	A,2,3,1
3	R,2,2	A,1,4,1	M,2,3,1	L,1,4
4	N	A,1,2,1	A,1,4,1	N
5	N	M,1,2,1	L,1,3	N
6	N	A,1,3,1	N	N

Проследим за содержимым памяти каждого процессора.

Начальное содержимое (рис. 23):

П1		*П2*		*П3*		*П4*
x	1	x	1	x	1	x
	2	a	2	c	2	d
	3	b	3		3	e
	4		4		4	

Рис. 23

После первого такта (рис. 24):

П1		*П2*		*П3*		*П4*
x	1	x	1	x	1	x
x^2	2	ax	2	c	2	dx
	3	b	3	x^2	3	e
	4		4		4	

Рис. 24

После второго такта (рис. 25):

П1		П2		П3		П4
x	1	$ax+b$	1	x	1	$dx+e$
x^3	2	ax	2	c	2	dx
	3	b	3	x^2	3	e
	4	x^2	4		4	

Рис. 25

После третьего такта (рис. 26):

П1		П2		П3		П4
x	1	x^2+ax+b	1	cx^2	1	$dx+e$
x^3	2	x^3	2	c	2	dx
	3	b	3	x^2	3	e
	4	x^2	4	$dx+e$	4	

Рис. 26

После четвертого такта (рис. 27):

П1		П2		П3		П4
x	1	x^3+x^2+ax+b	1	cx^2+dx+e	1	$dx+e$
x^3	2	x^3	2	c	2	dx
	3	b	3	x^2	3	e
	4	x^2	4	$dx+e$	4	

Рис. 27

После пятого такта (рис. 28):

$П1$		$П2$		$П3$		$П4$
x	1	$x^6+x^5+ax^4+bx^3$	1	cx^2+dx+e	1	$dx+e$
x^3	2	x^3	2	c	2	dx
	3	cx^2+dx+e	3	x^2	3	e
	4	x^2	4	$dx+e$	4	

Рис. 28

На шестом такте процессор $П2$ складывает содержимое первой и третьей ячеек и получает искомый результат.

12. Преобразование таблицы

Конечность алгоритма следует из того, что на каждом шаге общая сумма элементов (*полуинвариант*) становится на 1 больше. В то же время она ограничена сверху (хотя бы числом $7 \cdot M \cdot N$), поэтому процесс не может быть бесконечным.

Алгоритм заполнения клеток требует лишь аккуратной реализации, так как порядок сканирования таблицы и проверки соседних клеток уже полностью описан в условии. После изменения содержимого каких-либо двух клеток процедура начинается заново, а если изменений не произошло, то алгоритм заканчивает работу.

Результат работы алгоритма зависит от порядка перебора. Если изменить порядок просмотра клеток (по сравнению с описанным в условии), то результат может измениться. Достаточно рассмотреть три последовательные клетки с числами 4, 3, 2. Нетрудно заметить, что при просмотре слева направо они превратятся в 4, 4, 2, а при просмотре справа налево — сначала в 4, 3, 3, затем в 4, 4, 3 и окончательно — в 4, 4, 4. Остальные клетки исходной таблицы не изменят результат, если заполнить их нулями.

13. Торт

Очевидно, что такой разрез должен проходить через центр квадрата. Поэтому, если в центре стоит свеча, то такого разреза не существует. Если же свечи в центре нет, то следует рассмотреть прямую, соединяющую центр с каждой свечой. Если мы обнаружим такую прямую, для которой все остальные свечи находятся по одну сторону, то разрез выполнить можно (чуть сместив линию, чтобы она прошла сбоку от свечи).

14. Сложение с помощью логических операций

Сначала заметим, что у числа $c = (a \wedge \bar{b}) \vee (\bar{a} \wedge b)$ в каждом разряде стоит 1 тогда и только тогда, когда в соответствующих разрядах чисел a и b стоит ровно одна единица. Значит, при применении указанной операции к слагаемым в последнем разряде результата мы получим правильное значение суммы. Более того, в остальных разрядах это значение тоже было бы правильным, если бы при сложении из младших разрядов не было бы переносов. Но перенос в каждом разряде легко устанавливается операцией «$\wedge$». Так, переменная $r = a \wedge b$ хранит информацию о совпадении единиц слагаемых в каждом разряде. Эти единицы необходимо добавить к старшим разрядам суммы, т. е. перед сложением сдвинуть r на один разряд влево. После такого сдвига нам потребуется только сложить получившееся значение со значением переменной c. Обратим внимание, что ранее полученный правильный последний разряд при этом сложении не изменится. Ясно, что сложение остальных разрядов будет сопровождаться теми же проблемами, т. е. потребует аналогичных действий. Всего таких сложений для получения всех пяти правильных разрядов суммы необходимо проделать пять.

С учетом всего сказанного программа сложения двоичных чисел a и b будет следующей:

$c = (a \wedge \bar{b}) \vee (\bar{a} \wedge b)$,

$r = a \wedge b$,

$r = \leftarrow r$,

$c = (c \wedge \bar{r}) \vee (\bar{c} \wedge r)$,

$r = c \wedge r$,

$r = \leftarrow r$,

$c = (c \wedge \bar{r}) \vee (\bar{c} \wedge r)$.

Последние три строки надо повторить еще два раза. В итоге переменная c будет содержать значение искомой суммы.

15. Минимум функции

Решение данной задачи разбивается на два этапа. На первом из них (на этапе локализации минимума) мы должны найти какой-нибудь интервал, содержащий точку минимума, а на втором (на этапе уточнения) — сузить найденный интервал до необходимых размеров.

Локализация минимума. Сначала мы возьмем три произвольные точки $x_1 < x_2 < x_3$ и сравним значения функции в этих точках. При этом возможны случаи:

1) $F(x_1) < F(x_2) < F(x_3)$;
2) $F(x_1) > F(x_2) > F(x_3)$;
3) $F(x_1) > F(x_2) < F(x_3)$.

В случае (1) нужно искать точку минимума левее x_2, в случае (2) — правее x_2, а в случае (3) искомый интервал (x_1, x_3) найден. Определившись с направлением поиска, будем затем «шагать» в этом направлении или постоянными шагами, или увеличивая каждый следующий шаг, например, в два раза для ускорения поиска. Поиск следует вести, пока функция не начнет расти. На этом первый этап завершен.

Уточнение. Существует несколько алгоритмов уточнения точки минимума после того, как найден содержащий ее интервал. Самый простой способ — это *метод равномерного поиска*, когда мы двигаемся по интервалу от одного края с шагом E до тех пор, пока функция не начнет возрастать. Однако существует еще один, гораздо более быстрый способ — *метод поразрядного спуска*. Действуя по этому методу, мы сначала (аналогично методу равномерного поиска) определяем результат с точностью до единиц; затем меняем направление движения, уменьшаем шаг в 10 раз и определяем нужную точку с точностью до десятых; затем снова меняем направление движения и уменьшаем шаг в 10 раз и т. д.

Процесс заканчивается, когда ответ найден с нужным количеством знаков после запятой.

Кроме описанных, существуют и другие способы уточнения точки минимума — *дихотомия*, *метод золотого сечения* и *метод квадратичной интерполяции*, с которыми вы можете самостоятельно ознакомиться в литературе по численным методам.

16. Взаимодействие ЭВМ

Проанализируем действие ЭВМ2. Строки 10–30 ее программы предписывают постоянно читать содержимое регистра R, и если оно не меняется, то ничего не предпринимать. Если же в какой-то момент произошло изменение содержимого регистра R, то выдерживается длинная пауза (строки 40–60) и снова производится чтение регистра. Если за это время содержимое регистра не изменилось, то ЭВМ2 печатает «0», иначе — печатает «1» и вновь переходит на начало программы. Таким образом, при каждом изменении регистра (а это изменение может выполнить только ЭВМ1!) наша ЭВМ2, выдержав паузу, обязательно что-то напечатает. Причем, если содержимое регистра не будет изменено еще раз, то будет напечатан «0», а иначе — «1».

Посмотрим теперь, как ЭВМ1 изменяет содержимое регистра. При вводе какого-то значения с клавиатуры (строка 10) ЭВМ1 изменяет содержимое регистра на противоположное (строки 20–30) и выдерживает паузу. Если с клавиатуры была введена 1, то содержимое регистра меняется еще раз. На это повторное изменение регистра и отреагирует ЭВМ2, напечатав «1». Если же с клавиатуры был введен «0», то ЭВМ1 выдерживает дополнительную паузу, поэтому ЭВМ2 не успевает зафиксировать повторного изменения регистра, и печатает «0».

Итак, в любом случае, ЭВМ2 печатает то, что было введено с клавиатуры на ЭВМ1, но с задержкой.

17. Диагональ

Так как вершины многоугольника даны в произвольном порядке, то сразу невозможно определить, какие пары из них являются концами сторон, а какие — концами диагоналей.

Различить эти случаи легко, зная *определение выпуклого многоугольника*: все его вершины лежат по одну сторону от прямой, проходящей через любую сторону многоугольника.

Таким образом, для каждой пары вершин нам достаточно построить уравнение прямой $Ax + By + C = 0$, проходящей через эти вершины, а затем подставить в полученное уравнение координаты остальных точек. Если исходные точки являлись концами стороны многоугольника, то значения левой части уравнения будут при этом иметь одинаковые знаки. Все остальные пары являются концами диагоналей — из них и следует выбрать две наиболее удаленные точки.

18. Вершина прямоугольника

Для определения четвертой вершины прямоугольника нужно прежде всего найти, какая из трех заданных вершин является вершиной прямого угла. Для этого либо надо определить наибольший по длине отрезок (гипотенузу), соединяющий две из трех точек, либо проверить на равенство нулю скалярное произведенин векторов, выходящих из одной вершины в две другие. После этого координаты четвертой вершины легко вычислить или через координаты середины диагонали, или по правилу сложения векторов.

19. «Странное» свойство ЭВМ

Доказательство очевидно по *принципу Дирихле*. В шуточной форме этот принцип гласит, что «трех кроликов невозможно посадить в две одноместные клетки». Более серьезная его формулировка: если каждый из n предметов обладает одним из k свойств и $k < n$, то обязательно найдутся два предмета с одинаковыми свойствами.

В данной задаче ячейки ЭВМ можно считать предметами, а различное их содержимое — свойствами этих предметов. Легко подсчитать, что всего предметов (двухбайтовых ячеек) будет $128\,000/2 = 64\,000$, что меньше значения 2^{16}, а различных чисел, которые могут быть записаны в 16 разрядов, возможно 2^{16}.

Практический тур

20. Прямоугольники

Приведем алгоритмы решения для каждого пункта задачи.

А) Очевидно, что ответ будет положительным, если прямоугольники можно ровно приставить друг к другу. Это возможно, если у них есть одинаковые по длине стороны, что легко проверяется. Заметим, что размеры получившегося большого прямоугольника тоже легко вычислить, что может пригодиться при решении следующих пунктов задачи. Обратим также внимание, что в случае полного совпадения размеров исходных прямоугольников возможных больших прямоугольников может быть два.

Б) В этом случае один из прямоугольников должен быть приставлен к прямоугольнику, образованному двумя другими. Другого взаимного расположения прямоугольников существовать не может, в чем легко убедиться. Действительно, вершин у большого прямоугольника — четыре, а составляющих прямоугольников — три. Значит, две вершины должны быть вершинами одного из данных прямоугольников, т. е. тот должен быть приложен к прямоугольнику, составленному двумя другими. Таким образом, перебирая все пары исходных прямоугольников, мы применяем к ним алгоритм пункта (А), а затем в случае положительного результата применяем этот же алгоритм к получившемуся составному прямоугольнику и к третьему из исходных прямоугольников.

В) Рассуждения, аналогичные проведенным в предыдущем пункте, позволяют убедиться, что либо большой прямоугольник может быть составлен прикладыванием одного из исходных прямоугольников к прямоугольнику, ранее образованному тремя другими, либо каждая вершина большого прямоугольника принадлежит разным исходным прямоугольникам. В первом случае, перебирая все тройки прямоугольников, мы применяем к ним алгоритм пункта (Б), а затем — алгоритм пункта (А) к двум прямоугольникам. Второй случай, очевидно, возможен, только если прикладываются друг к другу два прямоугольника, каждый из которых ранее был составлен из двух исходных. Поэтому решение в этом случае сводится к троекратному применению алгоритма пункта (А) при различных способах разбиения исходных прямоугольников на пары.

21. Фрагменты последовательности

Достаточно простым является следующий алгоритм решения задачи. Считываем пары исходных данных в два массива: первое число пары — в первый массив, второе — во второй. Отыскиваем число, которое есть в первом массиве, но отсутствует во втором. Очевидно, это — первое число последовательности. Затем, уже зная пару (x_1,x_2), ищем пару (x_2,x_3) и т. д. Нетрудно видеть, что приведенный алгоритм имеет трудоемкость порядка n^2, так как для нахождения очередного числа требуется просмотр всех чисел обоих массивов.

Возможен и другой, более эффективный алгоритм. Заведем два массива: $P[1..32000]$ и $S[1..32000]$ и заполним их нулями. Считывая очередную пару исходных чисел (a,b), присваиваем $P[b] = a$, а $S[a] = b$. В результате значение i-го элемента массива P окажется равным элементу, предшествующему числу i в искомой последовательности, а значение i-го элемента массива S окажется равным элементу, следующему за числом i в искомой последовательности. Теперь нам остается выбрать элемент, у которого нет предшественника (т. е. соответствующее значение P осталось равным 0). Это — первый элемент последовательности x_1. Далее ясно, что $x_2 = S(x_1)$, $x_3 = S(x_2)$ и т. д. Такой алгоритм имеет линейную трудоемкость.

22. Шифр Кардано

Для решения задачи нужно составить процедуру поворота квадратной таблицы на 90°, переписывая первую строку в последний столбец, вторую — в предпоследний и т. д. (Конечно, при таком переписывании надо заботиться о сохранении информации, т. е. использовать для записи промежуточный рабочий массив.) Процесс расшифровки при этом состоит из последовательного выписывания символов текста, для которых соответствующие элементы перекодировочной сетки «вырезаны». Затем выполняется поворот сетки на 90°, снова выписывается текст и т. д. до полного поворота на 360°.

Олимпиады 1990/91 учебного года

Городская олимпиада

Теоретический тур

1. Записка Бормана

В условии задачи сказано, что если код буквы оказывается больше 33, то из него вычитается 34. Это значит, что число 34 считается равным нулю, 35 равно 1, и т. д. В математике в таком случае говорят, что мы имеем дело с *арифметикой вычетов*, или *сравнений по модулю* 34. В такой арифметике есть некоторые необычные свойства: например, два ненулевых сомножителя могут в произведении дать нуль (чтобы в этом убедиться, умножьте 17 на 2). Однако основные свойства, которые нужны для решения нашей задачи, сохраняются; в частности, из уравнения $a + x = b$ можно вывести, что $x = b - a$. Правда, нельзя забывать, что и любое другое x, отличающееся от найденного на 34 (или на число, кратное 34), тоже может быть решением. Чтобы подчеркнуть, что мы имеем дело не с обычными целыми числами, а с вычетами по модулю 34, после чисел или после арифметических выражений дописывают «(mod 34)», например: $35 = 1 (\mathrm{mod}\ 34)$. Подробнее о таких числах, а также о многих других числовых (и не числовых) множествах, в которых производятся обычные и необычные арифметические операции, можно прочитать в книгах по высшей алгебре. Нам же для решения задачи достаточно знать, что с этими числами можно производить все обычные операции и почти не обращать внимания на их некоторые «странности».

Вернемся к расшифровке записки Бормана. Местоимение «Я», которое должно быть в записке, может стоять либо в начале, либо в конце, либо в середине сообщения. Рассмотрим все эти случаи.

1) *«Я» стоит в начале сообщения.* Тогда второй символ должен быть пробелом. Значит, до прибавления к коду числа X начало записки выглядело так:

   33, 0, ...

Тогда после прибавления к этим числам «кодовой добавки» Бормана мы получим:

$33 + A + B = 15 \pmod{34}$;
$0 + 2A + B = 16 \pmod{34}$.

Отсюда $A = 0$, а $B = 16$. (Вообще-то, и A и B могут отличаться от этих значений на число, кратное 34, но в любом случае кодирование и декодирование сообщения от этого не меняются.) Расшифровывая записку дальше с найденными A и B, мы получаем, что ко всем первоначальным кодам букв было добавлено число $X = nA + B = 16$, и значит, до этой добавки записка выглядела так:

33, 0, 0, 0, 0, 22, 23, 26, 15, 9, 4, 20, 3, 2,

или, в буквенном виде, —

Я Ф Х Ш Н З Г Т В Б.

Это — очевидная бессмыслица.

2) *«Я» стоит в середине сообщения.* Тогда с обеих сторон от «Я» должны быть пробелы, и значит, в исходном сообщении должен был встретиться фрагмент, состоящий из пробела, буквы «Я» и еще одного пробела. После перевода в числовую форму это выглядело бы так:

... 0, 33, 0, ...

Если к коду буквы «Я» добавить 1, то 33 превратится в 34, что как мы уже отмечали, не отличается от нуля. Если же к этим подряд идущим трем нулям добавить $X = nA + B$, то мы получим:

$\ldots, nA + B, (n + 1)A + B, (n + 2)A + B, \ldots,$

что является тремя последовательными членами некоторой арифметической прогрессии.

Итак, если к коду буквы «Я» в записке Бормана добавить единицу, то три идущих подряд числа должны превратиться в арифметическую прогрессию. Легко проверить, что это свойство выполняется только при прибавлении единицы к седьмому числу. При этом шестое, седьмое и восьмое числа образуют последовательность:

4, 6, 8.

Это значит, что местоимение «Я» может быть только седьмым символом. А шестой и восьмой символы — пробелы. Тогда:

$6A + B = 4 \pmod{34}$;
$7A + B = 6 \pmod{34}$;
$8A + B = 8 \pmod{34}$.

Отсюда $A = 2$, $B = -8 = 26$ (это — еще одно странное свойство в арифметике вычетов по модулю 34). Вычитая теперь из кода каждого символа число $X = 2n - 8$ (или $X = 2n + 26$) и превращая числа в буквы, получим:

У Т Р О М Я У Л Е Т А Ю

Это и есть ответ в задаче.

3) *«Я» стоит в конце сообщения.* Этот случай полностью аналогичен случаю (1) и тоже не приводит к успеху. Убедитесь в этом самостоятельно.

2. Сортировка вагонов

Наиболее естественным методом решения этой задачи представляется алгоритмический. Это значит, что получив исходные данные, мы сразу попытаемся отсортировать вагоны. Если это удастся, то ответом на вопрос задачи будет «да», иначе — «нет».

Проще всего для сортировки вагонов использовать алгоритм, основывающийся на следующем правиле: поместить очередной вагон (номер которого равен K) на путь с минимально возможным номером, если последний вагон, стоящий на этом пути, имеет номер, меньший K, или путь свободен.

Данная задача оценивалась в 10 баллов. Предполагалось, что высшую оценку получит только тот участник, который сможет обосновать правильность алгоритма сортировки.

Если все вагоны состава будут таким образом расположены на станции, то очевидно, что вагон номер 1 на каком-то из путей окажется самым правым, и его можно подать на выход. Затем вагон номер 2 также может быть подан на выход, так как он не может стоять на пути за вагоном с номером, большим 2, и т. д.

Если же для какого-то вагона невозможно найти путь на станции, то это значит, что все пути уже «перекрыты» вагонами с бóльшими номерами, причем последние вагоны на этих путях расположены в порядке убывания, так как иначе какой-то вагон был бы помещен не на тот путь, на котором он сейчас находится, а на один из предыдущих. Вместе с последним вагоном, не нашедшим себе места, эти N вагонов составят подмножество, которое в принципе не может быть отсортировано на этой станции, так как каждый должен встать на свой отдельный путь, а вагонов больше, чем путей.

Отметим, что подобные залдачи имеют многочисленные приложения в области организации структур данных, а также в некоторых приложениях, связанных с проектированием печатных схем и др.

3. Вложенный цикл

Наборы параметров 1, 3, 4 и 6 определяют фрагменты программ, выполняющих одинаковые действия, также как и наборы параметров 2 и 5. Чтобы в этом убедиться, достаточно на координатной плоскости XY отметить точки, соответствующие слагаемым в строке 40, для всех шести вариантов значений параметров циклов.

Дополнительно необходимо заметить, что при замене x на y, а y на x сумма не изменится, — значит, 4-й набор параметров определяет ту же сумму, что и наборы 1, 3, 6, а 5-й — ту же сумму, что и 2-й.

4. Корень

Один из подходов к решению состоит в том, чтобы с помощью операции сложения реализовать умножение чисел, а затем перебирать (в разумных пределах) натуральные числа, возводить их в квадрат и сравнивать с данным числом n. Однако это решение не является оптимальным по количеству используемых операций. Более красивое решение задачи основано на тождестве: $1 + 3 + 5 + \ldots + (2k - 1) = k^2$. Оно реализовано в следующей программе:

```
procedure ROOT(n: integer; var i:integer);
var s:integer;
begin
  s:=0;
  i:=1;
  if s<n then
    begin
      s:=s+i;
      i:=i+1
    end;
    i:=i-1
end.
```

5. Четырехугольник и точка

В любом четырехугольнике есть диагональ, лежащая внутри этого четырехугольника. Ее концы легко найти, так как две другие вершины лежат с разных сторон от нее. Пусть 1-я и 3-я вершины исходного четырехугольника — концы этой диагонали. Легко заметить, что точка лежит внутри четырехугольника тогда и только тогда, когда она лежит в одном из двух треугольников — образованном 1-й, 2-й и 3-й вершинами и 1-й, 3-й и 4-й вершинами. А это уже легко проверить — см. задачу 87/88(5).

Возможны и другие решения. Найти их предлагаем самим читателям.

6. Фрагмент программы

Данный фрагмент программы проверяет делимость числа n на число b. Поскольку действие происходит с натуральными числами и при каждом шаге значение n уменьшается, то процесс конечен, и в результате получится одно из чисел от 0 до $b - 1$. Заметим, что обе используемые операции над n не меняют делимости (или неделимости) числа n на b.

Практический тур

7. Часы

Принимая во внимание, что движение стрелок является равномерным движением с угловой скоростью 6° в минуту

для минутной стрелки и 0,5° в минуту — для часовой, алгоритм решения задачи можно выбрать таким:

```
Procedure clocks(rr,mm:integer; var m:integer);
var rr,mm,k:integer;u,u1,m:real;
begin
  k:=rr*5;
   if k>mm then u:=(k+mm/12-mm)*6
           else u:=(mm-k-mm/12)*6;
  if u>180 then u:=u-180;
  if u>90 then u1:=u-90 else u1:=90-u;
  m:=round(u1*2/11);
end;
```

8. Игра

Очень часто при составлении стратегий выигрыша для игр с конечным числом состояний помогает построение *таблицы выигрышных состояний*. В такой таблице отмечаются состояния игры, для которых заранее известно, что дальнейший процесс игры приведет к проигрышу соперника.

Рассмотрим построение таблицы выигрышных состояний для случая (А); в случае (Б) рассуждения будут аналогичными.

Для случая (А) таблица выигрышных состояний имеет вид квадратной матрицы, имеющей 10 строк и 10 столбцов, пронумерованных от 0 до 9. Номер строки при этом соответствует первой цифре числа, а номер столбца — второй. Знаком «+» помечаем «выигрышные» клетки, соответствующие числам, назвав которые игрок обеспечивает проигрыш соперника, а знаком «−» помечаем проигрышные клетки.

Клетку (9,9) мы помечаем знаком «−», так как если игрок вынужден назвать число 99, то он проиграл. Клетки (8,9) и (9,8) мы тогда помечаем знаком «+», так как если игрок называет число 98 или 89, то его сопернику остается назвать только число 99, а значит, наш игрок выигрывает. Далее продолжаем пометку клеток, соблюдая следующие правила:

- все клетки, из которых есть ход в клетку, помеченную знаком «+», помечаем знаком «−»;
- клетки, из которых все ходы ведут в клетки, помеченные знаким «−», помечаем знаком «+».

В итоге мы построим следующую таблицу:

	0	1	2	3	4	5	6	7	8	9
0	–	+	–	–	–	–	–	–	–	–
1	+	–	–	–	+	–	–	–	–	–
2	–	–	+	–	–	–	+	–	–	–
3	–	–	–	+	–	–	–	+	–	–
4	–	+	–	–	–	+	–	–	–	+
5	+	–	–	–	+	–	–	–	+	–
6	–	–	+	–	–	–	+	–	–	–
7	–	–	–	+	–	–	–	+	–	–
8	–	+	–	–	–	+	–	–	–	+
9	+	–	–	–	+	–	–	–	+	–

Выигрышная стратегия будет заключаться в том, чтобы на каждом шаге называть число, которому соответствует клетка таблицы, помеченная знаком «+». Если же из начальной позиции нужного хода нет, то выигрышная стратегия есть у соперника. В этом случае можно дать только один совет — делать ходы, медленно ведущие к концу игры, — тянуть время. Вдруг соперник не владеет стратегией и ошибется!

В случае (Б) рассуждения, как уже говорилось, будут аналогичными, но таблица выигрышных состояний будет уже трехмерной размером $10 \times 10 \times 10$. Ее построение (вручную или с помощью компьютера ☺) мы оставляем читателям.

Областная олимпиада

Теоретический тур

9. Циклы

Два искомых случая довольно очевидны: когда цикл не исполняется ни разу, т. е. $(k + 6)/2 > (2k + 16)/2$ или $k < -10$, а также когда x принимает одни и те же значения, но в обратном порядке, т. е. когда k — четное число.

Казалось бы, что других вариантов быть не может, поскольку количество слагаемых одинаково, а при нечетном k в одной сумме каждое слагаемое на 0,5 меньше, чем соответствующее слагаемое в другой сумме. Однако существует еще один вариант, при котором суммы могут быть одинаковыми, поскольку слагаемые берутся по модулю. Ясно, что при этом в обеих суммах должны быть слагаемые разных знаков. Поэтому надо проверить нечетные $k > -10$, при которых $(k + 6)/2$ и $(2k + 16)/2$ имеют разные знаки, т. е. $k = -7$. Непосредственная проверка показывает, что данное значение k тоже подходит. Итак, окончательный ответ имеет вид: $k < -10$, где k — четные или $k = -7$.

10. Письмо

Из соображений симметрии следует, что если можно вложить открытку в конверт, то это можно сделать так, чтобы их центры совпадали.

Построим окружность с центром в точке пересечения диагоналей прямоугольника $a \times b$ и диаметром, равным диагонали открытки: $d^2 = x^2 + y^2$.

Тогда, если эта окружность целиком окажется внутри прямоугольника ($d < \min(a,b)$), то открытку вложить можно, причем как угодно. Если, наоборот, прямоугольник целиком окажется внутри окружности, то такое вложение осуществить нельзя, так как диагональ открытки никак не помещается в конверт.

Если же окружность и прямоугольник пересекаются, то возможны два варианта:

1) пересекаются две стороны;
2) пересекаются четыре стороны.

В варианте (1) открытку надо располагать параллельно сторонам конверта, и возможность вложения в этом варианте ($\min(a,b) < d < \max(a,b)$) определяется условием $\min(x,y) < \min(a,b)$.

В варианте (2) открытку надо располагать так, чтобы ее длинная сторона была почти параллельна диагонали прямоугольника. В этом варианте ($d > \max(a,b)$, но $d^2 < a^2 + b^2$)

критерием возможности вложения будет служить длина отрезка, соединяющего две точки пересечения окружности и прямоугольника, ближайшие к одной из его вершин. Если эта длина меньше $\min(x,y)$, то вложение возможно, а если больше, то нет.

11. Пересечение отрезков

Два отрезка пересекаются, если вершины второго отрезка лежат по разные стороны от прямой, содержащей первый отрезок и наоборот. Для проверки этого надо написать уравнение прямой, проходящей через концы каждого из отрезков, а затем подставить в уравнение координаты концов другого отрезка и сравнить знаки получившихся выражений.

12. Скобки

Закодируем нашу последовательность скобок числами от $-k$ до k, причем противоположные по знаку числа будут означать парные скобки одного и того же типа (соответственно, открывающую и закрывающую).

Один из вариантов решения задачи заключается в том, чтобы записать получившуюся последовательность в массив и проводить с ним, пока это возможно, следующую операцию: искать пару противоположных чисел, стоящих рядом, из которых первое число отрицательное (открывающая скобка), а второе — положительное, и удалять их, «сдвигая» массив. Если в результате останется пустой массив, то исходная последовательность была составлена правильно, а если останется непустой массив — то неверно. Трудоемкость такого алгоритма — порядка n^2.

Более эффективное решение основывается на использовании *стека*. Данные последовательно записываются в стек, а извлекаются из него в обратном порядке, т. е. по принципу «последним вошел, первым вышел». Место, куда помещается последний элемент и откуда он извлекается, называют *вершиной стека*. В программе стек обычно реализуется при помощи массива, в котором последний записанный элемент является вершиной стека. Тогда при удалении элемента из стека вершиной становится предыдущий элемент массива и т. д.

Теперь можно решить нашу задачу за один просмотр последовательности. Просматриваем исходные данные посимвольно. Если очередное число нашей последовательности — отрицательное (открывающая скобка), то мы помещаем его в стек. Если же число — положительное (закрывающая скобка), то мы проверяем, является ли последнее помещенное в стек число противоположным нашему по знаку. Если да, то мы удаляем его из стека и переходим к следующему элементу последовательности. Если же число в стеке тоже окажется положительным, то фиксируем ошибку.

Окончательным критерием правильности исходной последовательности будет пустота стека по окончании просмотра последовательности.

13. Игра «Даты»

Решение этой задачи очень напоминает решение задачи 90/91(8).

Построим таблицу 12×31, соответствующую дням года.

В варианте (А) пометим правую верхнюю клеточку (31 декабря) знаком «+» как «выигрышную». Тогда все клетки правого столбца (декабря) и верхней строки (31-е числа) будут проигрышными («−»). Из оставшихся клеток ближайшую свободную клетку (30 ноября) снова пометим «+» (как легко заметить, она тоже «выигрышная»), а оставшиеся клетки этого столбца и строки пометим знаком «−». Продолжение такого построения приведет к тому, что выигрышные клетки заполнят диагональ с 20.01 по 31.12. Соответственно, выигрышная стратегия будет заключаться в том, чтобы на каждом шаге называть одну из дат 20.01, 21.02, 22.03, 23.04, 24.05, 25.06, 26.07, 27.08, 28.09, 29.10, 30.11, 31.12. Выигрывает при этом первый игрок.

В варианте (Б) мы пометим правую верхнюю клеточку (31 декабря) знаком «−» (по условию она — «проигрышная»), а две ближайшие к ней клетки по горизонтали и вертикали (30.12 и 31.10, но не 31.11!) пометим знаком «+». Поскольку ход можно делать на любое количество клеток в одном направлении, надо пометить все оставшиеся клетки 10-го и 12-го столбца и 31-й и 30-й строк знаком «−». Из

оставшихся клеток ближайшую свободную клетку (29 ноября) снова пометим «+» (как легко видеть, она «выигрышная»), а оставшиеся клетки этого столбца и строки пометим знаком «−». Продолжение такого построения приведет к тому, что «выигрышные» клетки заполнят диагональ с 20.01 по 28.09. Соответственно выигрышная стратегия будет заключаться в том, чтобы на каждом шаге называть одну из дат 20.01, 21.02, 22.03, 23.04, 24.05, 25.06, 26.07, 27.08, 28.09, 31.10, 29.11, 30.12. Выигрывает при этом снова первый игрок.

14. Алгоритм

Легко увидеть, что вспомогательный алгоритм вычисления значения функции $M(x,y)$ является стандартным *алгоритмом Евклида* для нахождения наибольшего общего делителя, приведенным в школьном учебнике. Вторая часть основного алгоритма (цикл **пока**) реализует деление числа k на e. Тем самым в основном алгоритме ищется $ab/$НОД(a,b), т. е. наименьшее общее кратное — НОК(a,b).

15. Последовательность

Для решения пункта (А) задачи можно просто написать соответствующий цикл, который вычисляет цифры по данному правилу, например:

```
for i:=4 to 1991
 begin
    d:=a+b+c;
    d:=d-int(d/10)*10;  {или d:= d mod 10}
    a:=b; b:=c; c:=d;
 end;
```

Для решения пункта (Б) задачи такой алгоритм уже не годится, поскольку потребует выполнения слишком большого количества вычислений. Покажем, как можно обойти эту трудность. Будем рассматривать каждые три цифры нашей последовательности, идущие подряд, как одно трехзначное число. Поскольку различных трехзначных чисел существует

не более 1000, то среди первых 1001 чисел нашей последовательности обязательно найдутся хотя бы два одинаковых. Выберем два первых совпадающих числа и обозначим их номера как k и m ($k < m$). По правилам построения нашей последовательности легко заметить, что она — периодическая с периодом $m - k$, причем первые $k - 1$ чисел не попадают в период. Теперь нам осталось представить номер 1 000 000 000 в виде $k - 1 + (m - k) * n + r$ ($r < m - k$) и определить число с номером $k - 1 + r$, совпадающее с искомым числом.

В исходной задаче требовалось определить одну цифру, поэтому в полученном числе нам надо выделить последнюю цифру. Заметим также, что по условию построения последовательности мы можем, зная любые три цифры подряд, не только продолжить ее, но и восстановить все предыдущие элементы. Следовательно, при поиске периода можно существенно упростить проверку и просто сравнивать каждое очередное число с первым, так как обязательно будет выполнено условие $k = 1$.

16. Многочлен

Для вычисления значения многочлена в точке существует так называемая *схема Горнера*, которая подходит для решения первой задачи (пункт (А)). При вычислении по этой схеме переменной P (значению многочлена) сначала присваивают значение старшего коэффициента, а при вводе каждого нового коэффициента a значение P умножают на x и прибавляют a:

$$P = P \cdot x + a.$$

В результате после ввода всех коэффициентов величина P примет нужное значение.

Для вычисления значения производной многочлена в точке создадим аналогичную схему. Правда, она должна быть несколько сложнее, поскольку при увеличении степени многочлена на 1 не только увеличивается степень производной, но и сами коэффициенты при всех степенях x тоже увеличиваются на 1. Однако если совместить вычисление значения самого многочлена и значения его производной, то нужное

нам соотношение будет выглядеть очень компактно, а именно: новое значение Q (производной) получается из предыдущего умножением на x и добавлением старого значения многочлена, т. е. P:

$$Q = Q \cdot x + P.$$

Запишем все это в виде алгоритма.

1. Ввод точки x.
2. Ввод старшего коэффициента a.
3. Начальные присваивания: $P = a$; $Q = 0$.
4. **Пока** есть очередной коэффициент a:

```
| Q := Q * x + P
| P := P * x + a          | Порядок формул важен
```

Конец цикла

5. Вывод значения многочлена P и его производной Q.

Практический тур

17. Игра «7»

При решении данной задачи можно применить «лобовой» алгоритм — поставить на свое место сначала фишку 1, затем 2, 3 и т. д. Идейных сложностей на данном пути нет, нужна только техника программирования. Мы же рассмотрим другой способ решения, который требует меньшего числа ходов.

Как нетрудно видеть, всего различных расположений фишек в этой игре может быть $9!/2 = 181\,440$. Всю их структуру можно условно представить в виде *графа*. При этом из каждого состояния (*вершины*) существуют не более 6 различных передвижений, или ходов (*ребер*). Поэтому реально за разумное время построить для каждой вершины список смежных с ней вершин и хранить его в оперативной памяти. Для оптимизации этого этапа важен способ описания и нумерации различных состояний. Далее, для исходного состояния мы запускаем *алгоритм обхода полученного графа в ширину* и находим кратчайший путь до вершины, соответствующей конечному состоянию.

18. Имена

Данную задачу можно решать двумя способами.

Первый способ заключается в создании таблицы соответствия имен и их уменьшительной формы. Тогда решение сводится к поиску имени и выводу результата. При этом полнота таблицы имен зависит от эрудиции автора программы.

Второй способ связан непосредственно с правилами словообразования в русском языке.

В уменьшительно-ласкательных именах независимо от рода используется окончание «а» или «я». Общеупотребительными суффиксами являются «очк», «ечк», «еньк».

Суффикс «очк» используется, когда перед окончанием «а» стоят согласные «б», «в», «л», «м», «н», «п», «р», «с», или «т».

Суффикс «ечк» используется, когда окончание состоит из двух гласных («ая», «оя», «уя» — эти имена не имеют уменьшительных форм).

Суффикс «еньк» используется, когда либо окончание состоит из одной буквы «я», либо перед окончанием «а» стоит шипящая согласная.

Если перед окончанием «я» стоит согласная «л», то допускается равноправное использование суффиксов «ечк» и «еньк».

У этих правил есть небольшой недостаток — они относятся только к словам, являющимся именами собственными. Но для решения поставленной задачи их будет достаточно.

Олимпиады 1991/92 учебного года

Городская олимпиада

Теоретический тур

1. Бильярд

Ключевым при решении этой задачи является *соображение симметрии*. Поскольку при отскоке шарика от борта угол падения равен углу отражения, то можно симметрично отразить бильярдное поле относительно борта, о который ударяется шарик, и рассматривать движение шарика в «отра-

женном» бильярде. При этом траектория движения шарика будет просто прямой линией, проходящей *сквозь* борт. Поэтому для решения пункта (А) задачи достаточно отразить поле бильярда относительно двух дальних от лузы бортов и соединить точку начального положения шарика с образами лузы при этих отражениях. Вдоль этих прямых и следует наносить удар.

Для решения пунктов (Б) и (В) задачи нам потребуется многократно производить отражение бильярдного поля. Введем систему координат с началом в лузе и осями, направленными вдоль сторон бильярда. Тогда после всевозможных отражений вся плоскость разобьется на прямоугольники размером $a \times b$, а точки с координатами $(2ka, 2mb)$, где k и m — произвольные целые числа, будут являться образами лузы при этих отражениях. Для решения задачи остается соединить исходную точку с различными точками вида $(2ka, 2mb)$ и выбрать те из полученных прямых, которые пересекают ровно n прямых вида $y = 2mb$ или $x = 2ka$ (каждое пересечение с такой прямой соответствует одному отражению от борта). Так, при $n = 2$ надо соединить исходную точку с точками $(-2a, 0)$, $(0, -2b)$, $(2a, 2b)$. В общем случае перебор можно осуществлять в прямоугольнике от $-na$ до na по оси X и от $-nb$ до nb по оси Y. Можно и сократить перебор, устроив только один цикл (например по оси X), но на каждом шаге сначала смотреть, сколько вертикальных линий пересекается при данном X, а затем уже выбирать нужное значение по оси Y.

2. Роман

Обозначим через $B(m, j)$ толщину самого толстого тома при оптимальном разбиении первых m глав на j томов. Так, в частности, $B(m, 1) = \sum_{i=1}^{m} a_i$.

Чтобы найти $B(m, j)$, рассмотрим случаи, когда в последний том попадает одна последняя глава; две последние главы; три последние главы и т. д. Объем последнего тома при этом будет равен $\sum_{i=m-l+1}^{m} a_i$, где $l = 1, 2, 3, \ldots, m$.

Оставшиеся главы (их всего $m - 1$) должны быть разбиты на $j - 1$ том. При этом самый толстый том будет иметь объем $B(m - 1, j - 1)$. При таком разбиении самым толстым окажется либо последний том, либо один из предыдущих, т. е. его объем окажется равным $\max\left\{\sum_{i=m-l+1}^{m} a_i; B(m-1, j-1)\right\}$.

Нас интересует наилучший вариант, поэтому $B(m, j) = \min\left\{\max\left\{\sum_{i=m-l+1}^{m} a_i;\ B(m-1, j-1)\right\}\right\}$, где минимум считается по всем возможным l.

По указанным формулам можно найти сначала все $B(m,1)$, затем — $B(m,2)$ и т. д. Искомым же значением является $B(N,K)$.

Можно предложить и другой способ решения задачи. Предположим, что мы хотим издать роман так, чтобы самый толстый том не превышал S страниц. Тогда нам просто надо формировать тома по порядку, включая в каждый столько глав, сколько в него может поместиться, не превышая S. Если распределить главы таким образом удается, то можно попытаться уменьшить S. Если это не удается, то S необходимо увеличить. Задача же сводится к нахождению нижней границы S, при которой еще существует решение. Это можно сделать, например, *методом дихотомии* (деления пополам).

3. Польский калькулятор

Польская запись удобна тем, что все операции имеют одинаковый приоритет и вычисляются сразу, как только встретятся при просмотре заданного выражения. При этом для хранения исходных операндов и промежуточных результатов удобно использовать *стек*.

Просматривая строку слева направо, каждое число мы помещаем в стек. Если же встретится знак операции, то мы извлекаем из стека два верхних числа (из вершины стека — второй операнд, а следующий за ним — первый операнд), выполняем над ними данную операцию и результат помещаем обратно в стек.

Очевидно, после просмотра всей исходной строки в стеке останется одно-единственное число — результат вычисления выражения.

4. Список

Предполагается (и это видно из примера в условии задачи), что в списке фамилии не упорядочены по алфавиту (лексикографически). Значит, включение и исключение фамилий можно производить в произвольном месте списка, а алфавитный порядок поддерживать с помощью внутренних указателей.

Скажем, чтобы исключить из списка k-ю по алфавиту фамилию, ее надо найти, сделав k шагов по списку. Пусть эта фамилия оказалась в строке с порядковым номером m. Далее необходимо изменить два указателя, которые на эту фамилию указывают, — один находится в строке, содержащей предшествующую по алфавиту фамилию, а другой — в строке со следующей по алфавиту фамилией. Местоположение этих строк (p и n, соответственно) указано в строке m. Итак, в строке p надо указатель m заменить на n, а в строке n указатель m заменить на p. Если же удаляемая строка является первой или последней в списке, то соответствующую замену надо сделать только в одном месте. Заметим, что удаленная из списка фамилия продолжает занимать место в памяти, хотя становится недоступной для пользователя. На практике, чтобы такой «мусор» не засорял память, удаленные элементы списка либо учитывают в отдельном списке элементов, готовых к новому заполнению, либо как-то помечают (например, ставя «-1» в поле указателя предшествующего элемента) для той же цели.

Если же в список надо внести новую фамилию, то описанный процесс проделывается в обратном порядке: проходя по списку, находим место, куда можно записать новую фамилию; определяем номера строк, содержащих фамилии, которые по алфавиту должны непосредственно предшествовать новой фамилии и непосредственно следовать за ней; в этих, а также в строке с новой фамилией изменяем соответствующие указатели.

5. Билеты

Информационной моделью описанного в условии задачи зала может служить двумерный массив размером $k \times n$, где n — длина последнего ряда. Заметим, что $n = m + 2(k - 1)$. Нули

в таком массиве соответствуют свободным местам, а единицы — занятым. Программа поиска должна в двойном цикле — по рядам и по местам внутри ряда — искать три нуля, идущие подряд. Заметим, что ряды имеют разную длину, поэтому границы поиска в каждом ряду зависят от номера ряда (от 1 до $m + 2(i - 1)$, где i — номер ряда).

6. Алгоритм

Данный рекурсивный алгоритм предназначен для нахождения на отрезке от a до b точки x, где $\sin(x) = 0$. Для этого исходный отрезок разбивается на два меньших отрезка, из которых выбирается тот, на концах которого синус имеет разные знаки. Процесс продолжается, пока очередной отрезок не окажется достаточно маленьким. Описанный метод является одним из стандартных способов нахождения приближенного значения корня уравнения $f(x) = 0$. Заметим, что при отсутствии такой точки на заданном отрезке результатом будет его правый конец.

Таким образом, при $a = 1$ и $b = 2$ получается значение $x1 = 2$, а при $a = 3$ и $b = 4$ получается $x2 = 3{,}1415$, т. е. число π с точностью до 0,0001.

Практический тур

7. Умножение «столбиком»

Решение задачи предполагает аккуратную реализацию обычного алгоритма умножения «столбиком». Для красоты строки, соответствующие умножению на нулевую цифру, можно опускать.

Примечание: при решении этой задачи можно перевести заданные вещественные числа в строки и далее проводить посимвольный анализ этих строк, однако можно и просто выделять цифры поразрядно.

8. Кроссворд

Решение данной задачи предполагает аккуратную реализацию *переборного алгоритма*. При этом перебор можно осуществлять различными способами. Рассмотрим один из простейших.

Заметим, что два из четырех заданных слов должны располагаться по горизонтали, а два — по вертикали. Поэтому перебор надо начинать, выбирая какие-либо два слова в качестве горизонтальных, а другие — в качестве вертикальных. Кроме того, необходимо учитывать, какое из горизонтальных слов должно быть выше, а какое из вертикальных слов — левее другого. Нетрудно убедиться, что указанные различные расположения слов дают 24 варианта перебора.

Предположим, что задача решена и данные слова вписаны в некоторую сетку. Отметим, что эту сетку можно *транспонировать*, т. е. горизонтали сделать вертикалями, а вертикали — горизонталями, и при этом кроссворд останется составленным правильно. Таким образом, из всевозможных взаимных расположений слов достаточно перебирать только половину. Остается 12 возможных вариантов.

После того как слова выбраны, т. е. определено, какое из них первое и второе по горизонтали, а какое — первое и второе по вертикали, можно реализовать *алгоритм перебора с возвратом* (*backtracking*), поочередно накладывая слова друг на друга: сначала первое горизонтальное, затем — первое вертикальное, потом второе горизонтальное и, наконец, второе вертикальное.

Областная олимпиада

Теоретический тур

9. Порядок

Основная программа (использует в качестве вспомогательного алгоритм нахождения наибольшего общего делителя):

```
Алг ПОРЯДОК
нач цел i,m,n     | сумма, числитель, знаменатель
    нц для i от 2 до 100
      нц для m от 1 до i-1
          n:=i-m
        если НОД(m,n) = 1       | дробь несократима
          то вывод m,"/",n,", "
        все
      кц
    кц
кон
```

При вычислении НОД числителя и знаменателя используется *алгоритм Евклида* в рекурсивном виде.

```
Алг цел НОД(цел m,n)
нач
  выбор
    при m>n : знач:=НОД(m-n,n)
    при m<n : знач:=НОД(m,n-m)
    иначе знач:=m
  все
кон
```

10. Бензовоз

Если промежуточных пунктов между A и B нет, то оптимальная стратегия перевозки горючего состоит в том, чтобы загружать максимально возможное количество горючего в пункте A, перевозить его в пункт B, сливать там весь бензин (кроме необходимого на обратную дорогу), возвращаться в A и повторять все заново, пока горючее не закончится.

Промоделировав описанный процесс, легко получить ответ на пункт (А) задачи. Пусть описанный алгоритм называется ***БЕНЗОВОЗ***(d,V) и возвращает значение, равное ответу на вопрос задачи.

```
Алг вещ БЕНЗОВОЗ(вещ d,V)
нач вещ W | количество перевезенного бензина
    вещ U | количество оставшегося бензина
    вещ m
  если d>1000 или d>V |нельзя доехать
    то знач := 0
    иначе
      если V<1000 |сколько бензина можно влить
        то m:=V   |в бак в первый раз
        иначе m:=1000
    все
    U:=V-m |первая заправка
    W:=m-d |первая перевозка
```

```
    если W<d | не можем возвратиться
       то знач:=W
       иначе
         нц пока U>2*d |есть смысл возвратиться
          если U<1000  |сколько бензина можно влить
           то m:=U     |в бак в очередной раз
           иначе m:=1000
          все
          U:=U-m
          W:=W+m-2*d
         кц
      все
    все
  кон
```

Для решения задачи с промежуточным пунктом C можно дважды использовать описанный алгоритм: на участке от A до C и на участке от C до B. Заметим также, что после первого выполнения алгоритма бензовоз находится в пункте C. Тогда для первой части пути получаем $V1$ = ***БЕНЗОВОЗ***(x,V), а для оставшегося пути — $V2$ = ***БЕНЗОВОЗ***$(d-x,V1)$.

11. Слово

Отметим: если исходное слово содержит одну букву, то задача тривиальна. Будем поэтому предполагать, что букв — не менее двух. Тогда алгоритм построения следующего по алфавиту слова состоит из трех этапов.

На первом этапе просмотром исходного слова с конца ищется самая правая пара букв, в которой левая буква «младше» (по алфавиту), чем правая. Например, для слова «СУГРОБ» это пара «ГР». Если при этом поиске такая пара не найдется и будет достигнуто начало исходного слова, то следующего по алфавиту слова построить невозможно.

Левую букву в найденной паре (отметим ее) надо поменять местами с самой «младшей» из букв, более «старших», чем она, стоящих в слове правее. Для этого на втором этапе выполняется перебор букв в обратном направлении и сравнение их с отмеченной буквой, пока не найдется равная или

более «младшая» буква либо не кончится слово. В рассмотренном выше примере буква «Г» сравнивается сначала с «Р», затем с «О», а затем с «Б». После этого буква, стоящая перед последней найденной (т. е. «О») или в конце слова, меняется местами с отмеченной (т. е. с буквой «Г»). Получим слово «СУОРГБ». На третьем этапе остается упорядочить по алфавиту буквы, стоящие в слове после переставленной вперед на втором этапе (т. е. буквы «РГБ»). Полученное в результате слово и будет следующим по алфавиту после исходного. В приведенном примере это будет слово «СУОБГР».

12. Схема

Одна из возможных схем имеет следующий вид (рис. 29).

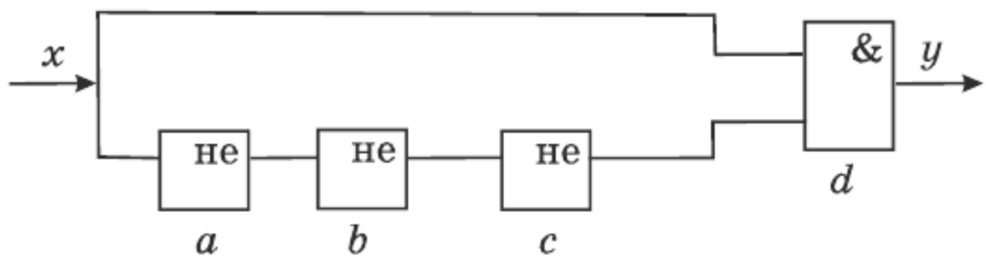


Рис. 29

В начале, когда сигнал x равен 0, на выходе элемента a мы имеем 1, на выходе b — 0, а на выходе c — опять 1. В итоге сигнал y равен 0, так как нулю равен верхний входной сигнал элемента d. Если теперь в некоторый момент времени изменить x с 0 на 1, то через секунду (время срабатывания d) сигнал y тоже станет равным 1, так как изменение по нижней ветке до элемента d дойдет с запаздыванием. Через 1 секунду после изменения x на выходе a будет сигнал, равный 0. Через 2 секунды на выходе b — 1, через 3 секунды на выходе c — 0, и, наконец, через 4 секунды y тоже станет нулевым. Таким образом, выходной сигнал y остается равным 1 ровно 3 секунды.

Аналогично работает и другая схема (рис. 30).

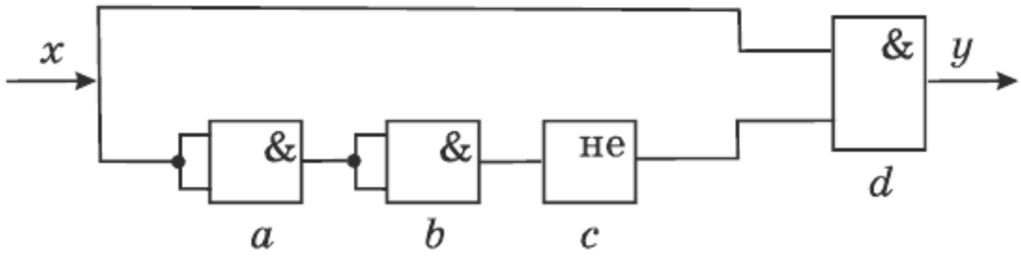


Рис. 30

Здесь, раздваивая один сигнал на входе, мы используем элементы a и b исключительно для получения временно́й задержки.

В первом и во втором решениях было использовано по четыре элемента. Однако, используя обратную связь, можно составить схему из трех элементов (рис. 31).

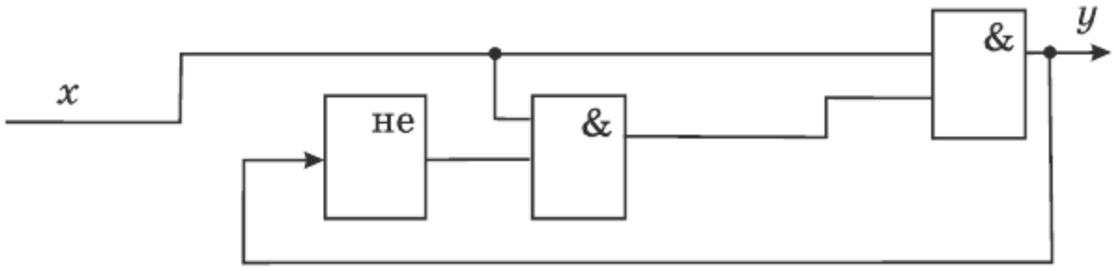


Рис. 31

Построив временно́й график, убедитесь самостоятельно, что сигнал y станет равным 1 через две секунды с момента изменения x с 0 на 1, а еще через 3 секунды снова станет равен нулю.

13. Польская запись

В задаче 91/92(3) уже рассматривался вопрос вычисления выражения, представленного в польской (или, если говорить более точно, в обратной польской, или постфиксной) записи. Теперь же нам требуется выполнить преобразование обычной скобочной (инфиксной) записи в польскую. Такое преобразование очень часто используется в трансляторах для языков программирования и даже в некоторых калькуляторах.

Приводимый ниже алгоритм использует для хранения операций и скобок вспомогательную память в виде стека. Операнды из входной строки сразу передаются в выходную.

Зададим по возрастанию приоритеты операций и скобок:

Операция	Приоритет
(	0
+ -	1
* /	2

Теперь опишем сам алгоритм.

1. Если входная строка пуста, то перейти к п. 6; иначе — прочитать следующий символ (цифру, скобку или операцию) и обозначить его через x.
2. Если x — цифра, то перенести ее в выходную строку и вернуться к п. 1.
3. Если x — открывающая скобка, то поместить ее в стек и вернуться к п. 1.
4. Если x — закрывающая скобка, то рассмотреть элемент, лежащий в вершине стека. Если он не является открывающей скобкой (т. е. это знак операции), то перенести его в выходную строку, поместив перед ним пробел, и повторить п. 4. Если теперь окажется, что стек пуст, то в исходном выражении был нарушен баланс скобок. Если в вершине стека находится открывающая скобка, то надо удалить ее из стека и вернуться к п. 1.
5. Остается случай, когда x — знак операции (другие ситуации — ошибочные). Если стек не пуст и приоритет элемента в вершине стека не меньше приоритета x, то перенести этот элемент из стека в выходную строку, поместить за ним пробел и повторить п. 5. Иначе — поместить x в стек, а в выходную строку поместить пробел (для разделения операндов) и вернуться к п. 1.
6. Освободить стек, удаляя каждый раз по одному элементу и помещая его в выходную строку с предшествующим пробелом.
7. Закончить работу.

Данный алгоритм можно описать и в более коротком виде, используя рекурсию. Эту задачу авторы оставляют читателям.

14. Алгоритм

Для обоснования конечности алгоритма достаточно найти какую-либо характеристику таблицы, которая бы монотонно изменялась (например, постоянно возрастала), но была ограничена (так называемый «*полуинвариант*»). В данном случае в качестве полуинварианта можно взять S — сумму всех чисел таблицы: ее значение не может превосходить сумму модулей всех элементов исходной таблицы. Так как на каждом шаге выбирается строка или столбец с суммой элементов меньше нуля, то после умножения на -1 их сумма, а вместе с ними и значение S, обязательно увеличивается. Всевозможных значений S — конечное число. Таким образом, через конечное число шагов не останется подходящих строк и столбцов. Следовательно, алгоритм конечен.

15. Спички

В задаче требуется определить, что искомое утверждение верно для любого начального распределения спичек по кучкам, или же что оно неверно для какого-то начального расположения. Если для любого начального положения условие окажется выполненным, то утверждение будет доказано. Если же с помощью ЭВМ удастся найти такое начальное положение, при котором условие не выполнится, то утверждение будет опровергнуто. (Невыполнение условия означает, что на каком-то шаге получится распределение спичек по кучкам, встречавшееся ранее и отличающееся от 1, 2, 3, ..., 9, т. е. что процесс зациклится.)

Итак, ЭВМ можно использовать для осуществления перебора всех возможных начальных распределений спичек по кучкам и моделирования процесса. Однако возможных начальных положений слишком много, и следует подумать о сокращении перебора.

Заметим, что если в течение нескольких (более 9) ходов на каждом шаге на столе окажется кучка с одной спичкой, то это значит, что ходом ранее на столе были кучки с одной и двумя спичками, два хода назад — с одной, двумя и тремя

спичками и т. д., а девять ходов назад спички удовлетворяли заданному условию. Это значит, что при переборе начальных положений спичек достаточно рассматривать только такие варианты, при которых нет кучек с одной спичкой. Отметим и еще один факт. Количество кучек, содержащих более 7 спичек, — не более пяти, поэтому через 7 ходов, кроме кучек, вновь образованных, останется не более пяти кучек. Таким образом, можно рассматривать расположение спичек не более чем в 12 кучках. Из этого дополнительно следует, что любая новая кучка не будет содержать более 12 спичек. Кроме того, совершенно безразличен порядок кучек на столе, поэтому перечислять начальные положения можно, учитывая кучки по убыванию (точнее, невозрастанию) количества спичек в них. Начать перечисление можно с положения, при котором в самой большой кучке лежит максимально возможное количество спичек. Тогда список начальных положений спичек будет выглядеть следующим образом:

12, 12, 12, 9
12, 12, 12, 7, 2
12, 12, 12, 6, 3
12, 12, 12, 5, 4
12, 12, 12, 5, 2, 2

Продолжить этот ряд нетрудно (конечно, не вручную, а с помощью ЭВМ).

Практический тур

16. Числительные

Решение задачи предполагает поразрядное выделение цифр заданного числа и их замену соответствующей словесной формулировкой.

17. Степень двойки

Прежде всего отметим, что запись числа n при $n > 3$ короче, чем запись числа 2^n. Поэтому в ходе поиска требуемых в задаче чисел при вычислении очередной степени двойки достаточно запоминать только последние n цифр. Так, при

поиске решений в диапазоне изменения n от 1 до 100 достаточно помнить две последних цифры степеней двойки, а при поиске решений в диапазоне до 1 000 000 достаточно помнить 6 последних цифр.

Приведенный ниже фрагмент программы осуществляет поиск чисел в диапазоне изменения n от 1 до 1 000 000:

```
n:=1; degree:=2;
while n<1000000 do
  begin
     if (n=right(degree)) or
        (n+1=right(degree)) then writeln(n);
     n:=n+1; degree:=degree*2 mod 1000000
  end
```

Примечание: для удобства именем `right(degree)` мы обозначили конец десятичной записи числа `degree`, имеющий столько же разрядов, сколько и число n.

Эта программа позволяет достаточно быстро найти две серии ответов:

1) n = 1, 7, 27, 727, 3727, 73727, 673727, ... (последние цифры соответствующих степеней двойки совпадают с числом $n + 1$);
2) n = 36, 736, 8736, 48736, 948736, ... (последние цифры соответствующих степеней двойки совпадают с числом n).

Если в первой серии ответов не обращать внимания на первую единицу, то нетрудно увидеть явную закономерность: каждое следующее число в этой серии оканчивается на предыдущее (и это при желании можно строго доказать). Данное соображение позволяет при получении последующего числа не перебирать все числа, а проверять только 10 «подозрительных», приписывая к началу последнего найденного числа все десятичные цифры по очереди. Для этого требуется процедура (или функция) быстрого возведения в степень, программа для которой приведена ниже. С ее помощью легко найти несколько следующих членов данных последовательностей.

```
function QuickDegree(n, mode:int64):int64;
{mode - степень 10, соответствующая количеству
 последних хранимых цифр}
begin if n=1
   then QuickDegree:=2
   else if n mod 2 =0
      then QuickDegree:=sqr(QuickDegree(n mod 2,
                          mode)) mod mode
      else QuickDegree:=(2*QuickDegree(n-1,
                          mode)) mod mode
end.
```

18. Цепочка

Сформулируем задачу в терминах *теории графов*.

Вершины графа — это двузначные числа (к однозначным спереди дописываем «0»). Вершины u и v соединены дугой, если из числа, соответствующего вершине u, можно получить число, соответствующее вершине v, указанным в условии задачи способом. Тогда задача сводится к отысканию кратчайшего пути в ориентированном графе, для нахождения которого можно использовать *волновой алгоритм* (или так называемый *обход графа в ширину*).

Олимпиады 1992/93 учебного года

Городская олимпиада

Теоретический тур

1. Лидеры

Следующие замечания помогут описать требуемый алгоритм.

1. Если до i-го спортсмена стартует спортсмен с большей скоростью, то i-й спортсмен никогда не станет лидером.
2. Стать лидером спортсмен сможет в момент, когда обгонит всех ранее стартовавших, и до этого момента никто, стартовавший позже, не станет лидером.

Для решения задачи заведем вспомогательную величину $V_{\max}$ и вспомогательный массив L, в котором на i-м месте будет записан нуль, если i-й спортсмен никогда не будет лидером, или время, когда этот спортсмен впервые станет лидером.

Алгоритм будет выглядеть так:

```
L[1]:=t1                    |первый становится лидером
Vmax:=v1
цикл для i от 2 до N
  если vi>Vmax
   то Vmax:=vi
      L[i]:=max{V[i]*T[i]-V[j]*T[j])/(V[i]-V[j]},
                            |где j меняется от 1 до i-1,-
                            |время, когда i-й обгонит
                            |всех предыдущих
   иначе L[i]:=0| i-й спортсмен не может стать лидером
  все
   цикл для j от 1 до i-1 |проверка, что стартовавшие
    если L[j]>L[i]        |ранее спортсмены могут
     то L[j]:=0           |остаться лидерами в момент
    все                   |времени L[j]
  кц
 кц
```

Остается выбрать спортсменов, для которых $0 < L[i] < t$.

Для ускорения работы можно отметить, что:

1) последовательность ненулевых $L[i]$ возрастает с ростом i;
2) при поиске максимума $L[i]$ можно проверять только тех спортсменов, у которых $L[j] > 0$, а не всех, кто ушел со старта раньше.

2. Циклы

Заведем одномерный массив L длины N и обнулим его. Будем двигаться по элементам массива в соответствии с продвижением по строкам, следуя командам GOTO исходной программы. Каждый элемент массива при этом будем помечать номером цикла, который проходится в данный момент. Очередной цикл заканчивается, как только в массиве встретится ненулевой элемент.

Начало нового цикла, если таковой есть, соответствует строке, которая еще не проходилась (что устанавливается по наличию нуля в соответствующем элементе массива L). Заметим, что на поиск таких элементов много времени не уйдет, так как каждый новый цикл начинается не ранее начала предыдущего цикла.

3. Фестиваль

Пронумеруем всех членов клуба числами от 1 до N. Наша задача заключается в перечислении всех различных наборов из k чисел (сочетаний из N по k).

Если бы число k было заранее известно, например, $k = 3$, то можно было бы просто написать k вложенных циклов, которые имели бы вид:

```
цикл для i1 от 1 до N-2
  цикл для i2 от i1+1 до N-1
    цикл для i3 от i2+1 до N
      вывод i1, i2, i3
    кц
  кц
кц
```

Однако k неизвестно, поэтому задачу следует решать другим способом.

Будем перечислять сочетания, в которых номера участников идут в возрастающем порядке. Первым будет набор (1, 2, ..., k). Каждое же следующее сочетание строится так. Мы просматриваем текущее сочетание справа налево и ищем самый правый элемент, не достигший своего максимального значения (для j-го места это $N - k + j$).

Если такого элемента не найдется, то значит, текущее сочетание — последнее и имеет вид ($N - k + 1$, ..., $N - 1$, N).

Если же элемент найден, то увеличиваем его значение на 1 и всем стоящим справа от него присваиваем новые значения по правилу: значение текущего элемента на 1 больше значения соседнего слева. Например, при $N = 6$, $k = 4$ первое сочетание — (1, 2, 3, 4), последнее — (3, 4, 5, 6), а из (1, 2, 5, 6) по алгоритму построится (1, 3, 4, 5).

Можно предложить и иной способ, основанный на так называемых «*индикаторных*» *множествах*. Договоримся кодировать каждый набор k чисел из первых N натуральных чисел N-разрядным двоичным числом, в котором 0 в i-м разряде будет означать, что число i отсутствует в данном наборе, а 1 — что оно присутствует. Тогда алгоритм, порождающий все наборы из k чисел среди первых N натуральных, будет заключаться в переборе всех N-разрядных двоичных чисел, проверке, что в их записи содержится ровно k единиц, и печати для каждого такого числа соответствующего набора натуральных чисел.

Существуют и другие способы перечисления подобных объектов. Их поиск читатели могут осуществить самостоятельно.

4. Программа

Из начальных присваиваний и вида команд присваиваний внутри цикла сразу следует, что величина b все время является некоторой целой степенью числа a. Покажем, что в результате выполнения алгоритма величина b (а вместе с ней — и y) примет значение n-й степени числа a. Для этого достаточно заметить, что произведение b на k-ю степень t является *инвариантом* цикла, т. е. не изменяется во время его работы. При этом вначале это произведение равно как раз n-й степени числа a, а в конце — числу b, так как после выполнения цикла $k = 0$. Следует также заметить, что цикл закончится за конечное количество шагов, поскольку k — целое положительное число и на каждом шаге оно уменьшается по крайней мере на 1.

Итак, мы обосновали, что $y = a^n$.

5. Сумма квадратов

Если находить ответ с помощью цикла, в котором суммируются все квадраты чисел от 1 до n, то потребуется выполнить $2n + 1$ действий. Более короткий способ вычислений основан на формуле, которую можно доказать по индукции:

$$1 + 4 + 9 + \ldots + n^2 = \frac{n \cdot (n+1) \cdot (2n+1)}{6}$$

и требует всего четырех арифметических операций, а именно:

$$a = n + 1; \quad b = a + n; \quad s = n \cdot a \cdot b.$$

6. Треугольник и точки

Выделим прямоугольник, в котором расположены все искомые точки. Для этого найдем минимальные и максимальные из координат вершин данного треугольника по каждой оси. Затем составим уравнения трех прямых, содержащих стороны треугольника. После этого для каждой из целочисленных точек выделенного прямоугольника проверим, лежит ли точка внутри треугольника. Для этого достаточно проверить условие: знак величины, полученной при подстановке координат этой точки в каждое из трех уравнений, совпадает со знаком величины, полученной при подстановке в него координат вершины треугольника, которая не лежит на этой прямой.

7. Робот

Для решения этой задачи можно применить классический *метод «покоординатного спуска»*. Из начального положения робота мы двигаемся сначала по одной из координат до тех пор, пока уровень радиации растет, а затем по второй координате до самого обломка.

Предположим для определенности, что робот находится в левом верхнем углу поля. Тогда алгоритм поиска будет выглядеть так:

1. Измерить уровень радиации $P1$.
2. Шаг вправо.
3. Измерить уровень радиации $P2$.
4. **Цикл пока** $P2 > P1$:
   | $P1=P2$;
   | шаг вправо;
   | измерить уровень радиации $P2$;
   Конец цикла.
5. Шаг влево.
6. Шаг вниз.
7. Измерить уровень радиации $P2$.
8. **Цикл пока** $P2 > P1$:
   | $P1=P2$;
   | шаг вниз;
   | измерить уровень радиации $P2$;
   Конец цикла.
9. Шаг вверх.
10. Убрать обломок реактора.

Отметим, что если поле ограничено, то в алгоритм надо также добавить проверку границ. На бесконечном же поле надо сначала определять направление движения робота по каждой координате, измерив уровень радиации в соседних клетках.

Практический тур

8. Минимум

Пункт (А) для остроугольного треугольника называется *задачей Ферма*.

Обозначим искомую точку (сумма расстояний от которой до вершин минимальна) за X. Докажем, что если в треугольнике все углы меньше $120°$, то точка X определяется тем, что из X все стороны треугольника видны под углом в $120°$. Если же в треугольнике один из углов не меньше $120°$, то точка X является вершиной тупого угла треугольника.

Лемма: в равнобедренном треугольнике точка X лежит на высоте к основанию треугольника (может быть, совпадает с вершиной).

Для доказательства этой леммы заметим, что если X, обеспечивающая минимум расстояний, не лежит на данной высоте треугольника, то можно построить точку X', полученную проецированием X на эту высоту параллельно основанию, и убедиться, что для нее сумма расстояний меньше: на рис. 32 прямая XX' параллельна основанию AC, а C_1 — симметричное отражение C относительно прямой XX'. Тогда $XC = XC_1$, $AX + XC > AC_1$ как сумма двух сторон и $BX > BX'$, так как гипотенуза длиннее катета.

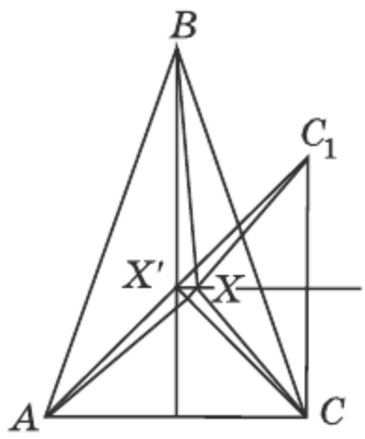


Рис. 32

Следствие: искомая точка X для правильного треугольника является центром, так как, согласно лемме, она должна лежать на всех трех высотах.

Далее, если данный угол треугольника меньше 120°, то искомая точка X не может совпадать с вершиной этого угла — достаточно сдвинуться немного по биссектрисе этого угла внутрь треугольника, и сумма расстояний уменьшится: на рис. 33 треугольник $A_1B_1C_1$ — правильный, CX' — биссектриса угла $\angle ACB$, X' — центр треугольника $A_1B_1C_1$.

Так как угол $\angle ACB$ меньше 120°, точка X' лежит внутри угла. Согласно следствию к лемме, $A_1X'+B_1X'+C_1X'< A_1C + B_1C + CC_1$, а значит, $A_1X' + B_1X' + CX' < A_1C + B_1C$. Тогда сумма $AA_1 + A_1X' + BB_1 + B_1X' + X'C < AC + BC$. Кроме того, $A_1X' < AA_1 + A_1X'$, $BX' < BB_1 + B_1X'$, и значит, сумма расстояний до точки X' меньше, чем до точки C.

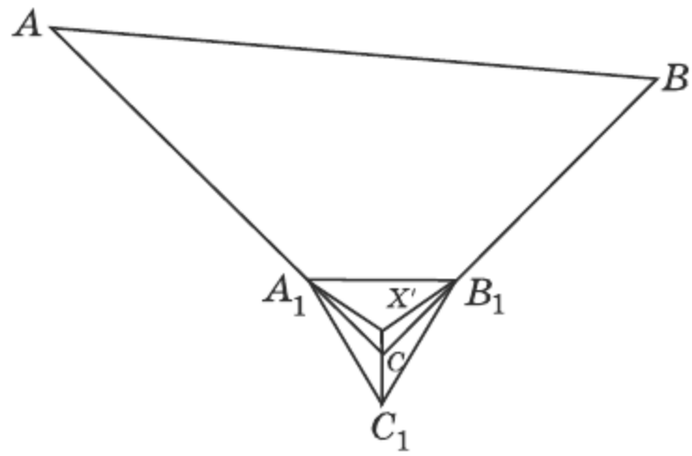


Рис. 33

Таким образом, если у треугольника все углы меньше 120°, то точка минимума X (которая всегда существует) лежит внутри треугольника. В этом случае углы $\angle AXB$, $\angle BXC$ и $\angle AXC$ равны 120°. Для доказательства достаточно на отрезках AX, BX и CX так выбрать точки A', B' и C', чтобы они образовали правильный треугольник. Точка X должна и в нем давать минимальную сумму расстояний до вершин A', B' и C', а значит, наше утверждение верно.

Построить точку X можно как пересечение дуг окружностей, из которых, например, стороны AB и BC видны под

углом 120° (тогда для третьей стороны AC условие на величину угла будет выполнено автоматически).

Для треугольника с углом больше 120° градусов обоснование утверждения похоже на уже рассмотренные приемы.

Для решения задачи о сумме квадратов расстояний отметим, что минимум достигается всегда в точке X, являющейся *центроидом* треугольника, т. е. точкой пересечения медиан, что легко увидеть, если записать выражение для суммы квадратов расстояний в общем виде в координатах.

Аналогичное утверждение для любого количества точек: минимум суммы квадратов расстояний есть центр тяжести системы исходных точек.

9. Путь коня

Сформулируем задачу в терминах *теории графов*.

Вершины графа — это клетки шахматной доски. Две вершины соединены ребром, если соответствующие им клетки могут быть соединены ходом коня. Тогда задача сводится к отысканию кратчайшего пути в графе, для нахождения которого можно использовать *волновой алгоритм* (или так называемый обход графа в ширину).

Областная олимпиада

Теоретический тур

10. Алгоритм

Данный алгоритм вычисляет числовое значение троичного числа, записанного в виде символьной строки, по схеме Горнера:

$$(\ldots((a[1] \cdot 3 + a[2]) \cdot 3 + a[3]) \cdot 3 + \ldots) \cdot 3 + a[n].$$

Аналогичный алгоритм часто используется в программах для получения значения десятичного числа, представленного в виде символьной строки; для этого надо умножать цифры не на три, а на десять, а в команду выбора добавить анализ недостающих цифр.

11. Таблица

Очевидно, что за два хода любой элемент таблицы можно поставить на свое место, на котором он должен находиться в упорядоченной таблице. Действительно, сначала горизонтальным ходом мы ставим его в свой столбец, а затем вертикальным ходом — в свою строку. Последовательно применяя эту процедуру, мы расставим все элементы таблицы по своим местам за $2 \cdot N \cdot M$ ходов (или чуть меньше, учитывая, что в оставшейся последней строке можно все элементы отсортировать за один ход).

Значительно лучшее решение получается, если за первый горизонтальный ход в первый столбец попадает число 1 (независимо от того, где оно находилось сначала), во второй столбец попадает число 2, в третий — 3 и т. д., а в столбец M — число M. Очевидно, что такой ход можно сделать. Вторым, вертикальным, ходом все эти числа перемещаются в первую строку. Таким образом, за первые два хода первая строка оказывается упорядоченной. Далее таким же способом мы сортируем вторую строку и т. д. Всего нам понадобится для упорядочения всей таблицы $2 \cdot N$ ходов.

Однако в данной задаче существует решение, не зависящее от N и M, и именно за него участники могли получить максимальное количество баллов. Это решение опирается на известную в математике *теорему Холла*, перед изложением которой мы введем несколько терминов.

Пусть имеется n множеств $S_1, S_2, \ldots, S_n$, составленных из набора элементов $A = \{a_1, a_2, \ldots, a_m\}$. *Системой различных представителей* этих множеств называется совокупность элементов $E = \{e_1, e_2, \ldots, e_n\}$, где все e_i, принадлежащие A, различны и e_i принадлежит множеству S_i (*представляет* множество S_i). Если же не каждое множество S_i представлено в E, то E называется *неполной системой различных представителей*.

Теорема Холла. Система различных представителей данных множеств существует тогда и только тогда, когда любые k множеств ($1 \leqslant k \leqslant n$) в совокупности содержат не менее k различных элементов.

Следствие. Пусть имеется n множеств, состоящих каждое ровно из r элементов, и каждый из элементов входит ровно в r множеств. Тогда для данных множеств существует система различных представителей.

Теорема Холла и ее следствие утверждают существование системы различных представителей, но не дают способа нахождения этой системы. Построение же системы различных представителей (в случае ее существования) дает следующий алгоритм, изложение которого можно проиллюстрировать на рис. 34. Изобразим исходные множества в виде точек в колонке слева, а элементы множества A — в колонке справа и соединим отрезками каждое множество с элементами, его составляющими. Например, на рис. 34 изображена система множеств $S_1 = \{a_1, a_2\}$, $S_2 = \{a_2, a_3, a_4\}$, $S_3 = \{a_1, a_5\}$, $S_4 = \{a_3, a_5\}$, $S_5 = \{a_4\}$.

L	R
$\{S_1\}$	$\{a_1, a_2\}$
$\{S_1, S_2, S_3\}$	$\{a_1, a_2, a_3, a_4\}$
$\{S_1, S_2, S_3, S_4, S_5\}$	$\{a_1, a_2, a_3, a_4, a_5\}$

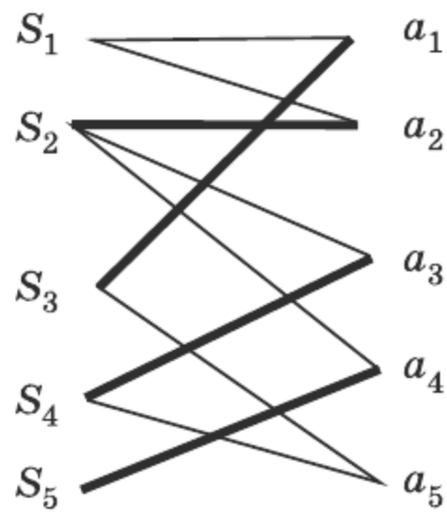


Рис. 34

Алгоритм.

1. Выберем какую-либо систему различных представителей множеств S_i в качестве исходной, — например, $\{a_1, a_2, a_3, a_4\}$, элементы которой представляют множества $\{S_1, S_2, S_3, S_4, S_5\}$, соответственно. На рис. 34 более жирной линией выделены те отрезки, которые соединяют множества с их представителями.
2. Если система полная, то закончим работу. Иначе — п. 3.

3. Выберем множество S_k, оставшееся без представителя, и включим его в качестве элемента (пока единственного) в некоторое множество L. В нашем примере на первом шаге в это множество войдет единственный элемент S_1. Из всех элементов, составляющих S_k, образуем множество R. На картинке в R войдут элементы, соединенные с S_1. Если теперь в R окажется элемент, не входящий в систему различных представителей, то добавим этот элемент в нашу систему в качестве представителя S_k и перейдем к п. 2.
4. Включим в L все множества, чьи представители входят в R. На рис. 34 в L входят элементы, соединенные с элементами из R жирными отрезками.
5. Из всех элементов множества L (а это какие-то множества исходного набора) выберем элементы, их составляющие, и включим эти элементы в множество R. (Последовательное построение множеств L и R показано на рис. 34.)
6. Если среди элементов R есть такие, которые не представляют ни одного исходного множества (можно доказать, что при выполнении условия теоремы Холла это обязательно когда-то произойдет), то это значит, что существует цепочка, идущая из S_k в какой-то элемент из R, не входящий в систему различных представителей, и проходящая последовательно по невыделенным и выделенным отрезкам. В нашем примере такой цепочкой является S_1, a_1, S_4, a_4. Заменим в этой цепочке все выделенные отрезки не выделенными и наоборот. При этом количество выделенных отрезков увеличится на 1. Это означает, что мы увеличили количество элементов в системе различных представителей. В нашем примере теперь a_1 будет представлять S_1, а a_4 будет представлять S_4. После такого увеличения перейдем к п. 2.
7. Если каждый элемент R представляет какое-либо множество, то перейдем к п. 4.

Теперь покажем, как все вышеизложенное можно применить к нашей задаче. Заметим, что каждая строка нашей таблицы состоит из чисел. Можно считать, что мы имеем множества S_i (всего N штук), составленные из элементов,

которые можно разбить на N классов по принадлежности к строке, в которой этот элемент должен находиться в отсортированной таблице. Нетрудно показать, что эти множества удовлетворяют теореме Холла, т. е. любые k множеств в совокупности содержат элементы не менее чем k классов.

По теореме Холла, из каждой строки таблицы можно выбрать по одному представителю каждого класса, т. е. так, чтобы среди выбранных присутствовало одно число от 1 до N, одно — от $N + 1$ до $2N$, одно — от $2N + 1$ до $3N$ и т. д. Применительно к нашей задаче это означает, что с помощью первого горизонтального хода все выбранные числа можно переместить на первое место — каждое в своей строке.

Итак, на первом месте в каждой строке будут собраны числа, представляющие каждую строку упорядоченной таблицы. А теперь забудем на время про получившийся первый столбец и рассмотрим оставшуюся таблицу. Она будет иметь N строк и $M - 1$ столбцов, и для нее вновь можно рассмотреть исходную задачу. Совершенно аналогичными рассуждениями можно показать, что в оставшейся таблице за один горизонтальный ход во второй столбец можно собрать по одному представителю каждой строки итоговой упорядоченной таблицы.

Очевидно, что формирование первого и второго столбцов можно провести за один ход, так как каждый элемент остается в той строке, в которой он и был. Совершенно аналогично за первый горизонтальный ход формируются и остальные столбцы. Таким образом, за первый горизонтальный ход можно в каждый столбец переместить по одному представителю каждой строки итоговой упорядоченной таблицы.

После этого уже все просто. За второй, вертикальный, ход в каждом столбце можно расставить числа по своим строкам. За третий, опять горизонтальный, ход числа упорядочиваются внутри каждой строки. Тем самым мы доказали, что для достижения поставленной цели достаточно трех ходов! При этом для реализации первого хода необходимо применить вышеизложенный алгоритм, а второй и третий шаги элементарны.

12. Маршрут

В основу решения этой задачи положен *метод динамического программирования*. Если бы на каком-то участке существовал кусочек пути с большей суммой цифр, то, заменив этот кусочек пути на оптимальный, мы тем самым увеличили бы и общую сумму, что привело бы к улучшению решения всей задачи. Указанное свойство позволяет применять метод динамического программирования и называется *свойством Беллмана* (по фамилии автора метода).

Будем искать наилучшие пути, идущие из клетки (1,1), во все остальные клетки таблицы и, в частности, в клетку (N,N). Для этого в некоторой вспомогательной таблице B того же размера, что и A, мы будем отмечать суммы цифр оптимальных путей, ведущих в соответствующие клетки. Так, в клетке (1,1) таблицы B окажется число $A(1,1)$, потому что путь, ведущий в нее, состоит из одной этой клетки. Так же легко заполняются клетки (1,2) и (2,1) таблицы B. В них тоже ведут единственные пути, и суммы цифр вдоль них равны $A(1,1) + A(1,2)$ и $A(1,1) + A(2,1)$, соответственно.

Дальнейшее легко проследить на примере. Пусть таблица A размером 5×5 имеет вид, представленный на рис. 35. Нумерация клеток идет от левого верхнего угла. Проследим за заполнением таблицы B. О клетках (1,2) и (2,1) мы уже говорили. Чтобы заполнить клетку (2,2), заметим, что в нее можно попасть либо из клетки (1,2), либо из клетки (2,1). Следовательно, в клетку (2,2) таблицы B надо записать либо число $B(1,2) + A(2,2)$, либо число $B(2,1) + A(2,2)$ — в зависимости от того, какое из них больше.

Заполнение остальных клеток таблицы B производится аналогично. Если путь, ведущий в эту клетку, единственный,

15. Родственники

Будем формировать из найденных родственников список S. В начале занесем в этот список самого человека X. Выберем из оставшихся людей его близких родственников и занесем их последовательно в S.

Далее двигаемся по списку. Рассматривая очередного человека из списка, выбираем из оставшихся (не занесенных еще в список) людей его близких родственников, и если таковые найдутся, то заносим их в конец списка.

Процесс закончится, когда будет рассмотрен последний человек из списка. При этом может оказаться, что все люди планеты Глюк попали в наш список! ☺

Проиллюстрируем работу этого алгоритма с помощью *графа*. Люди здесь соответствуют вершинам (на рис. 36 они обозначены квадратиками), а близкие родственные отношения — ребрам.

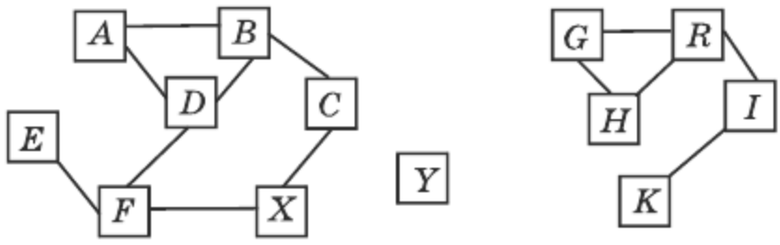


Рис. 36

В следующей таблице изображен результат заполнения списка S:

X	C	F	B	D	E	A	

16. Схема

Коммутатор должен обеспечивать соединения четырех входов с четырьмя выходами в произвольных сочетаниях, которых всего $4! = 24$. Каждое возможное сочетание должно соответствовать своему состоянию схемы. Поскольку один переключатель имеет только два состояния, то четыре пере-

ключателя обеспечат не более $2^4 = 16$ состояний схемы. Следовательно, количество переключателей в требуемом коммутаторе должно быть не менее пяти. Одна из возможных схем такого коммутатора приведена на рис. 37. Предлагаем читателям самостоятельно убедиться в том, что эта схема обеспечивает любые соединения входов с выходами.

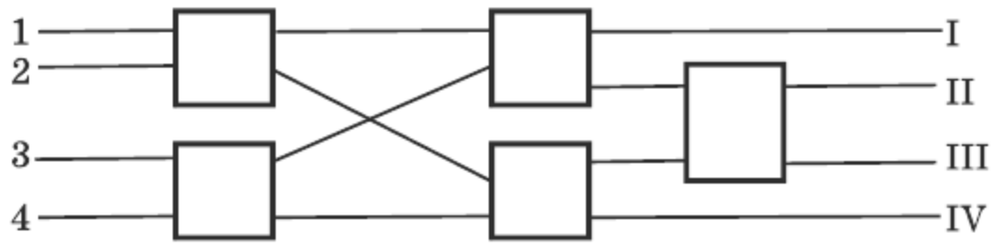


Рис. 37

Практический тур

17. Родословная

Описанная в условии задачи структура данных представляет собой *бинарное дерево*. Необходимо проверить заданный список телят на отсутствие в нем циклов, а также определить, не является ли какой-либо теленок одновременно особью мужского и женского пола. На это может указать номер теленка, который в каком-то паспорте встретился как номер отца, а в каком-то — как номер матери.

18. Параллельные вычисления

Эта задача относится к области *параллельного программирования*. Как правило, задачи в этой области достаточно трудны. Поэтому прежде чем приводить программы для n сумматоров, решающие поставленную задачу, сделаем несколько замечаний и оценок.

Назовем *рангом* суммы чисел количество слагаемых из множества $\{a_1, a_2, \ldots, a_{16}\}$, образующих эту сумму. Так, ранг самих этих чисел равен 1, а ранг частичной суммы $S_k = a_1 + a_2 + \ldots + a_k$ равен k.

Заметим, что в начале работы ранги всех имеющихся в системе чисел равны 1. После первого такта ранги некоторых чисел могут стать равными 2 (но не более); после второго такта ранги чисел будут не более 4; после третьего — не

тактах сумматоры вообще не работают с числами от a_9 до a_{16}, а за один второй такт из этих чисел невозможно составить ни одной суммы, в которую бы входило более двух таких слагаемых. Для получения же на четвертом и пятом тактах сумм от S_9 до S_{16} такие числа нужны.

Итак, получить все частичные суммы за 5 тактов нельзя.

За 6 тактов при $n = 4$ задача решается следующей программой:

Сумматор	Такт	Команда	Комментарий
1	1	R21=R1+R2	$S_2 = a_1 + a_2$
2	1	R22=R3+R4	$S_{3...4} = a_3 + a_4$
3	1	R23=R5+R6	$S_{5...6} = a_5 + a_6$
4	1	R24=R7+R8	$S_{7...8} = a_7 + a_8$
1	2	R25=R21+R22	$S_4 = S_2 + S_{3...4}$
2	2	R26=R23+R24	$S_{5...8} = S_{5...6} + S_{7...8}$
3	2	R27=R9+R10	$S_{9...10} = a_9 + a_{10}$
4	2	R28=R11+R12	$S_{11...12} = a_{11} + a_{12}$
1	3	R29=R25+R23	$S_6 = S_4 + S_{5...6}$
2	3	R30=R25+R26	$S_8 = S_4 + S_{5...8}$
3	3	R31=R27+R28	$S_{9...12} = S_{9...10} + S_{11...12}$
4	3	R32=R15+R16	$S_{15...16} = a_{15} + a_{16}$
1	4	R33=R29+R7	$S_7 = S_6 + a_7$
2	4	R34=R30+R27	$S_{10} = S_8 + S_{9...10}$
3	4	R35=R30+R31	$S_{12} = S_8 + S_{9...12}$
4	4	R36=R13+R14	$S_{13...14} = a_{13} + a_{14}$
1	5	R37=R25+R5	$S_5 = S_4 + a_5$
2	5	R38=R34+R11	$S_{11} = S_{10} + a_{11}$
3	5	R39=R35+R13	$S_{13} = S_{12} + a_{13}$
4	5	R40=R35+R36	$S_{14} = S_{12} + S_{13...14}$
1	6	R41=R21+R3	$S_3 = S_2 + a_3$
2	6	R42=R30+R9	$S_9 = S_8 + a_9$
3	6	R43=R40+R15	$S_{15} = S_{14} + a_{15}$
4	6	R44=R40+R32	$S_{16} = S_{14} + S_{15...16}$

При $n = 2$ оценку на количество тактов получить не удалось. Более того, жюри перед олимпиадой знало решение за 11 тактов, но на олимпиаде один из ее участников, В. Булатов предложил программу в 10 тактов! Вот эта программа.

Сумматор	Такт	Команда	Комментарий
1	1	R21=R1+R2	$S_2 = a_1 + a_2$
2	1	R22=R3+R4	$S_{3...4} = a_3 + a_4$
1	2	R23=R21+R3	$S_3 = S_2 + a_3$
2	2	R24=R21+R22	$S_4 = S_2 + S_{3...4}$
1	3	R25=R24+R5	$S_5 = S_4 + a_5$
2	3	R26=R6+R7	$S_{6...7} = a_6 + a_7$
1	4	R27=R25+R6	$S_6 = S_5 + a_6$
2	4	R28=R25+R26	$S_7 = S_5 + S_{6...7}$
1	5	R29=R28+R8	$S_8 = S_7 + a_8$
2	5	R30=R9+R10	$S_{9...10} = a_9 + a_{10}$
1	6	R31=R29+R9	$S_9 = S_8 + a_9$
2	6	R32=R29+R30	$S_{10} = S_8 + S_{9...10}$
1	7	R33=R32+R11	$S_{11} = S_{10} + a_{11}$
2	7	R34=R12+R13	$S_{12...13} = a_{12} + a_{13}$
1	8	R35=R33+R12	$S_{12} = S_{11} + a_{12}$
2	8	R36=R33+R34	$S_{13} = S_{11} + S_{12...13}$
1	9	R37=R36+R14	$S_{14} = S_{13} + a_{14}$
2	9	R38=R15+R16	$S_{15...16} = a_{15} + a_{16}$
1	10	R39=R37+R15	$S_{15} = S_{14} + a_{15}$
2	10	R40=R37+R38	$S_{16} = S_{14} + S_{15...16}$

Олимпиады 1993/94 учебного года

Городская олимпиада

Теоретический тур

1. Перестановки

Обсуждаемый в задаче алгоритм является немного видоизмененным известным *алгоритмом сортировки методом пузырька*. В классическом варианте этого метода выбор пары элементов для перестановки осуществляется последовательно от края таблицы. При этом гарантируется, что через некоторое определенное количество перестановок самый большой элемент окажется в конце таблицы и в дальнейших перестановках больше участия не принимает. Еще через несколько шагов на «свое» место в таблице встанет следующий по величине элемент и т. д.

Однако в нашем случае не задан порядок просмотра таблицы, и гарантировать конечность алгоритма с помощью тех же аргументов мы уже не можем. Для этого можно предложить другой путь.

Подсчитаем для таблицы число S, равное количеству пар элементов, в которых первый (раньше расположенный в таблице) элемент больше второго, причем элементы в такой паре могут и не следовать непосредственно один за другим. Что происходит с числом S после перестановки двух соседних элементов? Очевидно, что взаимное расположение остальных элементов (кроме указанных) не изменяется, т. е. S на каждом шаге алгоритма уменьшается на 1. Ясно, что $S = 0$ соответствует полностью упорядоченной таблице, а меньше нуля S быть не может. Таким образом, количество шагов в алгоритме обязано равняться целому числу S. Максимально возможным S будет, когда исходная таблица упорядочена по убыванию. При этом первый элемент больше второго в любой паре элементов. Всего пар элементов в таблице длины N имеется $N \cdot (N - 1)/2$. Это и есть ответ на второй вопрос задачи.

2. Химия

В условии задачи сказано, что реакция между веществами происходит мгновенно, поэтому все такие реакции успевают произойти прежде, чем очередное вещество будет добавлено

в пробирку. Состояние содержимого пробирки тогда можно описать с помощью некоторой линейной таблицы, содержащей номера веществ, расположенных в пробирке начиная с нижнего слоя. При добавлении нового вещества его номер должен быть записан в конец таблицы. Новое вещество вместе с предыдущим (лежащим ниже в пробирке) могут не взаимодействовать, т. е. все останется как есть, а могут и исчезнуть, оставив после себя какое-то другое вещество, которое вновь может вступить в реакцию с предыдущим. В любом случае изменения в нашей таблице происходят только с одного конца, поэтому в качестве информационной модели процесса удобно использовать *стек*, содержимое которого соответствует содержимому пробирки.

Алгоритм решения задачи будет выглядеть следующим образом.

1. Если веществ больше нет, то переходим к п. 7.
2. Добавляем очередное вещество в пробирку.
3. Если нет вещества, лежащего ниже, то переходим к п. 1.
4. Если реакция между верхним веществом (пусть это будет i-е вещество) и нижележащим (j-е) не идет, то переходим к п. 1.
5. Вместо двух верхних веществ образуется новое: $k = A[i,j]$.
6. Переходим к п. 3.
7. Перебирая сверху вниз вещества в пробирке, выводим их номера.

3. Фибоначчи

Обозначим количество обращений к алгоритму F при вычислении $F(N)$ через $S(N)$. Заметим, что вычисление $F(1)$ и $F(2)$ не требует рекурсивного обращения к алгоритму F, т. е. $S(1) = 0$ и $S(2) = 0$.

При вычислении $F(N)$ при $N > 2$ мы должны вычислить $F(N - 1)$ и $F(N - 2)$, т. е. дважды обратиться к алгоритму F, и в каждом из этих случаев количество рекурсивных обращений к F составит $S(N - 1)$ и $S(N - 2)$, соответственно. Тогда $S(N) = 2 + S(N - 1) + S(N - 2)$. А поскольку $S(1)$ и $S(2)$

мы уже знаем, легко последовательно подсчитать следующие значения:

$S(3) = 2 + S(2) + S(1) = 2 + 0 + 0 = 2;$
$S(4) = 2 + S(3) + S(2) = 2 + 2 + 0 = 4;$
$S(5) = 2 + S(4) + S(3) = 2 + 4 + 2 = 8;$
$S(6) = 2 + S(5) + S(4) = 2 + 8 + 4 = 14;$
$S(7) = 2 + S(6) + S(5) = 2 + 14 + 8 = 24;$
$S(8) = 2 + S(7) + S(6) = 2 + 24 + 14 = 40;$
$S(9) = 2 + S(8) + S(7) = 2 + 40 + 24 = 66;$
$S(10) = 2 + S(9) + S(8) = 2 + 66 + 40 = 108;$
$S(11) = 2 + S(10) + S(9) = 2 + 108 + 66 = 176;$
$S(12) = 2 + S(11) + S(10) = 2 + 176 + 108 = 286;$
$S(13) = 2 + S(12) + S(11) = 2 + 286 + 176 = 464;$
$S(14) = 2 + S(13) + S(12) = 2 + 464 + 286 = 752;$
$S(15) = 2 + S(14) + S(13) = 2 + 752 + 464 = 1218;$
$S(16) = 2 + S(15) + S(14) = 2 + 1218 + 752 = 1972;$
$S(17) = 2 + S(16) + S(15) = 2 + 1972 + 1218 = 3192;$
$S(18) = 2 + S(17) + S(16) = 2 + 3192 + 1972 = 5166;$
$S(19) = 2 + S(18) + S(17) = 2 + 5166 + 3192 = 8360;$
$S(20) = 2 + S(19) + S(18) = 2 + 8360 + 5166 = 13\,528.$

4. Вычисление

Словесное описание этого алгоритма таково. Многократно случайно выбираются координаты x и y точки, расположенной в квадрате $[0,1] \times [0,1]$. Если эта точка удалена от начала координат не более чем на 1, т. е. находится в четверти единичного круга (рис. 38), то счетчик S увеличивается на 1.

Рис. 38

В результате S равно количеству точек, попавших в закрашенную область. Затем S делится на n, т. е. оказывается равно доли точек, попавших в эту область. Так как точек много, то доля этих точек приблизительно равна отношению площади закрашенной области к площади единичного квадрата, т. е. $\pi/4$. Такой метод подсчета площади носит название *метода Монте-Карло*. Ответ к задаче: алгоритм вычисляет значение $\pi/4$ методом Монте-Карло.

5. Дни в месяце

Данная задача допускает различные подходы к решению, начиная от применения интерполяционных многочленов (при этом используется около 30 операций) до экзотических функций типа $D = \mathtt{int}(A \cdot (\mathtt{sin}(B \cdot n)))$, где A и B — некоторые константы, подбор которых является довольно сложной математической проблемой. Наиболее естественным, простым и достаточно эффективным является применение функций abs — абсолютная величина и sgn — знак числа. Те, кто изучал графики функций, содержащих абсолютные величины, знают, что эти графики могут иметь достаточно «изломанный» вид и, в частности, проходить через многие заранее заданные точки. Если этих точек 12 и координаты этих точек имеют вид $(n, D(n))$, где n — целое число от 1 до 12 (номер месяца), а $D(n)$ — количество дней в этом месяце, то наша функция и будет искомой. Вот пример одной из таких функций:

$$D = 2 \cdot n \cdot (\mathtt{sgn}(n-2.5)-1)+\mathtt{abs}(\mathtt{abs}(\mathtt{abs}(n-7.5)-5)-2)+30.$$

Первое слагаемое здесь добавлено для корректировки функции в точках 1 и 2, соответствующих январю и февралю.

График функции D приведен на рис. 39.

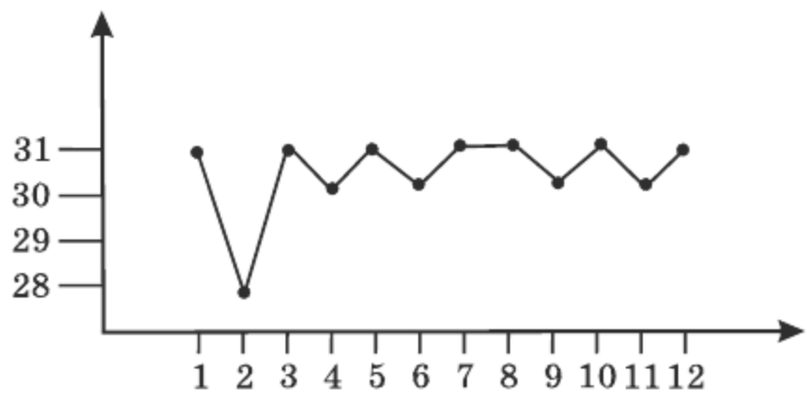


Рис. 39

6. Признак делимости

При делении числа 16^k на 5 в остатке всегда получается 1 (хотя бы потому, что любая степень числа, оканчивающегося на 6, тоже оканчивается на 6). Любое шестнадцатеричное число записывается в виде

$$A_0 \cdot 16^k + A_1 \cdot 16^{k-1} + A_2 \cdot 16^{k-2} + \ldots + A^k,$$

где A_i — шестнадцатеричные цифры. При делении на 5 в остатке получается сумма остатков от деления каждого слагаемого, т. е. $A_0 + A_1 + A_2 + \ldots + A_k$. Итак, шестнадцатеричное число делится на 5 тогда и только тогда, когда на 5 делится сумма его цифр. (На тех же соображениях основываются признаки делимости на 3 и на 9 в десятичной системе счисления.)

7. Кольцевая дорога

Предположим, что искомая дорога построена. На ней должны лежать некоторые точки, иначе радиус дороги можно было бы уменьшить на значение, равное расстоянию от дороги до ближайшей точки. Уменьшение радиуса возможно и в случае, если на дороге лежит только одна точка, — для этого достаточно предварительно сдвинуть центр окружности в сторону этой точки на малое расстояние (чтобы в окружность не попали и какие-то другие точки). Если же точек две и расположены они не в концах диаметра, то перед уменьшением радиуса и в этом случае можно на малое расстояние сдвинуть центр окружности в сторону середины хорды, соединяющей эти две точки. Воспрепятствовать такому уменьшению радиуса точки могут, только являясь концами диаметра окружной дороги.

Итак, для искомой дороги выполняется один из следующих случаев:

1) какие-либо две точки являются концами ее диаметра;
2) какие-либо три точки лежат на дороге.

Алгоритм построения дороги должен исследовать эти случаи, т. е. проверить дороги, построенные по каждой паре точек, взятых в качестве концов диаметра, и дороги, описанные около каждой тройки точек. Дорога, содержащая внутри себя все остальные точки и имеющая минимальный радиус, является искомой.

Для значительного сокращения перебора могут быть использованы следующие соображения:

1) достаточно рассматривать только те тройки точек, которые образуют остроугольный треугольник (так как в случае тупоугольного треугольника тоже возможен сдвиг центра окружности с последующим уменьшением радиуса);

2) среди всех троек точек достаточно выбрать такие, которые определяют треугольник с наибольшим радиусом описанной окружности. Проверять же, что все остальные точки при этом оказываются внутри окружности, необязательно: действительно, тройка точек, определяющая такую окружность, не сможет поместиться в круг меньшего радиуса!

Практический тур

8. Точки в круге

Сначала установим следующий факт: если число α — иррациональное, то расстояния от точки $(\alpha, 1/3)$ до всех целочисленных точек плоскости различны.

Действительно, предположим противное и допустим, что имеет место равенство расстояний для двух различных целочисленных точек (x_1, y_1) и (x_2, y_2):

$$(\alpha - x_1)^2 + (1/3 - y_1)^2 = (\alpha - x_2)^2 + (1/3 - y_2)^2.$$

Тогда после раскрытия скобок получаем:

$$-2\alpha x_1 + (1/3 - y_1)^2 = -2\alpha x_2 + (1/3 - y_2)^2,$$

т. е. равенство:

$$\alpha(2x_2 - 2x_1) = r,$$

где r — некоторое рациональное число (равное соответствующей разности квадратов). Если $x_2 \neq x_1$, то получается противоречие, так как тогда α представляется как частное двух рациональных чисел: $\alpha = r/(2x_2 - 2x_1)$. Значит, $x_2 = x_1$, и тогда мы получаем другое равенство:

$$(1/3 - y_1)^2 = (1/3 - y_2)^2,$$

откуда следует, что $y_1 = y_2$. Но это означает совпадение исходных целочисленных точек.

Теперь возможна следующая теоретическая рекомендация. Нужно рассматривать окружности с центром в точке $(\alpha, 1/3)$, радиусы которых возрастают от 0 до бесконечности. В соответствии с доказанным фактом все целочисленные

точки плоскости строго упорядочены по возрастанию расстояния от выбранного центра. Поэтому для любого n существует соответствующий радиус, при котором внутри окружности содержится ровно n целочисленных точек.

Практически в качестве α можно выбрать квадратичную иррациональность, а в качестве соответствующих приближений для α брать подходящие дроби, получающиеся из разложения α в непрерывную дробь. (Напомним, что разложение квадратичной иррациональности в непрерывную дробь периодично; все вычисления можно производить в целых числах, требуется только обеспечить «длинную» арифметику.)

В заключение заметим, что при вычислениях с фиксированной точностью при решении исходной задачи могут возникнуть проблемы, особенно при больших значениях n. Выходом может быть увеличение точности при проведении расчетов.

9. Ребусы

Решение этой задачи предполагает аккуратную реализацию *переборного алгоритма*. В первых двух ребусах достаточно в качестве сомножителей перебрать все трехзначные числа. В третьем ребусе надо перебрать все пары трехзначных сомножителей.

Областная олимпиада

Теоретический тур

10. Лототрон

Рассмотрим сумму чисел на всех шарах. Заметим, что эта сумма на каждом шаге уменьшается, но остается положительной. Отсюда следует, что описанный в задаче процесс не может продолжаться бесконечно долго. Однако пока в лототроне имеется более одного шара, мы всегда можем выбрать два шара для очередного хода. Следовательно, рано или поздно в лототроне должен остаться один шар.

Теперь покажем, что число d, которым помечен последний шар, является делителем всех чисел, которые были в лототроне. Начнем рассуждать с конца. Ясно, что последний удаленный из лототрона шар был помечен тем же числом d.

Перед этим там могли находиться либо два шара с числами $2d$ и d, либо три шара с числами d, d и d. В любом случае d делит все эти числа. Предположим, что мы сделали несколько шагов назад, причем указанное свойство сохранилось. Но тогда за секунду до этого в лототроне находился либо тот же набор чисел, либо вместо какого-то числа имелась сумма двух чисел, делящихся на d, т. е. тоже делящееся на d число. Утверждение доказано.

Отметим также, что если два числа a и b делятся на d, то и два числа $(a - b)$ и b тоже делятся на d, т. е. общие делители для всех имеющихся чисел во время работы лототрона не меняются.

Таким образом, оставшееся число d, во-первых, является делителем всех исходных чисел, а во-вторых, делится на любой другой делитель этих чисел, т. е. является НОД исходных чисел.

11. Латинский квадрат

Назовем имеющиеся k строк латинского квадрата «латинским прямоугольником». В каждом столбце такого прямоугольника находятся некоторые k чисел из набора 1, 2, 3, ..., n. Для каждого столбца «латинского прямоугольника» запишем множества чисел от 1 до n, которые не входят в этот столбец. Эти n множеств содержат по $n - k$ элементов. Каждое из чисел от 1 до n встречается в «латинском прямоугольнике» k раз (по одному разу в каждой строке), — значит, каждое число от 1 до n входит в $n - k$ различных множеств. Согласно *следствию теоремы Холла* (см. решение задачи 92/93(11)), существует система различных представителей этих множеств. Эта система может быть взята в качестве $(k + 1)$-й строки латинского прямоугольника. Построить ее можно с помощью алгоритма, описанного в решении задачи 92/93(11). Повторяя указанный процесс несколько раз, мы в конце концов построим искомый латинский квадрат.

12. Дешифровка

Количество чисел, превышающих заданное число и стоящих правее в исходной перестановке, очевидно, не больше, чем всего имеется чисел, стоящих справа. Поэтому необходимым условием того, что данная последовательность являет-

ся шифром некоторой перестановки, является тот факт, что каждое число последовательности не превышает количества позиций, находящихся справа от него.

Докажем, что это условие является и достаточным. Рассмотрим позицию i в исходной перестановке, на которой находится число n. Это самое большое число, поэтому левее позиции i в шифре не может стоять нуль (так как по крайней мере одно большее число в исходной перестановке имеется справа). В самой же позиции i стоит нуль, так как правее числа n больших чисел нет. Итак, самый левый нуль в шифре соответствует числу n в исходной перестановке. Ясно, что в шифре нуль должен быть хотя бы потому, что нуль стоит на последнем месте (правее нет ни одного числа).

После того, как число n расшифровано, его можно вычеркнуть, уменьшить на единицу все числа последовательности, стоящие левее, и продолжать расшифровку перестановки чисел от 1 до $n - 1$. Для нее можно проделать все указанные действия, причем полученная последовательность будет снова удовлетворять нашему условию.

Как видим, доказав необходимое и достаточное условие того, что данная последовательность является шифром некоторой перестановки, мы одновременно уже привели и алгоритм расшифровки этой перестановки. Для этого, однако, можно предложить и еще один, более простой, алгоритм.

Рассмотрим первое число в последовательности. Пусть это k. Это означает, что в первой позиции исходной перестановки стоит число, для которого имеются k чисел, больших его. Значит, это число равно $n - k$. Расшифровав это число, можно перейти к расшифровке следующего числа и т. д. Очевидно, что при этом надо учитывать, что какие-то числа уже разгаданы и находиться правее рассматриваемой позиции не могут.

13. Кулинария

Одной из возможных формальных моделей данной задачи является *ориентированный граф*. Все операции ставятся в соответствие вершинам графа. Дуга, идущая из одной вершины в другую, представляет ситуацию, когда операция,

соответствующая первой вершине, должна предшествовать операции, соответствующей второй вершине. Отсутствие же дуги между вершинами означает, что данные операции не находятся в отношении непосредственного предшествования друг другу.

В описанной модели решением задачи является некоторое упорядочение вершин (их нумерация числами от 1 до n), соответствующее порядку выполнения операций. При этом ни одна дуга не должна идти из вершины с большим номером в вершину с меньшим номером. (Эта ситуация означала бы «обратный ход времени».)

Очевидно, что необходимым условием существования решения задачи является отсутствие в графе *циклов*, т. е. замкнутых путей вдоль дуг, так как вершины такого цикла было бы невозможно упорядочить. Покажем, что указанное условие также является и достаточным.

Нетрудно показать, что если циклов в графе нет, то существует вершина, в которую не входит ни одна дуга. В самом деле, — если бы такой вершины не было, то, двигаясь по входящим в вершины дугам в обратном направлении, мы обязательно когда-нибудь пришли бы в вершину, в которой уже побывали ранее, т. е. обнаружился бы цикл. Тогда вершине, в которую не входят дуги, можно присвоить номер 1 (соответствующая операция может выполняться первой). Исключив эту вершину из графа вместе со всеми смежными ей дугами, продолжим указанный процесс, — это позволит выделить вершину с номером 2. Аналогично, повторяя указанные выше действия, можно определить и пронумеровать все остальные вершины вплоть до последней с номером n.

14. Алгоритм

Прежде всего отметим, что в процессе работы данного алгоритма сумма значений a и b не изменяется ни при одном рекурсивном вызове функции F. Окончание работы алгоритма предусмотрено только в случае $b = 0$, значит, если алгоритм завершится, то его значением будет сумма исходных чисел. Конечность же алгоритма видна из того, что в каждом новом вызове F значение b уменьшается по модулю на 1, т. е. когда-то оно обязательно станет равным нулю.

15. Джонсон & Джонсон

Приведенная задача является хорошо известной «*задачей двух станков*». Ее решает *алгоритм Джонсона*, носящий имя известного специалиста в области *теории расписаний* С. Джонсона, предложившего и обосновавшего этот алгоритм.

Суть алгоритма Джонсона заключается в следующем. Выбираем из множества значений t_i и f_i минимальное. Если это t_i, то i-ю деталь ставим на обработку первой; если минимальным оказывается f_i, то i-ю деталь ставим на обработку последней. Исключаем i-ю деталь из рассмотрения и рекурсивно повторяем вышеописанный процесс для оставшихся деталей.

Для обоснования этого алгоритма используется *перестановочный прием*, предложенный С. Джонсоном. (От участников олимпиады требовалось только решить задачу при помощи данного алгоритма, но его обоснование не требовалось.)

16. Подстрока

При решении задачи (А) алгоритм ***П1*** выделяет первые символы строк и сравнивает их. Если они совпадают, то алгоритм повторяется для оставшейся части строк и т. д. (Обратите внимание на условие завершения рекурсии.)

```
Алг лит П1(лит A,B)
нач лит A1,A2,B1,B2
  если B = ""                | строка B кончилась
    то знач:="Да"
    иначе
      если A = ""
        то знач:="Нет"
        иначе ПС(A,A1,A2)   |выделяем первые символы
              ПС(B,B1,B2)   |и остатки строк
        если A1=B1
          то знач:=П1(A2,B2)
          иначе знач:="Нет"
      все
    все
  все
кон
```

Для решения задачи (Б) используем алгоритм ***П1***. Здесь при несовпадении букв мы будем пытаться опознать строку *B* начиная со второго символа строки *A*; если опять неудача, то с третьего и т. д.

```
Алг лит П2(лит A,B)
нач лит A1,A2
  если A = ""
    то знач:="Нет"
    иначе знач:=П1(A,B)
        если знач = "Нет"
          то ПС(A,A1,A2); знач:=П2(A2,B)
        все
  все
кон
```

Практический тур

17. Игра

Несмотря на то, что в описанной игре всего 9 полуходов (причем последний из них — чисто формальный), она достаточно сложна и интересна как для игры «на бумажке» между людьми, так и для программирования. При решении этой задачи можно составлять различные *эвристические алгоритмы*. Например, если мы играем за первого игрока и нам приписывается сумма произведений по строкам, то можно пытаться собрать в какой-то одной строке три больших или три маленьких числа. Так, если нам удастся собрать в одной строке числа 1, 2 и 3, то мы гарантируем себе крупный выигрыш за счет двух оставшихся строк. Можно учитывать, что противник хочет сделать то же самое, и мешать ему сделать это. (В реальном турнире программ участников победу над «умными» процедурами одержала программа, которая в ячейку с самым большим номером ставила самое большое число, — она обычно успевала накопить в третьей строке три больших числа.)

Опишем подход, который (при некоторой оптимизации) можно довести до полного решения задачи. В данной игре возможно 9! = 362 880 окончательных позиций. Заметим, однако, что при перестановке строк и/или столбцов результат

игры не меняется. Поэтому можно считать, например, что в первой клеточке стоит число 1, а сумма второго столбца (и строки) меньше, чем сумма третьего. При этом количество позиций сократится в 36 раз и составит 10 080. Далее для каждой позиции мы, начиная с конца игры, строим функции, показывающие оптимальный результат игры при продолжении с этой позиции. Первая функция просто дописывает оставшееся число в оставшуюся ячейку и вычисляет результат игры. Вторая функция работает при двух незаполненных ячейках, перебирая четыре оставшихся варианта продолжения игры, вызывая первую функцию и находя лучший из вариантов хода и результат игры. (Заметим, что для оптимизации можно рассмотреть только два варианта хода.) Каждая последующая функция работает с уже построенной предыдущей, перебирая k вариантов хода (где k — количество оставшихся свободных клеточек), находит лучший из них и результат игры. Даже без оптимизации функции, работающие для 1, 2, 3, 4, 5, 6 и 7 свободных клеточек, будут очень быстро находить правильный ход. Для 8 и 9 клеточек можно либо оптимизировать работу функций, либо составить какие-то простые правила первых ходов (дебютов), — например, ставить 5 первым ходом в любую клетку.

18. Сумма простых

Для решения задачи используем *метод динамического программирования.*

Заполним таблицу S размером $A \times N$, в каждой ячейке (i,j) которой мы будем хранить количество способов представления числа i в виде суммы j простых слагаемых. Очевидно, что элемент первой строки будет равен единице, если номер строки является простым числом, и нулю — в противном случае. Далее мы заполняем таблицу построчно слева направо, используя формулу:

$$S[i, j] = S[i - 2, j - 1] + S[i - 3, j - 1] + \ldots + S[i - p, j - 1],$$

где p — наибольшее простое число, меньшее i. Также удобно считать что $S[i, j] = 0$ для всех отрицательных i.

Число в ячейке $S[A, N]$ будет являться количеством способов представления числа A в виде N простых слагаемых.

Олимпиады 1994/95 учебного года

Городская олимпиада

Теоретический тур

1. Капитал

Программа должна рассчитывать ежедневно растущую сумму вклада. При этом рекомендуется в качестве вспомогательной использовать функцию преобразования текущей даты в следующую. В частности, в ней должны использоваться информация о количестве дней в каждом месяце и правило вычисления високосных лет (см. решение задачи 87/88(2)).

2. Вирус

Эта задача «родилась» из известной математической задачи — как за наименьшее количество взвешиваний при помощи рычажных весов найти среди двенадцати монет фальшивую, если неизвестно, легче она или тяжелее обычной. В данной задаче слияние файлов аналогично складыванию монет в кучки, а сравнение длин — взвешиванию пары таких кучек на весах. Таким образом, задача о файлах может быть решена совершенно аналогично задаче о монетах. При изложении алгоритма знаком «+» мы будем обозначать файлы, слитые в один. Знак «?» означает сравнение файлов, а знаки «>», «<» и «=» — результат такого сравнения.

```
F1+F2+F3+F4 ? F5+F6+F7+F8  |первое сравнение
случай = |испорченный — среди файлов
         |F9, F10, F11 или F12
   F1+F2+F3 ? F9+F10+F11     |второе сравнение
   случай = |испорчен файл F12
   случай < |испорченный — F9, F10 или F11,
            |и он длиннее
      F9 ? F10                  |третье сравнение
      случай = |испорчен файл F11
      случай < |испорчен файл F10
      случай > |испорчен файл F9
   случай > |испорченный — F9, F10 или F11, и он короче
```

```
      F9 ? F10                    |третье сравнение
      случай = |испорчен файл F11
      случай < |испорчен файл F9
      случай > |испорчен файл F10
   случай < |испорчен файл F1, F2, F3 или F4,
            |и он короче, либо испорчен файл F5, F6,
            |F7 или F8, и он длиннее
   F1+F2+F3+F5 ? F4+F9+F10+F11  |второе сравнение
   случай = |испорчен файл F6, F7 или F8
      F6 ? F7                     |третье сравнение
      случай = |испорчен файл F8
      случай < |испорчен файл F7
      случай > |испорчен файл F6
   случай < |испорчен файл F1, F2 или F3
      F1 ? F2                     |третье сравнение
      случай = |испорчен файл F3
      случай < |испорчен файл F1
      случай > |испорчен файл F2
   случай > |испорчен файл F4 или F5
      F1 ? F4                     |третье сравнение
      случай = |испорчен файл F5
      случай < |испорчен файл F1
      случай > |испорчен файл F4
   случай > |испорчен файл F5, F6, F7 или F8,
            |и он короче, либо испорчен файл F1,
            |F2, F3 или F4, и он длиннее
|(далее действия полностью аналогичны
|предыдущему случаю)
```

При решении задачи с монетами мы были ограничены тем, что одна монета не может быть использована дважды в одном взвешивании. При работе же с файлами у нас есть возможность сливать их сколько угодно раз (даже один и тот же файл!). Можно воспользоваться этим для получения чуть более простого решения.

```
F1+F2+F3+F4 ? F5+F6+F7+F8              |первое сравнение
случай = |как в первом алгоритме
случай <
  F1+F2+F3+F5+F6+F7 ? F9+F9+F9+F9+F9+F9 |второе сравнение
  случай = |испорчен файл F4 (короче) или F8 (длиннее)
```

```
    F4 ? F9                        |третье сравнение
    случай = |испорчен файл F8
    случай < |испорчен файл F4
случай < |испорчен файл F1, F2 или F3, и он короче
    F1 ? F2                        |третье сравнение
    случай = |испорчен файл F3
    случай < |испорчен файл F1
    случай > |испорчен файл F2
случай > |испорчен файл F5, F6 или F7, и он длиннее
    F5 ? F6                        |третье сравнение
    случай = |испорчен файл F7
    случай < |испорчен файл F6
    случай > |испорчен файл F5
случай >   |аналогичен предыдущему,
           |как и в первом алгоритме
```

3. Библиотека алгоритмов

Линейная таблица, о которй идет речь в задаче, — это не что иное как строка. Условие того, что в строке нет положительных чисел, можно переформулировать в виде: «максимальный элемент строки не превосходит 0» или «сумма элементов строки с противоположным знаком совпадает с суммой модулей». Поэтому два самых простых решения пункта (А) предложенной задачи заключаются в проверке условия:

$$\boldsymbol{MAX}(n,X) \leqslant 0 \quad \text{или} \quad \boldsymbol{SUMMOD}(n,X) = -\boldsymbol{SUM}(n,X).$$

Чтобы найти минимум в строке, используя приведенные в условии задачи библиотечные алгоритмы (пункт (Б)), можно найти максимум в строке с измененными знаками и взять его с противоположным знаком. (После этого нужно восстановить знак всех чисел в исходной строке.) Соответствующий алгоритм будет иметь следующий вид:

```
SIGN(n,X)
min:=-MAX(n,X)
SIGN(n,X)     |восстановление исходной строки
```

Чтобы определить, что в строке имеется не менее $n-1$ нулей, надо сравнить сумму модулей элементов с максимальным по модулю элементом, и если они совпадут, то действи-

тельно, только одно число из строки может быть отличным от 0. Поэтому условие, которое надо проверять при решении пункта (В) задачи, имеет вид:

$$\boldsymbol{SUMMOD}(n,X) = \boldsymbol{MAXMOD}(n,X) \text{ и } \boldsymbol{SUMMOD}(n,X) > 0.$$

4. Альфа

Данный алгоритм вычисляет два очередных числа ряда Фибоначчи. Ряд Фибоначчи имеет вид: 1, 1, 2, 3, 5, 8, 13, ..., где числа $F(n)$, начиная с третьего, вычисляются по рекуррентной формуле: $F(n) = F(n-1) + F(n-2)$.

Чтобы об этом догадаться, достаточно выполнить алгоритм для нескольких значений n. Доказательство легко проводится методом математической индукции.

5. Сейф

Рассмотрим отдельно случаи (А) и (Б).

А) $n = 3$, $k = 3$. Удобно воспользоваться геометрической интерпретацией задачи. Представим себе трехмерную решетку $3 \times 3 \times 3$ (рис. 40).

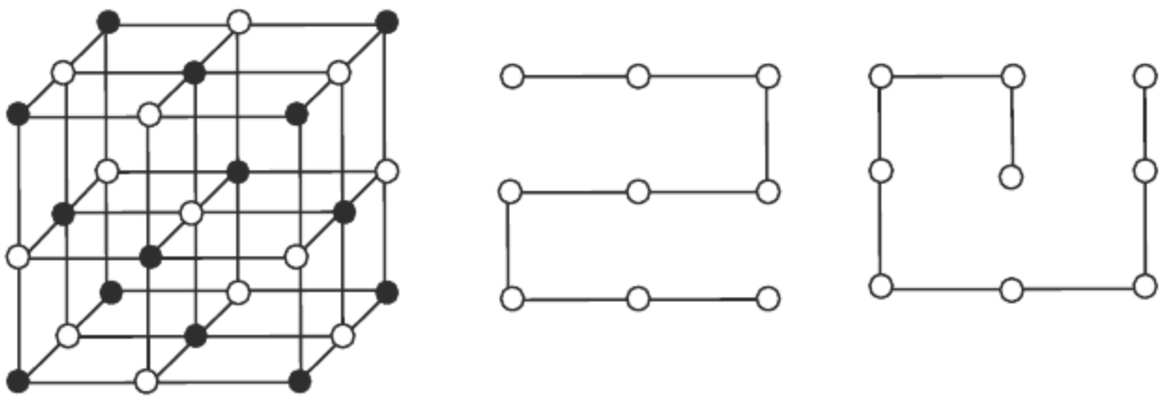


Рис. 40

Координаты узлов этой решетки — целые числа от 0 до 2. Ясно, что каждой возможной комбинации кодового замка в нашем случае соответствуют координаты одного из узлов. Перевод одного из рычагов в соседнее положение означает переход из текущего узла решетки в один из соседних узлов. Таким образом, мы должны выяснить, можно ли обойти узлы решетки, побывав в каждом один раз, если начало обхода задано.

Раскрасим узлы в шахматном порядке. Любой маршрут по ребрам решетки тогда чередует черные и белые узлы. Мы должны пройти 13 белых и 14 черных узлов. Это возможно, только если мы начнем путь из черного узла. Значит, существуют начальные комбинации кодового замка, которые делают стоящую перед грабителями задачу неразрешимой, — эти комбинации характеризуются нечетной суммой чисел, стоящих напротив каждого рычага. Однако из всех остальных положений обход возможен. Заметим, что разрешенными начальными узлами остались либо углы нашей решетки, либо середины внешних граней. В первом случае обход возможен по слоям — «змейкой», а во втором — первый слой проходится по спирали, а последующие — «змейкой» (см. рис. 40, справа).

Б) $n = 4$, $k = 4$. В этом случае геометрическая интерпретация затруднительна, так как она уже требует выхода в четырехмерное пространство. Поэтому разобьем задачу на части меньшей размерности. Рассмотрим отдельно первую и вторую пары рычагов. Каждой из этих пар соответствует двумерная решетка 4×4. Состояние замка в этом случае описывается парой узлов — по одному на каждой решетке. Переход из одного положения замка в другое разрешается по ребру либо первой, либо второй решетки. Опишем алгоритм обхода всех возможных положений замка из любого начального положения.

Обход одной двумерной решетки возможен из любого начального положения. Все принципиально различные случаи указаны на рис. 41.

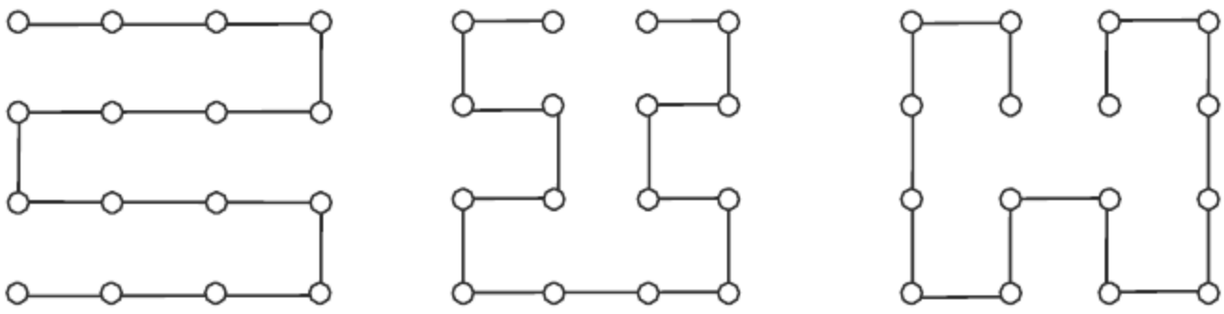


Рис. 41

Алгоритм обхода.

1. Зафиксируем начальное положение второй решетки и обойдем все узлы первой решетки по одной из указанных на рис. 41 схем.
2. Сделаем шаг по маршруту обхода на второй решетке и перейдем к п. 1. Если же вторая решетка пройдена полностью, то закончим работу.

6. Алгоритм Маркова

Если последняя цифра числа отличается от 9, то процесс прибавления 1 сведется просто к замене последней цифры на следующую по порядку. Поскольку после последней цифры стоит символ «*», то соответствующие правила замены последней цифры (или подстановки) будут иметь вид:

(«0*»,«1»), («1*»,«2»), («2*»,«3»), («3*»,«4»), («4*»,«5»), («5*»,«6»), («6*»,«7»), («7*»,«8»), («8*»,«9»).

Если последняя цифра — 9, то мы должны заменить ее на 0 и начать обрабатывать по описанным правилам предыдущую цифру. Для этого мы переставим «*» на одну позицию влево, чтобы она как бы отмечала ту цифру, которую мы обрабатываем в данный момент. Соответствующая подстановка будет иметь вид: («9*», «*0»).

Этот алгоритм будет нормально обрабатывать почти все числа. При замене 9 на 0 он начинает по тем же правилам обрабатывать цифру более старшего разряда, а когда «*» дойдет до цифры, отличной от 9, то эта цифра возрастет на 1, «*» исчезнет и алгоритм завершит работу, так как в левой части всех подстановок «*» присутствует. Исключение составляют числа, состоящие только из 9. При обработке таких чисел алгоритм заменит все 9 на 0, после чего перед «*» не будет никаких цифр, и алгоритм завершит работу с неверным результатом. Выход из этой ситуации прост — надо добавить в конец алгоритма подстановку («*»,«1»). Важно, чтобы эта подстановка стояла последней и применялась только в том случае, когда перед «*» нет никакой цифры. Окончательно самый простой алгоритм, решающий задачу, имеет вид:

(«0*»,«1»), («1*»,«2»), («2*»,«3»), («3*»,«4»), («4*»,«5»), («5*»,«6»), («6*»,«7»), («7*»,«8»), («8*»,«9»), («9*»,«0»), («*»,«1»).

Практический тур

7. Арифметика по-русски

Решение задачи предполагает аккуратное выполнение анализа заданной строки, реализацию указанного арифметического действия и перевод результата в словесную формулировку по правилам русского языка.

8. Колесо

По условию задачи необходимо в цикле рисовать окружность, центр которой смещается вдоль оси X, и несколько радиусов (спиц) в ней. Для создания эффекта вращения колеса углы между спицами и осями координат должны меняться. Чтобы не возникло «эффекта проскальзывания» колеса, надо согласовать его линейную скорость (смещение по оси X за один такт) и угловую скорость (изменение углов спиц). Это делается по простой физической формуле: *линейная скорость = радиус колеса • угловая скорость* или, иначе, *сдвиг по оси X за один такт = радиус колеса • приращение угла для каждой спицы* (заметим, что углы здесь измеряются в радианах). Для рисования спиц координаты концов отрезков вычисляются по простейшим тригонометрическим формулам:

$$\begin{cases} x_i = x_c + R\cos\left(\varphi + \dfrac{2\pi}{k} i\right); \\ y_i = y_c + R\sin\left(\varphi + \dfrac{2\pi}{k} i\right), \end{cases}$$

где (x_c, y_c) — координаты центра окружности, причем ордината постоянна, а абсцисса меняется каждый раз на одно и то же значение, k — количество спиц, i (от 1 до k) — номер спицы, а φ — угол поворота спиц, который на каждом шаге тоже изменяется на одну и ту же величину.

Заметим, что при решении данной задачи можно повысить качество изображения за счет использования режима работы графики с несколькими графическими страницами.

Областная олимпиада

Теоретический тур

9. «ЧЕГОНАДО»

В условии задачи явно не сказано, что $l < k$, поэтому следует сразу оговорить, что в случае некорректных данных ответом будет число k. В дальнейшем все рассуждение мы будем проводить при предположении, что $l < k$.

Существуют переборные способы решения данной задачи, один из которых можно организовать, например, так: составить вспомогательный алгоритм, который по заданной величине n вычисляет количество дней, за которое эта сумма станет меньше k, а затем с помощью какого-либо метода перебирать различные значения n и определить минимальную необходимую сумму денег.

Мы же для нахождения ответа будем проводить рассуждение с конца. Сначала решим следующую задачу: из какого минимального количества денег на следующий день может получиться x рублей после сдачи пустых бутылок. Для этого надо в предыдущий день иметь pk рублей, которые будут потрачены на покупку полных бутылок, и $x - pl$ рублей сдачи, причем, по условию задачи, обязательно должно быть выполнено неравенство $x - pl < k$. Следовательно, в предыдущий день должно иметься $x + p(k - l)$ рублей. Чтобы минимизировать эту величину (а менять в ней можно только p), надо взять в качестве p минимальное из чисел, при которых $x - pl < k$. Переписав данное неравенство в виде $p > (x - k)/l$, мы видим, что это число можно определить по формуле $p = \mathtt{int}((x - k)/l) + 1$. Итак, для получения x рублей надо в предыдущий день иметь не менее $x + (k - l) \cdot \mathtt{int}((x - k)/l + 1)$ рублей.

Из полученной формулы легко вытекает важный факт. Если $x < y$, то и в предыдущий день минимальное количество денег, из которых получаются данные суммы, удовлетворяет тому же соотношению. Отсюда уже следует, что в последний день у друзей должно быть ровно k рублей, и после четырехкратного применения полученной формулы можно

получить ответ задачи. Тогда алгоритм вычисления n будет иметь вид:

```
Алг A(вещ n,k,l)
арг k,l
рез n
нач цел i,p
   n:=k
   нц для i от 1 до 4
    p:=int((n-k)/l)+1
    n:=n+p*(k-l)
   кц
кон
```

10. Список

Задача имеет смысл, если количество элементов в списке конечно. Заметим, что если в списке есть цикл, то он — один (петля в конце списка). Вторым существенным замечанием является то, что длина списка может быть очень большой, поэтому использование массива для хранения всех просмотренных элементов недопустимо. Наконец, нам не требуется, чтобы алгоритм обнаруживал цикл сразу, как только появится элемент, содержащий адрес одного из предшествующих элементов.

Пожалуй, самым простым решением данной задачи является вложенный цикл. Запомним адрес первого элемента списка. У каждого нового элемента будем сравнивать имеющейся в нем адрес следующего элемента сначала с нулем (если таковой встретится, то это — конец списка, и цикла нет), а при ненулевом значении — с адресами всех уже пройденных элементов. Для этого, используя хранящийся у нас адрес первого элемента, заново каждый раз просматриваем их от начала списка до текущего элемента.

Однако у приведенного алгоритма имеются два существенных недостатка.

Во-первых, весь список должен быть доступен в течение всей работы, чтобы можно было просматривать его каждый раз заново. Данное требование не всегда выполнимо на практике.

Во-вторых, этот алгоритм имеет большую трудоемкость (и время выполнения), так как требует сравнения каждого элемента с каждым.

Существенно лучшим является алгоритм с двумя *указателями*. Первый указатель используется для обычного перехода от одного элемента списка к другому, при этом для обнаружения конца списка значение указываемого элемента обязательно сравнивается с нулем. Второй указатель содержит адрес одного из уже просмотренных элементов, но его значение меняется в два раза реже, чем у первого: в то время как при помощи первого указателя мы просматриваем два элемента, второй смещается только на один элемент. Каждый раз сравниваются значения указателей, и как только они совпадут, выдается сообщение о наличии цикла, а алгоритм заканчивается. Это обязательно произойдет, если цикл существует, так как, войдя в цикл, указатели из него уже не выйдут, а внутри цикла первый указатель на каждом шаге приближается ко второму на 1 — «догоняет» его.

Это быстрый алгоритм, так как количество сравнений адресов линейно зависит от длины списка. Но здесь, аналогично первому алгоритму, надо иметь возможность для повторного просмотра списка. Чтобы этого не требовалось, будем запоминать только один из уже просмотренных элементов: сначала 1-й, потом 2-й, 4-й, 8-й, 16-й и т. д. Переходя к очередному элементу списка, будем сравнивать его значение с запомненным. Поскольку шаг неограниченно растет, то обязательно наступит момент, когда очередной запомненный элемент будет элементом из цикла, а длина цикла окажется меньше расстояния до следующего элемента, который надо запомнить. При этом текущий элемент (после полного или частичного прохода по циклу) совпадет с запомненным.

В заключение отметим, что три описанные выше разновидности алгоритма не исчерпывают всех возможностей решения задачи, поиск которых оставим читателям.

11. УХТЫ-01

Требуемая программа для машины УХТЫ-01 (с комментариями) имеет вид:

```
Ввести a,b
z=F(a,a)     | z=0
e=F(z,a)     | e=1
x=F(e,a)     | x=1-1/a
i=F(x,e)     | i=1-1+1/a=1/a
x=F(b,i)     | x=1-b*a
y=F(x,e)     | y=1-1+b*a=a*b
Вывести y    | Вывод произведения
x=F(b,a)     | x=1-b/a
y=F(x,i)     | y=1-(1-b/a)*a=1-(a-b)
t=F(y,e)     | t=1-1+(a-b)=a-b
Вывести t    | Вывод разности
x=F(a,b)     | x=1-a/b
y=F(x,e)     | y=1-1+a/b=a/b
Вывести y    | Вывод частного
x=F(b,e)     | x=1-b
y=F(a,x)     | y=1-a/(1-b)
t=F(e,x)     | t=1-1/(1-b)
x=F(t,e)     | x=1-1+1/(1-b)=1/(1-b)
t=F(y,x)     | t=1-(1-a/(1-b))*(1-b)=1-(1-b-a)=a+b
Вывести t    | Вывод суммы
```

Необходимо отметить, что сумма $a + b$ в данной программе вычисляется при дополнительном условии: b не равно 1. Можно было бы это условие заменить на аналогичное (a не равно 1) или какое-то другое, но обойтись совсем без ограничений на допустимые значения аргументов, по-видимому, не удастся.

12. Лабиринт

Подходящей структурой данных для представления информации о лабиринте является квадратная таблица L, каждая клетка которой соответствует одной комнате. Размеры таблицы можно установить заведомо большими лабиринта, для чего достаточно подсчитать количество букв в исходной записи переходов. Значение $L(i, j)$ следует устанавливать так,

чтобы по нему можно было определить возможные переходы в соседние комнаты, — например, в виде четырехразрядного двоичного числа, где первый разряд равен 1, если из комнаты (i, j) возможен переход на север, второй разряд соответствует переходу на юг, третий — на запад, а четвертый — на восток. Вход в лабиринт находится в некоторой клетке (i_0, j_0); клад — в клетке (i_k, j_k). Далее вся имеющаяся о лабиринте информация представляется в виде *графа*, вершины которого — это клетки, а ребра — возможные переходы. Тогда для решения задачи достаточно применить, например, *волновой алгоритм*.

13. Дисперсия

Пусть $m = (a1 + a2 + \ldots + an) / n$. Тогда

$$\begin{aligned} d &= ((a1 - m)^2 + (a2 - m)^2 + \ldots + (an - m)^2) / n = \\ &= ((a1^2 + a2^2 + \ldots + an^2) - 2m(a1 + a2 + \ldots + an) + nm^2) / n = \\ &= ((a1^2 + a2^2 + \ldots + an^2) - 2nm^2 + nm^2) / n = \\ &= (a1^2 + a2^2 + \ldots + an^2) / n - m^2. \end{aligned}$$

Введем следующие обозначения:

$$Sn = a1^2 + a2^2 + \ldots + an;$$
$$Kn = a1^2 + a2^2 + \ldots + an^2.$$

Легко видеть, что

$$Sn = Sn - 1 + an; \quad S0 = 0;$$
$$Kn = Kn - 1 + an^2; \quad K0 = 0.$$

Отсюда мы получаем формулы для вычисления среднего значения и дисперсии после чтения n-го элемента последовательности:

$$m = Sn / n;$$
$$d = Kn / n - m^2.$$

На данных соотношениях можно построить искомый алгоритм:

```
Алг ДИСПЕРСИЯ
нач цел n, вещ S,K,a,m,d
   n:=0; S:=0; K:=0
   нц пока есть элементы
      ввод элемента a
      n:=n+1; S:=S+a; K:=K+a*a
      m:=S/n; d:=K/n-m*m
      вывод m,d
   кц
кон
```

14. Алгоритм

Легко видеть, что n последовательно принимает значения квадратов целых чисел от 0 до k^2 в соответствии с формулой $(i+1)^2 = i^2 + 2i + 1$.

Ответ: k^2.

15. Сапер

Обозначим числа, записанные в верхней строке, через a_i, а содержимое нижней строки будем кодировать числами 0 (нет мины) или 1 (есть мина) и обозначать через x_i. Тогда задача сводится к решению системы n линейных уравнений с n неизвестными:

$$\begin{cases} x_1 + x_2 = a_1; \\ x_1 + x_2 + x_3 = a_2; \\ x_2 + x_3 + x_4 = a_3; \\ \ldots \\ x_{n-2} + x_{n-1} + x_n = a_{n-1}; \\ x_{n-1} + x_n = a_n. \end{cases}$$

Существует целый ряд различных методов решения линейных систем уравнений, но наша система имеет специальный вид — это так называемая *трехдиагональная система*, и для ее решения можно применять специальные методы,

например, *метод прогонки* или *метод стрельбы* (см. решения задач 88/89(19) и 00/01(5)). Суть первого из них заключается в том, что мы считаем заданным значение первой переменной в качестве параметра, затем из первого уравнения выражаем x_2, из второго — x_3, ..., из предпоследнего — x_n и подставляем в последнее уравнение. При этом может возникнуть три случая:

- получится линейное уравнение, из которого мы находим значение x_1 и через него — значения всех остальных переменных;
- выяснится, что система несовместна;
- получится тождество, что означает наличие множества решений.

В последнем случае любое решение можно получить, взяв произвольное значение x_1 и вычислив по полученным формулам значение остальных неизвестных.

Теперь отметим, что в нашей задаче не только сама система имеет специальный вид, но и значения переменных могут принимать только значения 0 и 1, что позволяет модифицировать метод прогонки следующим образом: мы берем $x_1 = 0$ и, осуществляя прогонку, смотрим, найдено ли решение, а затем повторяем ту же процедуру при $x_1 = 1$. Это дает ответ на первый вопрос задачи. (Заметим, что при фиксированном значении x_1 остальные неизвестные определяются однозначно, поэтому всего решений возможно не более двух.)

Вместе с простейшими примерами данных (`1 1` или `1 2 2 2 1`, или `1 1 1 1 1`), при которых возможны два решения, мы здесь автоматически получили ответ и на второй вопрос задачи: возможно максимум два разных решения. Для ответа же на третий вопрос требуется более подробное исследование задачи. Отметим, что если нам известно значение двух соседних элементов нижней строки, то мы можем восстановить ее всю, применив тот же самый метод прогонки, но начиная не с края, а с середины массива X и идя в две стороны. Уже из этого факта следует, что при наличии в строке данных хотя бы одного числа 0 или 3 либо числа 2 с края

поля решений не может быть несколько, а существование решения легко проверить применением метода прогонки.

Следующее полезное наблюдение заключается в том, что мы всегда можем найти значение переменной $x_3 = a_2 - a_1$, а зная его — значение $x_6 = a_5 - a_4 + x_3$, и таким образом определить значения всех переменных с номерами вида $3k$. Действуя аналогично с другого края, мы определим значения переменных с номерами $n + 1 - 3k$. Отсюда следует, что задача не может иметь несколько решений, если n делится на 3 или дает в остатке от деления на 3 единицу, поскольку тогда мы получим два известных соседних элемента нижней строки. Тем самым мы обосновали, что несколько решений может быть только на поле размерами 2×2, 2×5, 2×8 и т. д., и при этом все числа верхней строки должны быть только 1 или 2.

Последнее соображение, которое позволит нам дать полное описание тех исходных данных, при которых возможны два решения, следующее: если в верхней строке $a_i = 1$ и $a_{i+1} = 2$, то $x_{i-1} = 0$ и $x_{i+2} = 1$, а если $a_i = 2$ и $a_{i+1} = 1$, то $x_{i-1} = 1$ и $x_{i+2} = 0$. Отсюда следует, что в верхней строке 1 может меняться на 2 или 2 на 1 только на местах с номерами $3k + 1$ и $3k + 2$, поскольку иначе мы снова получим две известные соседние клетки в строке мин и однозначно восстановим всю последовательность X. Другими словами, числа 2 в строке A могут встречаться только группами по 3, 6 и т. д. подряд, причем средние двойки из каждых трех имеют номер клетки, кратный 3.

Чтобы доказать, что любая последовательность, удовлетворяющая всем перечисленным выше требованиям, дает два разных решения задачи, проведем следующее простое рассуждение. Для строки из двух (5, 8, 11, ...) единиц решений будет два (одно — когда мины стоят на местах с номерами $3k + 1$ и второе — когда мины стоят на местах с номерами $3k + 2$). Кроме того, к любому из этих решений можно добавлять на любом месте с номером 3, 6, ... еще мины, каждая из которых приводит к появлению группы из трех двоек.

Практический тур

16. Уравнение по-польски

Известно, что *суперпозиция* (многократное применение) арифметических функций (а в исходном выражении встречаются только они) является дробно-рациональной функцией, т. е. имеет вид:

$$\frac{a + b \cdot X}{c + d \cdot X},$$

где a, b, c, d — целые числа. Тогда для решения задачи достаточно получить представление исходного выражения в указанном виде и решить соответствующее уравнение.

Вначале проведем обработку исходного выражения. Для преобразования выражения, заданного в обратной польской записи, воспользуемся стеком для хранения коэффициентов a, b, c и d.

При обработке входной строки возможны следующие случаи:

- если очередной элемент — число s, то заносим его в стек:

  `a:= s, b:=0, c:=1, d:=0;`
- если очередной элемент — переменная X, то заносим ее в стек:

  `a:= 0, b:=1, c:=1, d:=0;`
- если очередной элемент — операция, то для двух верхних элементов стека (удаляя их из стека) выполняем эту операцию (с сохранением дробно-рационального вида) и результат помещаем в стек.

Коэффициенты (k, l, m, n) результата операции

$$\frac{a + bX}{c + dX} \otimes \frac{e + fX}{g + hX} = \frac{k + lX}{m + nX}$$

обычно вычисляются *методом неопределенных коэффициентов*. (Под $\otimes$ здесь понимается любая арифметическая операция.)

На последнем шаге мы анализируем и решаем уравнение:

$$\frac{a + b \cdot X}{c + d \cdot X} = 0,$$

коэффициенты которого остаются в стеке после окончания обработки. При представлении результата в виде дроби числитель и знаменатель сокращаем на их наибольший общий делитель.

Укажем и еще одно возможное решение. Будем обрабатывать исходное выражение слева направо так, как будто мы его вычисляем, записывая исходные числа и промежуточные результаты в стек (см. задачу 91/92(3)). Если операндами являются числа, то производим с ними указанное действие. Если же одним из операндов окажется неизвестное (например, $x + 5$), то обозначаем это выражение новым неизвестным ($y = x + 5$) и продолжаем обработку. В итоге у нас останется лишь одно элементарное (содержащее одно действие) арифметическое выражение с одним неизвестным. Так как значение этого выражения должно равняться нулю, то это неизвестное найти просто. Все остальные неизвестные, которые были введены в процессе обработки исходного выражения, могут быть вычислены в обратной последовательности путем решения простейших арифметических уравнений.

17. Шахматы

Теоретически решить данную задачу можно, разработав способ кодировки позиции, построив граф позиций и найдя кратчайший путь к выигрышу. Однако на этом пути есть много проблем. Реально же в задаче подразумевалась аккуратная реализация *линейного мата* двумя ладьями. При этом надо следить, чтобы чужой король не мог побить какую-нибудь из ладей.

Олимпиады 1995/96 учебного года

Городская олимпиада

Теоретический тур

1. Хорошая таблица

Определим вид «хорошей» таблицы. Из условия видно, что «шагать» вдоль пути можно только вправо или вниз; в противном случае путь будет содержать больше чем $N + M - 1$ клеток. Отсюда следует, что каждый следующий шаг пути переходит на следующую *побочную диагональ* (множество клеток, одинаково удаленных от начала). Таким образом,

a	*b*	*c*	*d*	*e*
b	*c*	*d*	*e*	*f*
c	*d*	*e*	*f*	*g*
d	*e*	*f*	*g*	*h*

Рис. 42

любой путь содержит по одной клетке каждой побочной диагонали. Нетрудно видеть, что сумма чисел в клетках каждого пути в таблице будет одна и та же только тогда, когда на любой побочной диагонали все числа равны (на рис. 42 клетки побочных диагоналей помечены одинаковыми буквами).

Исходя из этих соображений, процедура, проверяющая является ли таблица «хорошей», выглядит следующим образом:

```
Procedure table(a:mass;var otv:string);
begin
  otv:="таблица хорошая"
  for i:=2 to max(m,n) do
    for j:=2 to i do
      if a[i-j+1, j]<>a[i,1]
        then otv:="таблица не хорошая"
end;
```

2. Краевая перфорация

В каждом вертикальном ряду возможны три вида прорезей: «нулевая» (отсутствие прорези), одинарная (захватывающая одно отверстие) и двойная (захватывающая два отверстия). Таким образом, набор прорезей на каждой карте ставится в соответствие троичному числу. Заметим, что если одно число X поразрядно не больше Y, то карточка, соответствующая Y, всегда провалится, если проваливается карточка, соответствующая X, поэтому оба эти числа не могут одновременно участвовать в кодировке.

При использовании двух вертикальных рядов количество разрядов в этих числах равно двум. Разобьем все двузначные троичные числа на группы: (00, 01, 02, 12, 22), (10, 20, 21), (11). Заметим, что из каждой группы для кодировки можно выбрать не более одного числа (по причине, указанной выше). Такими числами могут быть 02, 20 и 11. Таким образом, два вертикальных ряда обеспечивают кодирование не более трех признаков.

Трехзначные троичные цифры можно разбить на следующие 7 групп, из которых для кодировки можно взять не более одного числа: (000, 001, 002, 012, 022, 122, 222), (010, 020, 021, 121, 221), (011, 111, 112, 212), (100, 101, 102, 202), (110, 120), (200, 201, 211), (210, 220). Очевидно, что три вертикальных ряда обеспечивают кодирование не более семи признаков, т. е. их не хватит для кодировки десяти цифр.

Используя четыре вертикальных ряда, десятичные цифры можно закодировать, например, следующим образом: 0 — 2210; 1 — 2201; 2 — 2120; 3 — 2102; 4 — 1022; 5 — 1220; 6 — 2012; 7 — 2021; 8 — 0221; 9 — 1202. Десять карт, соответствующих этим цифрам, представлены на рис. 43.

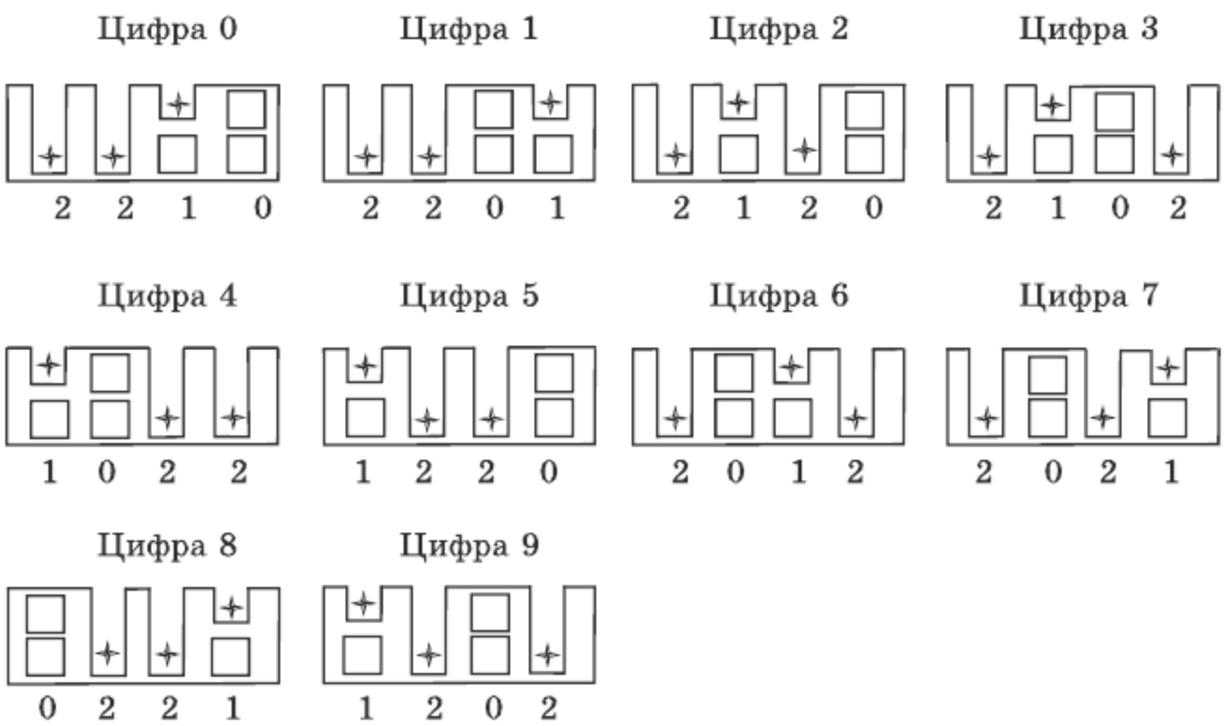


Рис. 43

Наборы штырей, позволяющие отбирать карты, указаны в нижних позициях прорезей, т. е. в точности соответствуют коду каждой цифры. Легко убедиться, что такая кодировка удовлетворяет условиям задачи.

Для оценки количества признаков, которые можно закодировать с помощью четырех вертикальных рядов, рассмотрим 7 групп, которые мы использовали в предыдущем случае. Добавим впереди каждого числа цифру 0 — получим 7 групп четырехзначных чисел, начинающихся с нуля. Затем вместо нуля добавим цифру 1 и получим 7 групп четырехзначных чисел, начинающихся с единицы. Аналогично, добавив цифру 2, получим 7 групп четырехзначных чисел, начинающихся с двойки. Всего получилась 21 группа, где из каждой можно использовать не более одного кода, — значит, более 21 признака закодировать таким способом невозможно.

Таким образом, для кодировки 26 букв латинского алфавита потребуется не менее пяти вертикальных рядов отверстий. При использовании вышеприведенных обозначений это может быть, например, такой набор кодов:

(12200), (21200), (22100), (10220), (20120), (20210), (10022), (20012), (20021), (10202), (20102), (20201), (01202), (02102), (02201), (01022), (02012), (02021), (12020), (21020), (22010), (12002), (21002), (22001), (00122), (00212).

3. Ковер и моль

Будем называть дырки, проеденные молью, «точками», а ковер — «исходным прямоугольником». Добавим в множество точек две, от которых решение не зависит, — одну, принадлежащую левой стороне исходного прямоугольника, а другую — правой.

Отсортируем точки по возрастанию абсцисс. Добавленные точки при этом окажутся первой и последней, что даст возможность корректно начать и завершить алгоритм.

Заведем список M, в котором будем хранить множество отрезков — допустимых левых сторон искомых прямоугольников. Каждый отрезок будет задаваться тремя параметрами: ординатами начала и конца, а также абсциссой.

Сначала список M — пустой. Далее мы обрабатываем все точки в порядке возрастания абсцисс и корректируем M. Каждая точка разбивает отрезок на два, если ее ордината находится строго внутри данного отрезка; в противном случае отрезок остается неизменным. Кроме того, после просмотра всех отрезков нужно добавить отрезок во всю высоту прямоугольника с координатой x, равной абсциссе обрабатываемой точки.

В итоге мы получаем следующий алгоритм решения задачи.

1. Сортируем точки по возрастанию абсцисс.
2. Для каждой точки выполняем следующее:
   1) перебираем все отрезки из списка и запоминаем прямоугольник с максимальной площадью (левая сторона которого — рассматриваемый отрезок, а правая граница задается абсциссой текущей точки);
   2) перебираем все отрезки, и если точка находится строго внутри отрезка, то разбиваем его на два и заносим их в M вместо исходного, иначе — оставляем отрезок неизменным;
   3) добавляем в список отрезок высотой с исходный прямоугольник с абсциссой, равной абсциссе текущей точки.

Максимальный среди всех прямоугольников, найденных в пункте 2(1), является искомым.

4. Кольцевая дорога

Докажем сначала, что существует бензоколонка A, бензина в которой хватит для проезда до следующей бензоколонки $Б$. Если бы это было не так, то сумма участков, которые можно проехать от всех бензоколонок, оказалась бы меньше полного круга, что противоречит условиям задачи.

Теперь ясно, что, доехав до следующей колонки, машина всегда может взять имеющийся там бензин и продолжать движение. Поэтому ситуация эквивалентна такой, когда бензин из колонки *Б* уже находится в *А*. Значит, исходная задача сводится к задаче с меньшим количеством бензоколонок.

Продолжая решать эту задачу рекурсивно, мы придем к ситуации, когда на кольце останется одна колонка, «содержащая в себе весь бензин». Именно с этой колонки и надо начинать движение.

Указанный алгоритм можно описать более формально:

```
Procedure Benzo;
begin
  if в списке всего одна колонка
   then печатаем ее исходный номер
   else
    begin
      находим колонку i, для которой отношение
      находящегося там бензина к расстоянию
      до следующей колонки максимально (от этой
      колонки можно гарантированно доехать до
      следующей).
      Увеличиваем количество бензина в колонке i
      на количество бензина в следующей колонке.
      Исключаем следующую колонку из списков.
      Соответственно увеличиваем расстояние
      до следующей колонки
      Benzo
   end;
end;
```

Для описания бензоколонок, стоящих вдоль дороги, можно использовать *циклический список* (сначала в этом списке стоят числа от 1 до n). Информацию о количестве бензина в каждой из колонок и о расстояниях до следующей колонки также можно хранить в циклических списках.

5. Шайка

Структуру шайки удобно изобразить в виде *бинарного дерева*, например как на рис. 44.

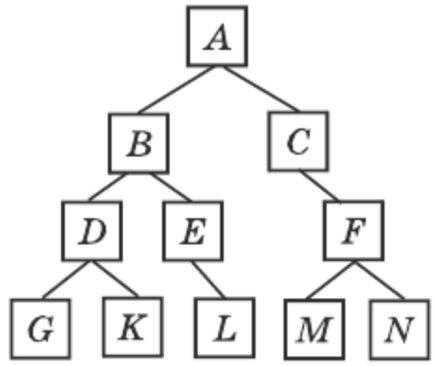


Рис. 44

Следующий рекурсивный алгоритм, примененный главарем *A*, решает пункт (*A*) задачи.

1. Позвонить подручным (если они есть) и дать им указания выполнить тот же самый алгоритм.
2. Перезвонить подручным и узнать результаты.
3. Просуммировать результаты и добавить к сумме 1 (себя ведь тоже надо сосчитать!).
4. Сообщить результат начальнику, когда тот перезвонит.

В случае (Б) главарь может узнать численность шайки, дав поручение подручным применить вышеописанный алгоритм, т. е. сосчитать количество элементов в поддеревьях, а затем сообщить результаты одному из подручных для окончательного суммирования.

6. Робот

В результате выполнения алгоритма Робот закрасит все клетки тупика, кроме последней.

7. Кинотеатр

Рассмотрим типовой процесс продажи билетов, происходящий в кассе кинотеатра. Вообще говоря, это процесс не такой уж простой, если учитывать все различные экономические тонкости нашей повседневной жизни, поэтому мы будем сначала строить упрощенную модель, а затем постепенно

снимать ограничения, пока не учтем все случаи, происходящие в реальной жизни.

В любом случае процесс продажи билетов начинается с того, что перед кассиром лежит план кинозала, в котором крестиком отмечается проданное место, когда кассир выдает соответствующий этому месту билет. Следовательно, если представить план кинозала в виде прямоугольной таблицы A из m строк (рядов) и n столбцов (мест в каждом ряду), то естественно закодировать цифрой 0 каждое свободное место, а 1 — уже проданное. При этом мы предполагаем, что все ряды содержат одинаковое число мест, хотя в действительности это не всегда так. Тем не менее на первом этапе построения модели мы ограничимся этим случаем.

Далее нам необходимы данные для расчета дневной выручки. Очевидно, нам нужны данные о количестве сеансов в кинотеатре и стоимости билетов в каждом ряду, поскольку, как мы знаем, первые и последние ряды обычно дешевле, чем средние. Пусть k — количество сеансов, $P[1...k]$ — таблица, каждый элемент которой содержит значение выручки за соответствующий сеанс, C — таблица, каждый элемент которой содержит цену билета в соответствующем ряду, а S — дневная выручка. Заметим, что здесь мы снова сделали упрощение, предположив, что цена билета не меняется от сеанса к сеансу, хотя в действительности билеты на утренние сеансы дешевле, чем на вечерние, а кроме того существует еще разница в цене билета в зависимости от дня недели.

Таким образом, в первом приближении мы приходим к следующей модели.

- $A[1...m, 1...n]$ — таблица, отражающая заполнение зала на данном сеансе, а именно:

– $A[i, j] = 0$, если место свободно;

– $A[i, j] = 1$, если место продано,

которая заполняется после каждого сеанса;

- $C[1...m]$ — таблица, где $C[i]$ — цена билета в соответствующем ряду;
- $P[1...k]$ — таблица, где $P[i]$ — выручка от продажи билетов на i-м сеансе;
- S — общая сумма за день.

Конечно, можно было бы сразу накапливать выручку в ячейке S и обойтись без P, но иногда полезно проанализировать данные по сеансам, дням и т. д.

Это, конечно, далеко не полное построение модели, поскольку оно учитывает не все особенности ценовой политики кинотеатра. Решение можно продолжать практически бесконечно, что авторы и предоставляют читателям.

Практический тур

8. Как из мухи сделать слона

Сформулируем задачу в терминах *теории графов*. Пусть слова — это вершины графа. Две вершины соединены ребром, если соответствующие им слова отличаются на одну букву. Тогда для решения задачи достаточно использовать *волновой алгоритм* (*обход графа в ширину*).

9. Перестановка книг

А) При решении достаточно аккуратно запрограммировать описанный в условии задачи алгоритм. Информацию о текущем расположении книг можно хранить в массиве или в списке. Использование списка позволит быстро совершать операции, описанные в условии задачи — достаточно в нужных местах поменять соответствующие указатели.

Б) Простейшим алгоритмом, решающим поставленную задачу, является такой, при котором каждая книга с номером i ($i = 1, 2, \ldots, N$) в одиночку ставится на свое место. Однако ясно, что по количеству перестановок такой алгоритм не оптимален. Несколько улучшить алгоритм можно, заметив, что книги, имеющие последовательные номера и стоящие рядом, можно переставлять группой.

Опишем алгоритм построения решения, дающего лучший результат, т. е. минимальное число троек (перестановок).

Предположим, что какое-то множество из k книг стоит так, что каждая из книг этого множества, имеющая больший номер, стоит левее каждой, имеющей меньший номер. Назовем такую группу книг «*кликой*». Менее чем за $k - 1$ операцию поставить эти книги в нужном порядке невозможно, так как любые две последовательные книги в этой группе хотя бы раз должны попасть в разные группы переставляе-

мых книг. Таким образом, «клика» с наибольшим количеством элементов («*максимальная клика*») дает оценку снизу на количество операций. Найти «максимальную клику» несложно. В нашей задаче это эквивалентно выделению из последовательности номеров книг максимальной убывающей подпоследовательности, что решается известным алгоритмом, использующим *метод динамического программирования* (см. решение задачи 97/98(4)). А при сортировке книг на каждом шаге надо совершать такую операцию, при которой размер наибольшей «клики» уменьшается.

Если же «максимальных клик» несколько, то рекомендуется выбрать такую операцию, при которой уменьшается размер наибольшего количества «максимальных клик». Это достигается перебором всевозможных операций, количество которых не превышает N^2. Обратим внимание, что для поиска оптимальной сортировки нам приходится выполнять алгоритм гораздо более трудоемкий, чем сама сортировка!

Областная олимпиада

Теоретический тур

10. Автобус

Пусть $q(k)$ — максимальное число пассажиров, одновременно ехавших в автобусе с первой (конечной) по k-ю остановку, а $s(k)$ — количество пассажиров, ехавших с $(k-1)$-й по k-ю остановку. Попытаемся получить рекуррентные формулы для вычисления $q(k)$ и $s(k)$.

Достаточно очевидно, что

$$q(2) = \sum_{i=2}^{n} p(1, i); \quad s(2) = \sum_{i=2}^{n} p(1, i).$$

Пусть известны значения $q(k)$ и $s(k)$ для некоторого k. Тогда для $q(k+1)$ и $s(k+1)$ будут выполняться следующие формулы:

$$s(k+1) = s(k) - \sum_{i=1}^{k-1} p(i, k) + \sum_{i=k+1}^{n} p(k, i);$$

$$q(k+1) = \max(q(k), s(k+1)).$$

Таким образом, последовательно вычисляя s и q, мы найдем $q(n)$, т. е. максимальное число пассажиров, ехавших одновременно с первой по n-ю остановку.

11. Сеть

Покажем, как можно решить задачу за 8 тактов. Будем считать, что компьютеры помещены в таблицу 4×4 и обмен информацией разрешается между компьютерами, клетки которых имеют общую сторону.

Соберем всю информацию на четырех центральных компьютерах за два такта (рис. 45).

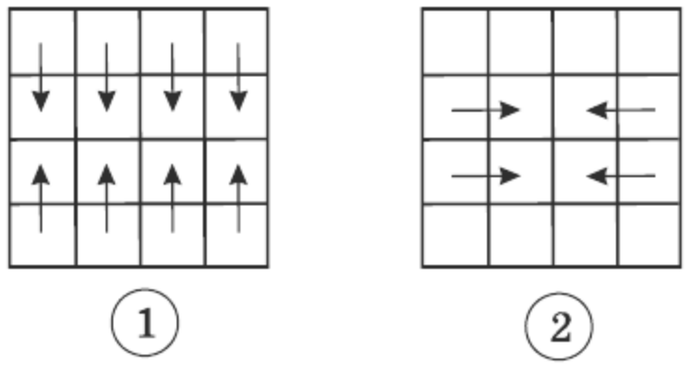


Рис. 45

Затем еще за четыре такта добьемся того, чтобы в каждом из четырех центральных компьютеров была вся информация (рис. 46).

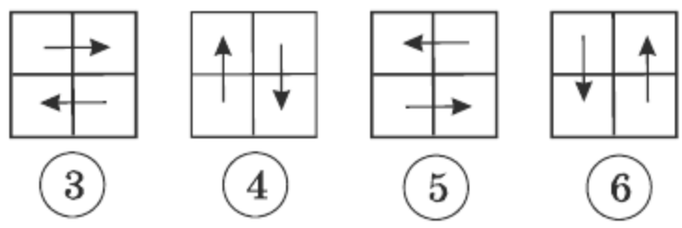


Рис. 46

Наконец, из центральных компьютеров разошлем информацию в оставшиеся компьютеры в порядке, обратном к первым двум тактам (7, 8).

Решение за меньшее количество тактов невозможно, хотя доказать этот факт не очень просто.

12. Прожекторы

Рассмотрим два прожектора с координатами (x_1, y_1) и (x_2, y_2). Очевидно, если $x_1 < x_2$ и $y_1 < y_2$, то прожектор с координатами (x_2, y_2) можно выключить, и при этом освещенная часть плоскости не уменьшится.

Тогда алгоритм, выключающий максимальное количество прожекторов, будет выглядеть так:

1. Определим координаты прожекторов с минимальной ординатой (X_k, Y_k) и минимальной абсциссой (X_l, Y_l).
2. Выключим все прожекторы, у которых $x \geqslant X_l$ и $y \geqslant Y_k$. Прожекторы с координатами (X_k, Y_k) и (X_l, Y_l) мы не выключаем, но в дальнейшем уже не рассматриваем.
3. Сдвинем систему координат на вектор (X_l, Y_k).
4. Если новая система координат не содержит ни одного прожектора, то задача решена, иначе повторяем рассуждения для новой системы координат.

Заметим, что приведенный алгоритм представляет собой алгоритм нахождения *множества минимальных элементов в частично упорядоченном множестве* (двумерных векторов).

13. Ранг числа

Сначала докажем по индукции, что функция $R(n)$ не убывает.

Пусть $R(n)$ не убывает на промежутке от 0 до $k-1$. Докажем, что $R(k) \geqslant R(k-1)$. По определению $R(k) = R(k - S(k)) + 1$, а значит $R(k-1) = R(k - 1 - S(k-1)) + 1$.

Если последняя цифра числа $k-1$ не равна 9, то выражение $k - S(k)$ равно $k - 1 - S(k-1)$. Если последняя цифра числа $k-1$ есть 9, а следовательно, сумма цифр $S(k)$ стала меньше $S(k-1)$, то выражение $k - S(k)$ больше, чем $k - 1 - S(k-1)$. Таким образом, по предположению индукции, $R(k - S(k)) \geqslant R(k - 1 - S(k-1))$. Отсюда следует, что $R(k) \geqslant R(k-1)$.

Так как функция $R(n)$ — монотонная, то можно применить *метод дихотомии* для поиска минимального n, при котором $R(n)$ будет равняться заданному числу s. Положим $a = 0$, $b = s \cdot 10$. Далее, удваивая b, найдем такое его значение, при котором $R(b) \geqslant s$. Затем в цикле ищем значение $R(t)$,

где $t = \frac{a+b}{2}$. Если $R(t) \geqslant s$, то заменяем b на $t - 1$, иначе заменяем a на $t + 1$. Цикл прерывается, как только a становится больше b.

В результате a будет минимальным натуральным числом, имеющим заданный ранг R.

14. Сломанный калькулятор

Для решения данной задачи достаточно воспользоваться формулой:

$$a \cdot b = \frac{(a+b)^2 - a^2 - b^2}{2}.$$

15. НОК

Воспользуемся известной формулой:

$$\text{НОК}\,(a_1, a_2) = \frac{a_1 \cdot a_2}{\text{НОД}\,(a_1, a_2)}.$$

НОД двух чисел можно найти с помощью *алгоритма Евклида*. Заменяя любые два числа на их НОК, мы уменьшаем общее количество чисел, но не изменяем общего НОК. Последовательно применяем такие действия, пока не останется одно искомое число.

Практический тур

16. Расшифровка текста

Достаточно просто, перебирая сдвиг от 1 до 32, получить различные варианты расшифровок исходного текста. Главная проблема — определить, какая из расшифровок является исходным текстом, т.е. осмысленным текстом на русском языке. Существуют различные способы решения этой задачи.

1. Можно определить частоту вхождения в различные расшифровки каждой буквы и сравнить результат с имеющейся статистикой.

2. Можно определить некоторые правила русского языка. Например, если слово состоит более чем из одной буквы, то в нем должна присутствовать гласная буква; слово не может начинаться с символов «ъ», «ь», «ы» и т. п.
3. Можно осуществить проверку однобуквенных слов. Если слово состоит из одной буквы, то это могут быть только буквы «а», «б», «в», «и», «к», «о», «с», «у», «я».

По совокупности выполнения этих правил можно делать вывод о том, какой из вариантов расшифровки правильный.

17. Фигурки из правильных треугольников

Сначала определим способ представления фигурок. Информацию о них можно хранить в двумерном массиве A, размер которого достаточно задать равным $\frac{n}{2} \times n$. В нем $a[i, j] = 1$, если треугольник входит в состав фигуры, или $a[i, j] = 0$ — в противном случае. Например для $n = 4$ одна из фигур выглядит следующим образом (рис. 47).

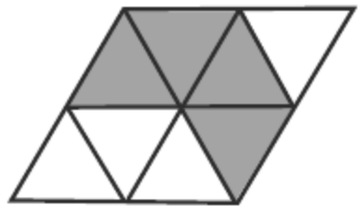


Рис. 47

Ей соответствует массив A:

1	1	1	0
0	0	0	1

Для перечисления всех фигурок достаточно использовать *алгоритм исчерпывающего поиска* (*backtracking*), который на каждом шаге добавляет одну единицу в уже построенный массив построчно сверху вниз, слева направо, пока общее

количество единиц не станет равным n. При этом добавление единицы не должно нарушать связности. Это обеспечивается следующим правилом: добавляемая единица может следовать непосредственно за единицей в той же строке или стоять на четном месте в строке под единицей предыдущей строки.

При получении очередной фигуры необходимо сравнить ее со всеми ранее построенными фигурами (эталонами), информация о которых должна храниться в списке.

Заметим, что, по правилу построения, матрица очередной фигуры, как и матрица эталона, содержит единицу в первой позиции первой строки. Для сравнения с эталоном фигуру необходимо дважды повернуть на 60°, после каждого поворота сдвинуть вверх и влево до совмещения левой верхней единицы с первой позицией матрицы и после каждого сдвига убедиться, что их матрицы не совпадают. Для завершения сравнения фигуру необходимо симметрично отразить относительно любой оси и повторить описанные операции. Все перечисленные действия являются несложными, их реализацию читатели могут выполнить самостоятельно.

Олимпиады 1996/97 учебного года

Городская олимпиада

Теоретический тур

1. Машины

Рассмотрим дорогу от Ярославля до Рыбинска как числовой отрезок от 0 до L и выделим на нем точки, в которых находятся все машины в данный момент времени t: $v_i(t - a_i)$; $L - w_i(t - b_i)$. Пометим точки из диапазона $[0,L]$, в которых находятся машины, едущие из Ярославля в Рыбинск, единицами, а точки, в которых находятся машины, едущие из Рыбинска в Ярославль, — минус единицами. Заметим, что некоторые точки могут оказаться помеченными и 1, и −1. Также выделим концы отрезка и пометим их нулями.

Требуемый отрезок можно найти по следующему алгоритму.

1. От Ярославля откладываем отрезок в один километр в направлении Рыбинска. Вычисляем сумму чисел, находящихся в выделенных точках полученного отрезка, и обозначаем ее как S.
2. Если $S = 0$, то переходим к п. 5.
3. Сдвигаем данный отрезок в сторону Рыбинска на минимальное положительное расстояние, при котором изменится множество покрываемых им точек. Если этого сделать невозможно, то выводим ответ: «Такого отрезка нет» и завершаем алгоритм.
4. Определяем сумму чисел, находящихся в выделенных точках полученного отрезка. Для этого из S вычитаем сумму пометок точек, которые перестали покрываться отрезком, и добавляем сумму пометок точек, которые вошли в отрезок. Переходим к п. 2.
5. Выводим ответ и завершаем алгоритм.

Нетрудно показать, основываясь на *принципе дискретной непрерывности*, что отрицательного ответа в данной задаче не может быть; впрочем, это нам и не требовалось.

2. Смежные перестановки

Обозначим через $\{n\}$ множество чисел от 1 до n.

Докажем методом математической индукции, что для любого n можно расположить все перестановки $\{n\}$ по кругу так, чтобы рядом стояли только смежные.

Для $n = 2$ решение очевидно: перестановки (1, 2) и (2, 1) являются смежными и их можно расположить по кругу.

Предположим, что для $\{n - 1\}$ мы можем расположить все перестановки по кругу так, чтобы рядом стояли только смежные. Тогда опишем алгоритм решения задачи для $\{n\}$.

Заметим, что перестановка чисел $\{n\}$ получается из перестановки $\{n - 1\}$ добавлением числа n, которое можно вставить в начало, в конец или между любыми элементами.

Расположим перестановки $\{n - 1\}$ (их всего $(n - 1)!$) в одну строку по порядку, — это можно сделать согласно индукционному предположению, — начиная с перестановки

(1, 2, ..., n − 1). Допишем к первой перестановке справа число n и будем переставлять его с ближайшим левым соседом до тех пор, пока оно не окажется крайним левым. Теперь справа от числа n находится перестановка (1, 2, ..., n − 1). Заменим ее на следующую и будем переставлять число n с ближайшим правым соседом до тех пор, пока оно не окажется крайним правым. Продолжая таким образом, мы получим $n!$ смежных перестановок, и все они различны. Значит, получены все перестановки $\{n\}$. Так как количество перестановок — четное число, то в последней перестановке число n будет крайним правым; следовательно, первая и последняя перестановки — тоже смежные.

Приведенное доказательство включает в себя также и алгоритм построения всех требуемых перестановок.

Рассмотрим пример построения смежных перестановок длины 3 и 4.

Из перестановки (1, 2) описанным способом можно получить перестановки: (1, 2, **3**), (1, **3**, 2), (**3**, 1, 2). Из перестановки (2, 1) получаем (**3**, 2, 1), (2, **3**, 1), (2, 1, **3**). Таким образом, мы получили все шесть перестановок длины 3, причем расположенные так, что рядом стоят только смежные.

Аналогичным способом из этих перестановок можно получить перестановки длины 4:

(1, 2, 3, **4**), (1, 2, **4**, 3), (1, **4**, 2, 3), (**4**, 1, 2, 3), (**4**, 1, 3, 2), (1, **4**, 3, 2), (1, 3, **4**, 2), (1, 3, 2, **4**), (3, 1, 2, **4**), (3, 1, **4**, 2), (3, **4**, 1, 2), (**4**, 3, 1, 2), (**4**, 3, 2, 1), (3, **4**, 2, 1), (3, 2, **4**, 1), (3, 2, 1, **4**), (2, 3, 1, **4**), (2, 3, **4**, 1), (2, **4**, 3, 1), (**4**, 2, 3, 1), (**4**, 2, 1, 3), (2, **4**, 1, 3), (2, 1, **4**, 3), (2, 1, 3, **4**).

3. Генеалогическое древо

Очевидно, что решение задачи (Б) является решением и задачи (А).

Для нумерации всех, кто попал в список, можно использовать рекурсивный алгоритм, который для каждого элемента нумерует его предков по материнской линии, заканчивая на номере k, потом нумерует сам элемент числом $k + 1$, а потом всех предков по отцовской линии, начиная с номера $k + 2$.

Процедура, реализующая описанный алгоритм, будет выглядеть следующим образом:

```
procedure numerate(n:integer; x:tree;
                   var k:integer);
  var m:integer;
  begin
    if x=nil then k:=n-1
     else
      begin
        numerate(n,x^.left,m);
        x^.key:=m+1;
        numerate(m+2,x^.right,k);
      end;
  end.
```

Примерно те же действия можно описать и нерекурсивным образом. Чтобы поставить номер в вершину, являющуюся потомком по отцовской линии своего предка, мы должны подсчитать количество элементов в левом поддереве данной вершины, добавить к номеру предка данной вершины это число и плюс еще единичку и присвоить данной вершине полученный номер. При нумерации самого Остапа Бендера мы считаем, что номер его предка равен 0. Чтобы поставить номер в вершину, являющуюся потомком по материнской линии своего предка, мы должны подсчитать количество элементов в ее правом поддереве, вычесть из номера предка данной вершины это число и еще единичку и присвоить данной вершине полученный номер. Процесс нумерации происходит при *обходе дерева в ширину*, т. е. сначала нумеруется корень дерева (Остап Бендер), затем номера получают его мать и отец и т. д.

4. Гири

Пусть массив A содержит вес данных гирь. В массив B будем заносить все значения веса, которые можно получить с помощью этих гирь, по следующему алгоритму:

1. Упорядочить массив A.
2. $B[1] = A[1]$; $n = 2$.
3. Занести в B значения $A[n]$ и суммы $A[n]$ со всеми элементами B.
4. Упорядочить B; $n = n + 1$.

5. Удалить из B все $B[i]$, для которых $i > k$.
6. Если $A[n]$ существует и $A[n] < B[k]$, то перейти к п. 3, иначе вывести массив B.

5. Рациональный умножитель

Представим данное число n в виде произведения простых сомножителей. Заметим, что операция умножения на $\frac{p}{q}$ равносильна замене простого сомножителя q на сомножитель p. Чтобы получить цепочку максимальной длины, необходимо заменять сомножитель ближайшим к нему меньшим простым числом. Действительно, если мы заменяем q на p и при этом между ними есть еще одно простое число r, то можно построить цепочку длиннее, а именно произвести два действия: умножить на $\frac{r}{q}$ и на $\frac{p}{r}$.

Требуемый алгоритм:

1. Разложить n на простые сомножители.
2. Расположить сомножители в порядке возрастания:

$$n = a1 \cdot a2 \cdot \ldots \cdot ak; \qquad n = n \cdot \frac{p}{ak},$$

где p — ближайшее к ak меньшее простое число.

3. Если $n = 1$, то цепочка построена; иначе возвратиться к п. 1.

6. Алгоритм

Данный фрагмент программы реализует алгоритм поиска простых чисел, известный как «*решето Эратосфена*», т. е. $a[i] = 0$, если i — простое число.

7. Три круга

Сначала сделаем несколько простых замечаний.

Так как круги покрывают треугольник, каждый из этих кругов пересекается хотя бы с одним другим кругом. При этом либо существует точка, принадлежащая всем трем кругам, либо нет.

Если такая точка X существует, то существует и точка O, принадлежащая всем трем кругам и лежащая в треугольнике. Тогда легко понять, что треугольники OAB, OAC и OBC покрыты кругами, а их объединение дает исходный треугольник ABC.

Если не существует точки, принадлежащей всем трем кругам, то существует круг, пересекающийся с двумя другими, например, круг 1 (с центром A) с кругами 2 и 3. При этом либо круг 1 содержит круг 2, либо обе точки пересечения окружностей 1 и 2 лежат вне треугольника. (Если бы точка пересечения окружностей 1 и 2 лежала бы в треугольнике, то некоторая ее окрестность не была бы покрыта.) Аналогичная ситуация — для пары 1, 3.

Теперь опишем алгоритм.

Рассматриваются три неравенства:

$$R_1 + R_2 \geqslant AB;$$
$$R_1 + R_3 \geqslant AC;$$
$$R_2 + R_3 \geqslant BC.$$

Круги покрывают треугольник тогда и только тогда, когда либо выполняются все три этих неравенства (это соответствует первому рассмотренному случаю), либо выполнены два из трех неравенств (например, первые два) и точки пересечения окружностей 1 и 2 лежат вне треугольника, либо круг 1 содержит круг 2, а также аналогичные условия для кругов 1 и 3.

Примечание: нахождение точек пересечения окружностей сводится к решению системы уравнений для окружностей; треугольник определяется тремя прямыми, и его внутренние точки удовлетворяют системе из трех линейных неравенств (см. 87/88(5)).

Практический тур

8. Поиск

Задача сводится к перебору всех слов и к проверке, подходит ли каждое слово по маске (шаблону). Это можно сделать с помощью следующей рекурсивной функции.

Сравниваем символы шаблона и слова. Если текущий символ в шаблоне — «*», то перебираем оставшиеся части слова начиная с текущей буквы и рекрсивно вызываем ту же самую функцию проверки (передавая в нее остаток слова и остаток шаблона). Если хотя бы для одного «хвоста» функция вернет значение «истина», то слово подходит по шаблону.

Если же текущий символ в шаблоне — «?», то переходим к следующему символу в шаблоне и в слове. В противном случае смотрим, равны ли текущие символы в слове и в шаблоне. Если они равны, то переходим к следующим символам, иначе возвращаем значение «ложь».

Кроме того, наша функция должна возвращать значение «ложь», если есть символы (отличные от «*») в шаблоне, а символы в слове закончились, и наоборот.

9. Табло

Существует несколько способов решения данной задачи. Опишем один из них.

Состояние каждой лампочки может измениться при нажатии на эту лампочку или на любую соседнюю. Заметим, что двойное нажатие на одну и ту же лампочку-кнопку (необязательно последовательное) состояния системы не меняет. Кроме того, для конечного результата порядок нажатия кнопок не имеет значения.

Обозначим как $x_{i,j}$ переменную, которая будет указывать, нужно ли нажимать кнопку (i, j). Если $x_{i,j} = 1$, то нужно; если $x_{i,j} = 0$, то нет. Тогда наша задача сводится к решению системы линейных уравнений в числах 0 или 1. (В таком случае говорят, что мы работаем «в поле Z_2». Все законы в этом поле такие же, как всегда, кроме операции сложения, где $1 + 1 = 0$.)

0	1	0
0	1	0
1	1	0

Рис. 48

Например, для табло на рис. 48 система имеет вид:

$$\begin{cases} x_{11} + x_{12} + x_{21} + x_{22} = 0; \\ x_{11} + x_{12} + x_{21} + x_{22} + x_{13} + x_{23} = 1; \\ x_{12} + x_{13} + x_{22} + x_{23} = 0; \\ x_{11} + x_{12} + x_{21} + x_{22} + x_{31} + x_{32} = 0; \\ x_{11} + x_{12} + x_{13} + x_{21} + x_{22} + x_{23} + x_{31} + x_{32} + x_{33} = 1; \\ x_{12} + x_{13} + x_{22} + x_{23} + x_{32} + x_{33} = 0; \\ x_{21} + x_{22} + x_{31} + x_{32} = 1; \\ x_{21} + x_{22} + x_{23} + x_{31} + x_{32} + x_{33} = 1; \\ x_{22} + x_{23} + x_{32} + x_{33} = 0. \end{cases}$$

Решив полученную систему, получим, что $x_{12} = 1$, $x_{13} = 1$, $x_{21} = 1$, а остальные переменные равны нулю. Таким образом, чтобы выключить все табло, достаточно нажать на три кнопки: (1, 2), (1, 3) и (2, 1).

Областная олимпиада

10. МПС

Один из методов решения данной задачи — *метод динамического программирования*.

Будем последовательно заполнять два массива M и P, где $M[i]$ — минимальная стоимость проезда с нулевой по i-ю остановку, а $P[i]$ — номер последней станции в оптимальном пути. Тогда:

$$M[1] = \text{окр}(s[1]), \quad P[1] = 0;$$
$$M[2] = \min(\text{окр}(s[1] + s[2]),\ M[1] + \text{окр}(s[2]));$$
$$\ldots$$
$$M[n] = \min(\text{окр}(\sum_{i=1}^{n} s[i]),\ \text{M}[1] + \text{окр}(\sum_{i=2}^{n} s[i]),\ \ldots,$$
$$M[n-1] + \text{окр}(s[n])),$$

где $\text{окр}(x)$ — это число x, округленное по правилу, указанному в условии задачи.

Последовательно вычисляя значения массивов M и P, можно найти ответ на вопрос задачи.

Трудоемкость предложенного метода — $O(n^2)$. Однако существует *жадный* линейный (трудоемкости $O(n)$) алгоритм, который решает поставленную задачу. От каждой станции, на которой покупается очередной билет, вычисляется первая станция, стоимость проезда до которой уменьшилась после округления. До нее и покупается очередной билет. Улучшить полученный таким образом результат невозможно. Доказательство правильности такого жадного алгоритма предоставляется читателям.

11. База данных

Первая часть программы состоит в чтении, анализе данных и их записи в таблицу или другую структуру данных.

Затем мы начинаем анализировать фильтр. В данной задаче можно обойтись без использования известных методов лексического анализа, а просто аккуратно разобрать выражение на части и перебрать различные варианты условий отбора. Можно при анализе составных условий использовать вторую таблицу, в которую заносить данные из первой таблицы (или только их номера), удовлетворяющие условию части фильтра. Это позволит разбирать условия по одному. Можно оптимизировать решение и другими способами, но при столь малом объеме базы данных и столь простом фильтре это, наверное, излишне.

Олимпиады 1997/98 учебного года

Городская олимпиада

Теоретический тур

1. Восстановление таблицы

Заметим, что сумма элементов новой таблицы совпадает с суммой элементов исходной. Действительно, каждый элемент таблицы A при вычислении элементов новой таблицы участвует в качестве слагаемого для каждого элемента той же строки и каждого элемента того же столбца, т. е. всего $n + m - 2$ раза, каждый раз с коэффициентом $\dfrac{1}{n + m - 2}$.

Обозначим сумму элементов исходной матрицы как Σ_A, а сумму элементов преобразованной как Σ_B. Тогда:

$$\Sigma_B = \sum_{i=1}^{m}\sum_{j=1}^{n} b(i, j) =$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\frac{1}{m+n-2}\cdot\left(\sum_{r=1}^{n} a(i, r) + \sum_{i=1}^{m} a(l, j) - 2\cdot a(i, j)\right)\right) =$$

$$= \frac{1}{m+n-2}\cdot\left(\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{r=1}^{n} a(i, k) + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{m} a(l, j) - 2\cdot\Sigma_A\right) =$$

$$= \frac{1}{m+n-2}\cdot\left(n\cdot\Sigma_A + m\cdot\Sigma_A - 2\cdot\Sigma_A\right) = \Sigma_A.$$

Выразим сумму элементов k-го столбца таблицы A через элементы таблицы B и Σ_A:

$$\Sigma_A = \sum_{i=1}^{m} a(i, k) + \Sigma_{ост};$$

$$\sum_{i=1}^{m} b(i, k) = \frac{m-1}{m+n-2}\cdot\sum_{i=1}^{m} a(i, k) + \frac{1}{m+n-2}\cdot\Sigma_{ост} =$$

$$= \frac{m-2}{m+n-2}\cdot\sum_{i=1}^{m} a(i, k) + \frac{1}{m+n-2}\cdot\Sigma_A.$$

Отсюда:

$$\sum_{i=1}^{m} a(i, k) = \frac{m+n-2}{m-2}\cdot\sum_{i=1}^{m} b(k, i) + \frac{1}{m-2}\cdot\Sigma_A.$$

Аналогично выразим сумму элементов k-й строки таблицы A через элементы таблицы B и Σ_A:

$$\sum_{i=1}^{m} a(k, i) = \frac{m+n-2}{n-2}\cdot\sum_{i=1}^{m} b(i, k) - \frac{1}{m-2}\cdot\Sigma_A.$$

Зная сумму элементов k-й строки и l-го столбца таблицы A, а также сумму всех элементов k-й строки и l-го столбца, кроме элемента $a(k,l)$, $((m + n - 2) \cdot b(k, l))$, легко найти сам элемент $a(k,l)$.

Таким образом, вычислив суммы элементов всех строк и столбцов матрицы A и зная элементы матрицы B, можно вычислить элементы исходной матрицы A, используя полученные формулы. Последние выглядят довольно сложными, но их смысл достаточно прост. Главное — не пугайтесь!

2. Программа

Представим k в виде $k = n^2 - m^2 = (n - m) \cdot (n + m)$. Из этого равенства следует, что $n - m$ и $n + m$ должны быть делителями числа k. Тогда, найдя очередную пару p и q, для которых $k = p \cdot q$, можно найти m и n по формулам: $m = (p + q)/2$, $n = (q - p)/2$.

3. Роботы-парашютисты

Решение основано на следующей идее. Роботы должны идти в одном направлении, так как программы у них одинаковые. Но один из них может найти следы другого, а другой — нет. Поэтому после нахождения таких следов один из роботов должен ускорить движение, чтобы догнать второго. Эту идею реализует следующая программа:

```
Алг Поиск
 ПОЛОЖИТЬ ПАРАШЮТ
 ШАГ ВЛЕВО
 пока (не НАЙДЕН ПАРАШЮТ) и (не НАЙДЕН РОБОТ)
       ШАГ ВЛЕВО
 кц
 пока не НАЙДЕН РОБОТ
      ШАГ ВЛЕВО
 кц
кон
```

4. Удаление элементов массива

Для решения задачи заведем вспомогательный массив L, в котором $L[k]$ будет равно минимально возможному значению последнего элемента в максимальной возрастающей последовательности длины k (при этом мы просматриваем исходный массив A слева направо и пересчитываем массив L).

Заметим, что массив L является возрастающим. В начале алгоритма массив L пуст. При обработке i-го элемента исходного массива мы ищем место в массиве L, куда можно поставить данный элемент, — это будет минимальный из индексов k, такой что $A[i] \leqslant L[k]$. Если же таких индексов k нет (т. е. $A[i]$ больше всех элементов массива L), то дописываем $A[i]$ в конец массива L.

При выполнении указанной выше операции массив L остается возрастающим, но мы оставляем больше возможностей для построения возрастающей последовательности, так как мы уменьшаем (или по крайней мере не увеличиваем) последний элемент в последовательности длины k.

Длиной максимальной возрастающей последовательности будет количество элементов в массиве L.

Заметим, что поиск индекса k на каждом шаге можно осуществить за $O(\log(N))$ операций с помощью бинарного поиска (так как массив L упорядочен). Таким образом, трудоемкость всего алгоритма будет составлять $O(N \cdot \log(N))$.

5. Шифровка

Нетрудно заметить, что первые буквы сообщения на каждом уровне сложности шифрования получаются из букв сообщения предыдущего уровня сложности с помощью замены на следующую букву алфавита. Дальнейший анализ показывает, что шифровка на каждом уровне сложности производится с помощью циклического сдвига по алфавиту букв шифровки предыдущего уровня, причем сообщение, закодированное на втором уровне сложности, получается из сообщения, закодированного на первом уровне, следующим образом: первые две буквы сдвигаются на одну букву алфавита; третья и четвертая — на две; пятая и шестая — снова на одну и т. д. Сообщение, закодированное на третьем уровне сложности, полу-

чается из сообщения, закодированного на втором уровне, похожим образом: первые три буквы сдвигаются по алфавиту на одну; следующие три буквы — на две и т. д.

Таким образом, логично предположить, что сообщение, зашифрованное на первом уровне сложности, получено из исходной фразы путем сдвига букв, стоящих на нечетных местах, на одну букву алфавита, а букв, стоящих на четных местах, — на две буквы алфавита. Аналогично, четвертый уровень сложности предполагает сдвиг первых четырех букв сообщения на одну букву алфавита, следующих четырех — на две и т. д.

Расшифрованная фраза выглядит так: ПО-МОЕМУ, ЭТО НЕ ТАКАЯ УЖ ПРОСТАЯ ЗАДАЧА, а фраза, зашифрованная на четвертом уровне сложности, — УУ-СХМУЩ, ДЦФ УМ ЧЖРИД ЩЛ ХХХШЪ ДД МЖ ЙЗЭЗ.

6. Алгоритм

Данный алгоритм вычисляет сумму элементов таблицы с i-го по j-й.

Практический тур

7. «Приятное» число

Сначала рассмотрим вопрос: как из n-значного «приятного» числа получить $(n+1)$-значное «приятное» число? Найдем остаток от деления данного числа на $n+1$, умножим полученный остаток на 10 и посмотрим, какую цифру необходимо приписать к полученному, чтобы оно делилось на $n+1$. Дописав к данному числу найденную цифру (возможно, таких чисел может быть несколько), мы и получим $(n+1)$-значное «приятное» число.

Например, 123 654 — шестизначное «приятное» число. Из него можно получить одно «семизначное» приятное число следующим образом: 123 654 при делении на 7 дает остаток 6; $6 \times 10 = 60$; ближайшее делящееся на 7 число — 63; следовательно, для делимости на 7 надо дописать 3; значит, 1 236 543 делится на 7, т. е. является семизначным «приятным» числом.

Окончательный алгоритм, строящий все большие «приятные» числа, выглядит так:

1. Устанавливаем $n = 1$. Тогда 1, 2, ..., 9 — однозначные «приятные» числа.
2. Строим $(n+1)$-значные «приятные» числа для всех n-значных «приятных» чисел, если такие есть. Если их нет, то завершаем работу.
3. Увеличиваем n на 1 и повторяем процесс.

Наибольшее найденное число является ответом к задаче.

8. Множества

Для решения задачи необходимо умение анализировать логические выражения, рисовать на экране круги и окрашивать области. Остальное в особых пояснениях не нуждается.

Областная олимпиада

Практический тур

9. Прогрессия из простых чисел

Заметим, что разность арифметической прогрессии q должна делиться на 2. В противном случае одно из чисел p и $p + q$ делится на 2; следовательно, длина прогрессии не может быть больше двух, а единственным вариантом прогрессии из двух чисел является прогрессия, начинающаяся с двойки. Аналогично, если разность арифметической прогрессии не делится на 3, то одно из чисел p, $p+q$, $p + 2 \cdot q$ делится на 3; следовательно, длина прогрессии не может быть больше трех. Таким образом, если мы хотим получить арифметическую прогрессию, состоящую из простых чисел, длина которой больше 10, то разность этой прогрессии должна делиться на числа 2, 3, 5, 7 и 11. Так как в условии задачи сказано, что простые числа не должны превосходить 32 768, то разность наиболее длинной прогрессии не превосходит $2310 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11$. В этой прогрессии должно быть не больше 13 членов. (Если взять прогрессию, состоящую хотя бы из 14 членов, то ее разность должна делиться на 30 030, тогда последнее число прогрессии будет не меньше 420 420, что превосходит указанную границу.)

Исходя из указанных соображений, алгоритм решения задачи выглядит следующим образом: надо найти все простые числа, меньшие 32 768, и организовать двойной перебор полученных чисел по первым числам прогрессии и различным ее знаменателям, удовлетворяющим вышеизложенным замечаниям.

10. Пробирки

Построим граф, изображающий возможные состояния набора из трех пробирок и возможные переходы между ними. Вершины этого графа будут соответствовать всем возможным различным распределениям кваса в пробирках. При этом каждое распределение задается парой неотрицательных целых чисел (x_1, x_2), где x_i — количество кваса в пробирке i; заметим, что $x_3 = 100 - (x_1 + x_2)$. Две вершины мы соединяем ребром, если соответствующие им состояния могут быть достигнуты за одно переливание. Тогда задача сводится к нахождению кратчайшего пути в данном графе, для чего можно использовать *волновой алгоритм*.

Олимпиады 1998/99 учебного года

Городская олимпиада

Теоретический тур

1. Доска

Произведение всех чисел, записанных на доске, при выполнении указанного в задаче алгоритма не меняется, т. е. является *инвариантом*. Обозначим его как P. Из условия задачи следует, что $P = -1$, а это значит, что в результате выполнения алгоритма на доске останется одно число, которое будет равно -1.

2. Алгоритм

Данная программа осуществляет перевод числа из десятичной системы счисления в двоичную.

3. Матричный порядок

Для начала заметим, что верны следующие утверждения:

- если два элемента таблицы принадлежали одной строке (столбцу), то после любой перестановки строк и столбцов таблицы они также будут принадлежать одной строке (столбцу);
- в последней матрице первая строка и первый столбец будут упорядочены по убыванию.

Исходя из этих соображений, алгоритм, позволяющий определить элемент, стоящий на последнем месте последней строки в последней таблице, будет выглядеть так.

1. Найдем наибольший элемент таблицы и обозначим его как M. Поскольку все элементы таблицы различны, он — единственный.
2. Найдем в строке, содержащей M, наименьший элемент и обозначим его как K. Тогда столбец, содержащий K, будет последним в последней матрице.
3. Найдем в столбце, содержащем M, наименьший элемент и обозначим его как N. Тогда строка, содержащая N, будет последней в последней матрице.
4. В исходной таблице общий элемент строки, содержащей N, и столбца, содержащего K, и является искомым.

4. Зеркало

Задачу можно переформулировать следующим образом: в выпуклый многоугольник необходимо вписать круг наибольшего радиуса.

Заметим, что искомый круг должен касаться либо двух параллельных сторон многоугольника, либо трех его сторон: только в этих случаях его радиус нельзя увеличить (в первом случае круг можно сдвинуть между параллельными сторонами в крайнее положение, и он коснется третьей стороны). Таким образом, искать требуемый круг можно только среди таких кругов, которые *опираются* на три стороны. Отсюда название метода, широко применяемого в аналогичных задачах, — *метод опорных точек*.

В данной задаче метод опорных точек дает следующий алгоритм.

1. Перебираем все тройки сторон и для каждой строим окружность, вписанную в треугольник, образованной прямыми, содержащими эти стороны. В случае параллельности двух прямых можно считать треугольник вырожденным, а окружность строить обычным способом (ее центр лежит в точке пересечения биссектрис внутренних углов).
2. Для каждой построенной окружности проверяем, лежит ли она внутри многоугольника. Для этого надо убедиться, что внутрь не попала ни одна из вершин многоугольника и что расстояние от центра до каждой стороны не меньше радиуса.
3. Окружность наибольшего радиуса среди удовлетворяющих условиям п. 2 является искомой.

Примечание: можно рассматривать только такие тройки сторон, которые определяют остроугольные или прямоугольные треугольники — это даст некоторое ускорение перебора.

5. Теннисный турнир

Заметим, что целая часть от деления m на 3 — это и есть количество выигранных партий. Определим, какие ограничения необходимо наложить на k. Рассмотрим два случая.

Первый случай: m не делится на 3. Тогда одна партия была проиграна, значит $k \geqslant 3$ и $k \leqslant \mathtt{int}(m/3) \cdot 2 + 3$. Отсюда $3 \leqslant k \leqslant \mathtt{int}(m/3) \cdot 2 + 3$.

Второй случай: m делится на 3. Тогда проигранных партий могло не быть совсем. Отсюда $0 \leqslant k \leqslant \mathtt{int}(m/3) \cdot 2 + 3$.

Используя указанные замечания, решить задачу нетрудно.

6. Максимальное произведение

Рассмотрим разбиение числа N на натуральные слагаемые, произведение которых максимально. Пусть минимальное из этих слагаемых равно A, а максимальное — B.

Заметим, что:

1) $A > 1$, иначе можно было бы на 1 увеличить любое другое слагаемое, увеличив произведение;
2) $A < 5$, иначе A можно было бы заменить на сумму $(A - 2)$ и 2, произведение которых больше A;
3) в разложении должны присутствовать все натуральные числа между A и B, кроме, может быть, одного.

Докажем это утверждение. Пусть в разложении отсутствуют числа p и q $(p < q)$, а $p - 1$ и $q + 1$ присутствуют. Обозначим произведение всех чисел, кроме $p - 1$ и $q + 1$, через Π, а сумму — через S. Тогда произведение всех слагаемых равно $\Pi(p - 1)(q + 1) = \Pi(pq + p - q - 1) < \Pi \cdot p \cdot q$ (так как $p < q$), а сумма равна $S + (p - 1) + (q + 1) = S + p + q$. Следовательно, наше разложение не является оптимальным, так как числа $(p - 1)$ и $(q + 1)$ можно заменить на числа p и q, — при этом сумма не изменится, а произведение увеличится.

С учетом всех этих замечаний решение ищется так.

Будем строить сумму из подряд идущих натуральных чисел начиная с двух, с трех и с четырех до тех пор, пока эта сумма не станет больше заданного числа N. Затем найдем разность q получившейся суммы и заданного числа N. Если $q \geqslant A$, то q из разложения исключим, иначе исключаем из рассмотрения само получившееся разложение. Далее найдем произведения, соответствующие найденным разложениям. Наибольшее из них и определяет искомое разбиение.

7. Максимальная сумма

Введем обозначения: $S(m,k) = a_m + a_{m+1} + \ldots + a_{k-1} + a_k$, где $1 \leqslant m \leqslant k \leqslant n$; $P(i)$ — наибольшее возможное значение $S(m,k)$ при $1 \leqslant m \leqslant k \leqslant i$; $Q(i)$ — наибольшее возможное значение $S(l,i)$ при $1 \leqslant l \leqslant i$. Тогда первоначальная задача сводится к вычислению $P(n)$.

Найдем рекуррентные формулы для вычисления $P(i)$ и $Q(i)$. Очевидно, что $P(1) = Q(1) = A(1)$. Пусть известны значения $P(i)$ и $Q(i)$ для некоторого i, которые достигаются при $l = L(i)$, $m = M(i)$, $k = K(i)$. Выведем формулы для $P(i + 1)$, $Q(i + 1)$, $L(i + 1)$, $M(i + 1)$, $K(i + 1)$.

При нахождении $Q(i + 1)$ возможны два случая. В первом из них $L(i + 1) = i + 1$, тогда $Q(i + 1) = A(i + 1)$. Во втором случае $L(i + 1) \leqslant i$, тогда $L(i + 1) = L(i)$, $Q(i + 1) = Q(i) + A(i + 1)$.

Общая формула для $Q(i + 1)$ будет следующей:

$$Q(i + 1) = \mathtt{max}(A(i + 1),\ Q(i) + A(i + 1)).$$

Аналогично, два случая возможны и при нахождении $P(i+1)$. В первом из них $K(i + 1) = i + 1$, тогда $P(i + 1) = Q(i + 1)$. Во втором случае $K(i + 1) \leqslant i$, тогда $P(i + 1) = P(i)$.

Общая формула для $P(i + 1)$ имеет вид:

$$P(i + 1) = \mathtt{max}(P(i),\ Q(i + 1)).$$

Вычисляя последовательно значения $P(i)$, мы найдем значение $P(n)$, что и позволит определить искомую подпоследовательность.

Практический тур

8. «Пестрое» число

Эта задача имеет несколько решений.

Первое решение основывается на построении чисел с различными цифрами в убывающем порядке начиная с числа 9 876 543 210 и на проверке каждого числа на делимость. При этом алгоритм перечисления «пестрых» чисел аналогичен описанному в задаче 91/92(11).

Другой алгоритм. Самое большое «пестрое» число — это 9 876 543 210. Вычитая из него все числа от 1 до n, мы найдем то, которое делится на n. Из полученного числа вычитаем n, $2n$, $3n$ и т. д. пока не получим «пестрое» число. Для больших n данный алгоритм работает заметно быстрее.

9. Электронное письмо

Считав данные, мы заполняем двумерный массив кодов, одномерный массив символов и одномерный массив кодов сообщения (для первой части задачи) или одномерный массив символов (для второй ее части). Далее по очереди для каждой из кодировок мы проверяем, все ли коды сообщения в ней можно перевести в символы, и если это возможно, то

декодируем и выводим сообщение. Если же ни одна из кодировок не содержит все коды сообщения, то выводим фразу, что письмо невозможно раскодировать.

Во второй части задачи мы пробуем перевести сообщение из каждой кодировки в каждую из других. Для этого, по очереди перебирая все кодировки, мы переводим исходное сообщение в последовательность кодов и применяем программу из первой части задачи.

Областная олимпиада

Теоретический тур

10. Слово

Заметим, что предложенный процесс обратим: зная какое-либо состояние системы, можно просчитать все предыдущие ходы. Очевидно, что начальное положение указателя входить в цикл не может. Таким образом, даже если для данного слова и имеется последовательность, образующая цикл, в нее невозможно попасть, а это значит, что алгоритм конечен.

11. Алгоритм

А) Данный алгоритм вычисляет C_n^m — количество сочетаний из n по m.

Б) Докажем, что количество рекурсивных вызовов при этом равно $2C_n^m - 2$.

Для $m = n$ и $m = 0$ формула справедлива. Обозначим количество рекурсивных вызовов при вычислении C_n^m через $f(C_n^m)$. Докажем, что $f(C_n^m) = 2C_n^m - 2$, методом математической индукции по n.

Для $n = 2$ значение f рассчитать достаточно легко при всех m.

Пусть данная формула выполняется для n при всех m. Докажем, что она выполняется при всех m и для $n + 1$:

$$f(C_{n+1}^m) = 2 + f(C_n^{m-1}) + f(C_n^m) =$$
$$= 2 + (2C_n^{m-1} - 2) + (2C_n^m - 2) =$$
$$= 2 + 2(C_n^m + C_n^m) - 4 = 2C_{n+1}^m - 2.$$

Таким образом, значение функции и количество рекурсивных вызовов при ее вычислении связано следующим соотношением: $f(C_n^m) = 2C_n^m - 2$.

12. Прогрессия

Структура данных. Заведем битовый массив, который мы будем расширять по мере обработки чисел из файла. Кроме того, заведем три отдельные переменные — S (первый член последовательности), F (последний член последовательности) и D (разность последовательности). Таким образом, i-й бит массива будет соответствовать числу $S + iD$.

Начальная инициализация. Прочитав первые два члена, установим S как минимальный из них, F — как максимальный, D — как разность. Битовый массив на первом этапе будет состоять из двух элементов, соответствующих S и F (кроме случая $D = 0$, когда мы просто проверяем, что все числа одинаковы).

Основной ход алгоритма. Обрабатываем оставшиеся числа по очереди. При обработке очередного числа возможны следующие варианты:

1) число попадает в массив, и в нем уже установлен соответствующий бит; при этом последовательность составить нельзя, так как D должно быть равно 0;
2) число попадает в массив, и соответствующий бит в нем не установлен, — тогда просто устанавливаем его;
3) число не попадает в массив — тогда меняем S, F и D так, чтобы оно попадало в массив, т. е. «раздвигаем» и «смещаем» массив и после этого устанавливаем соответствующий бит.

В последнем случае может возникнуть ситуация, когда количество неустановленных битов заведомо меньше оставшихся в файле чисел. Тогда надо сразу давать ответ: «Прогрессию составить невозможно».

13. Точки в круге

Заметим, что окружность, ограничивающая искомый круг, содержит либо одну данную точку, либо по крайней мере две, причем в первом случае эта точка — самая удаленная от оси X.

Это замечание позволяет применить для решения задачи *метод опорных точек* (сравните решение с задачей 98/99(4)).

Согласно этому методу, решение надо искать среди окружностей, которые проходят либо через точку с наибольшей по модулю ординатой, либо через пару заданных точек. Центр таких окружностей в первом случае лежит на оси X и имеет ту же абсциссу, что и удаленная точка, а во втором — лежит на пересечении серединного перпендикуляра к отрезку, соединяющему пару точек, с осью X. Все такие окружности легко строятся. При этом проверку, что все остальные точки попадают внутрь кругов, можно не делать. Достаточно заметить, что в наибольшую такую окружность обязаны попасть все точки, так как точки, определяющие эту окружность, не смогут поместиться ни в какую другую.

14. Невозможная сумма

Заметим, что число P не принадлежит множеству M.

Докажем следующее утверждение: если из первых k чисел (N_1, N_2, ..., N_k) можно получить все суммы подряд до s включительно и следующее число N_{k+1} меньше или равно $s + 1$, то из первых $k + 1$ чисел можно получить все суммы подряд до ($s + N_k + 1$) включительно.

Доказательство. Пусть из чисел N_1, N_2, N_3, ..., N_k можно получить суммы от 1, 2, 3, ..., s. Тогда, если $N_{k+1} \leqslant (s+1)$, то прибавляя к каждой из предыдущих сумм N_{k+1}, мы получим суммы $1 + N_{k+1}$, $2 + N_{k+1}$, $3 + N_{k+1}$, ... , $s + N_{k+1}$. Принимая во внимание уже имеющиеся суммы и вновь полученные, видно, что из первых $k + 1$ чисел можно получить все суммы подряд до ($s + N_{k+1}$) включительно.

Алгоритм нахождения наименьшего натурального числа P, которое не представимо в виде суммы элементов никакого подмножества M, приведен ниже.

1. Упорядочим данное множество.
2. Если первое число множества больше 1, то $P = 1$, иначе $k = 1$, $s = 1$ и переходим к п. 3.
3. Если $N_{k+1} \leqslant s + 1$, то $s = s + N_{k+1}$, $k = k + 1$ и переходим к п. 3, иначе переходим к п. 4.
4. $P = s + 1$.

15. Пути коня

Наиболее эффективный алгоритм использует *метод динамического программирования*.

Будем заполнять k таблиц размером 8×8. В первой таблице будем отмечать количество различных путей с поля [1,1] на все остальные поля, содержащих 1 ход. Во второй таблице будем отмечать количество различных путей с поля [1,1] на все остальные поля, содержащих 2 хода, и т. д. В k-й таблице, соответственно, будет содержаться информация о количестве различных путей с поля [1,1] на все остальные поля, содержащих k ходов.

Сначала обнулим исходные таблицы. Затем в первой таблице клеткам [2, 3] и [3, 2] присвоим значение 1, а всем остальным присвоим значение 0.

Далее будем заполнять каждую следующую таблицу, используя предыдущую. Клетке с координатами $[i, j]$ следующей таблицы присвоим значение суммы чисел в клетках с координатами $[i - 1, j - 2]$, $[i + 1, j - 2]$, $[i - 2, j - 1]$, $[i + 2, j - 1]$, $[i - 2, j + 1]$, $[i + 2, j + 1]$, $[i + 1, j + 2]$, $[i - 1, j + 2]$ предыдущей таблицы (причем необходимо также проверять существование этих клеток).

В итоге в k-й таблице в клетке с координатами [8, 8] будет записано количество различных путей с поля [1,1] на поле [8, 8], содержащих k ходов.

Заметим, что при заполнении очередной таблицы мы используем значения ячеек только предыдущей таблицы; следовательно, для реализации предложенного алгоритма достаточно двух таблиц. Более того, так как конь всегда ходит на поле противоположного цвета, то для реализации алгоритма достаточно и одной таблицы! Нечетные ходы тогда будут учитываться на «белых» клетках, а четные — на «черных».

16. Перевозки

Упорядочим все рейсы по дате и времени вылета и по дате и времени прилета. Заметим, что каждый рейс в эту упорядоченную последовательность будет входить дважды.

Далее, проходя по полученным массивам, надо прибавлять и вычитать количество пассажиров, запоминать максимальное их количество и время, ему соответствующее.

Практический тур

17. Ноты

Решение данной задачи предполагает аккуратную реализацию алгоритма, описанного в условии.

18. Колье

Пусть исходные данные считаны в массив P. Будем обходить этот массив в обратном порядке; при этом для каждого кольца i будем хранить информацию о двух цепочках: о самой длинной ($L[i]$) и о следующей по длине ($R[i]$), образованных уже просмотренными кольцами.

Когда мы обрабатываем кольцо i, оно вместе со своей самой длинной цепочкой образует цепочку длины $L[i] + 1$. Значит, у «родительского» кольца $P[i]$ (за которое зацеплялось кольцо i) мы можем обновить информацию о самых длинных цепочках (т. е. мы выбираем две максимальные цепочки из $L[P[i]]$, $R[P[i]]$ и $L[i] + 1$).

Ответом на задачу будет максимум из чисел $L[i] + 1 + R[i]$ (две максимальные подцепочки плюс само кольцо).

Олимпиады 1999/00 учебного года

Городская олимпиада

Теоретический тур

1. Алгоритм

В результате выполнения данного алгоритма на экран будут выведены суммы четных и нечетных элементов таблицы A.

Рекурсивные обращения будут выглядеть следующим образом:

$X(20, A, 0, 0)$;
$X(19, A, 0, A[20])$;
$X(18, A, A[20], A[19])$;
$X(17, A, A[19], A[20] + A[18])$;
$X(16, A, A[20] + A[18], A[19] + A[17])$;

и т. д.

Таким образом, после окончания работы алгоритма переменная B будет хранить сумму четных элементов массива, а C — сумму нечетных элементов массива.

2. Шестеренки

Заметим, что из двух вращающихся сцепленных шестеренок одна обязательно вращается по часовой стрелке, а другая — против. Поэтому всю систему шестеренок можно привести в движение, если выполняются два условия:

1) все шестеренки связаны между собой;
2) в системе нет конфликтов, т. е. ситуаций, когда две спленные шестеренки должны крутиться в одну сторону. Это условие эквивалентно отсутствию в системе шестеренок циклов нечетной длины.

Оба эти условия можно проверить с помощью *волнового алгоритма*. Припишем каждой шестеренке пометку, сначала равную 0. Изменим у первой шестеренки пометку на 1. Такая пометка означает, что шестеренка начала вращаться (пусть — по часовой стрелке). Тогда все шестеренки, связанные с первой, пометим числом 2, означающим вращение против часовой стрелки. Аналогично продолжаем помечать другие шестеренки: сцепленные с 1 помечаем числом 2 и наоборот. Если в какой-то момент возникнет необходимость пометить шестеренку, уже имеющую пометку, и эта пометка — не та, что нужна, то значит, к этой шестеренке мы подобрались от первой двумя путями, имеющими длину разной четности, что равносильно наличию цикла нечетной длины.

В конце алгоритма надо также проверить, все ли шестеренки оказались помечены.

3. Делимость

Любое число S в системе счисления с основанием K можно представить в виде:

$$a_m \cdot K^m + a_{m-1} \cdot K^{m-1} + \ldots + a_1 \cdot K + a_0,$$

где a_m, a_{m-1}, ..., a_1, a_0 — цифры данного числа. Тогда число S делится на N, если $\sum_{i=0}^{m}(a_i \cdot K^i) \bmod N$ делится на N. Исходя

из этих соображений, требуемый алгоритм будет выглядеть следующим образом:

```
procedure delimost1
begin
 L:=1; R:=0;
 while есть цифры do
    begin
       берем последнюю цифру;
       R:=(R+цифра*L) mod N;
       L:=(L*K) mod N
    end;
 if R=0 then число S делится на N
        else S не делится на N
end;
```

Можно также реализовать описанный выше алгоритм, используя схему Горнера:

```
procedure delimost2
begin
 R:=0;
 while есть цифры do
    begin
      R:=(R*K+цифра) mod N;
    end;
if R=0 then число S делится на N
       else S не делится на N;
end;
```

4. Система тестов

Искомая система тестов должна включать в себя тесты для каждого из следующих пяти случаев:

- треугольник правильный;
- треугольник равнобедренный;
- треугольник неравнобедренный;
- треугольника не существует;
- треугольник вырожденный (иногда этот случай включают в предыдущий, но лучше проверить его отдельно, как и любой другой крайний случай).

Также эта система должна включать в себя тесты на каждый из возможных пограничных случаев. Кроме того, программу необходимо проверить на равноправность сторон треугольника. Например, тест со сторонами 2, 2, 3 желательно повторить: 2, 3, 2 и 3, 2, 2.

Отметим также, что все тесты желательно варьировать на числах из всей области их допустимых значений.

5. Фляга

Переформулируем условие задачи: пусть каждый велосипедист имеет по фляге, и при обгоне велосипедисты обязательно меняются флягами. Отсюда следует, что фляги друг друга не обгоняют, — значит, расстояние, пройденное любой флягой, отличается от расстояния, пройденного любой другой флягой не более чем на один круг. Заметим также, что в сумме все фляги за один час проедут то же расстояние, что и велосипедисты. Обозначим его через S, тогда $S = \sum_{i=1}^{n} v_i$, где v_i — скорость i-го велосипедиста. Из условия задачи следует, что все фляги проедут целое количество кругов, причем какая-то часть фляг проедет на один круг больше, чем другие. Тогда каждая фляга проедет либо (S div n), либо (S div n) + 1 кругов.

Чтобы определить, сколько кругов проедет данная фляга, вычислим значение $p = S \bmod n$, указывающее, сколько фляг проедут на один круг больше среднего. Найдем p велосипедистов, скорости которых наибольшие. Если k-й велосипедист попадает в их число, то исходная фляга проедет $\frac{S}{n} + 1$; иначе $\frac{S}{n}$.

6. Перестановки

Для обращения части массива может быть использована следующая процедура:

```
procedure obr(var a:mass; k,l:integer;);
   var i:integer;
begin
   for i:=0 to (l-k)div2 do
       swap(a[k+i],a[l-i])
end;
```

Циклический сдвиг элементов массива на k элементов можно осуществить, используя процедуру ***obr*** следующим образом:

```
obr(a,1,k);
obr(a,k+1,n);
obr(a,1,n)
```

7. Стадион

Ответ можно получить за один проход по кругу вдоль нашего ряда, начиная с произвольно выбранного места. При этом обходе надо следить за несколькими счетчиками и изменять их значения:

- M — количество просмотренных мест;
- K — количество сидящих подряд болельщиков в красных шарфах;
- S — количество сидящих подряд болельщиков в синих шарфах.

Правила изменения этих счетчиков просты, и мы их оставляем читателям для самостоятельного рассмотрения. Максимальное текущее значение счетчиков K и S — это и есть искомый результат.

Отметим, что при этом остались нерассмотренными два случая:

1) если начало обхода попало внутрь группы болельщиков одной команды;
2) когда в ряду нет ни одного зрителя.

Эти случаи читателям тоже предлагается исследовать самостоятельно.

Практический тур

8. Частотный словарь

В данной задаче при обработке текста после прочтения очередного слова требуется выполнять следующие функции:

1) находить нужное слово, чтобы определить, сколько раз оно встретилось ранее;
2) включать новое слово в рассматриваемую структуру данных.

Для указанных целей лучше всего использовать *бинарное дерево*. В каждом узле такого дерева хранится слово и коли-

чество его вхождений в текст. Слова же записываются в дерево так, что все слова, предшествующие данному слову по алфавиту, находятся в его левом поддереве, а все последующие — в правом. Указанный порядок позволяет быстро находить слово, двигаясь по дереву от корня, а дойдя до листа и не найдя нужного слова, записать в дерево новый элемент.

9. Электронная таблица

Можно разбить решение этой задачи на два этапа.

На первом этапе мы проверяем наличие циклов, используя для этого *волновой алгоритм*. Выбираем произвольную ячейку и отмечаем ее единицей во вспомогательной таблице (тех же размеров, что и исходная). Далее отмечаем те ячейки, на которые она ссылается, двойкой. Затем те из еще не помеченных ячеек, на которые ссылаются формулы из ячеек, помеченных двойкой, помечаем тройкой и т.д. Если на каком-то шаге мы получим ссылку на уже помеченную в данной серии ячейку, то это означает наличие цикла, и программа заканчивает работу. Второй вариант окончания серии («распространения волны») — когда на каком-то шаге не появилось ячеек, помеченных очередным числом. В этом случае мы ищем еще не помеченную ячейку (а если ее нет, то и циклов нет!) и начинаем процесс с нее заново с увеличенного номера. Ячейки, помеченные в предыдущих сериях (те, у которых номер меньше начального номера данной серии), мы уже не помечаем, так как они не могут входить в новый цикл.

На втором этапе (при отсутствии циклов) мы реализуем вычисление значений ячеек. При указанных небольших размерах таблицы это можно сделать с помощью короткого (по записи программы) рекурсивного алгоритма.

Областная олимпиада

Практический тур (день первый)

10. Упорядочивание файла

Заведем целочисленный массив с индексами от 1 до 65 535 и будем подсчитывать, сколько раз встретилось каждое из чисел при чтении исходного файла: встречая некоторое число, мы увеличиваем ячейку массива с индексом, равным этому числу, на 1. По окончании считывания исходных дан-

ных мы выводим каждое из чисел от 1 до 65 535 столько раз, сколько это число входит в исходный массив (это число лежит в соответствующей ячейке массива). Описанный алгоритм часто называют *математической сортировкой*.

11. Архипелаг в океане

Очевидно, что мост следует строить либо между вершинами островов, либо между вершиной и стороной. Тогда расстояние между двумя островами можно найти, найдя все расстояния между вершинами одного и вершинами другого острова, а затем — между вершинами каждого острова и сторонами другого острова. Если оказывается, что отрезок, соединяющий ближайшие точки островов, проходит через границу третьего острова, то это означает, что между первыми двумя островами мост заведомо не нужен, тогда расстояние между ними можно обозначить как бесконечное (любым числом, заведомо бóльшим ответа на задачу).

Далее мы можем построить полный граф, в котором вершинам будут поставлены в соответствие острова, а ребрам приписаны расстояния между этими островами. Отметим, что мосты между некоторыми островами построить нельзя (например, если они должны будут пройти через территорию других островов), но на это можно не обращать внимания, так как такой мост никогда не будет включен в искомое множество мостов. Действительно, вместо него в это множество можно включить два других моста — до промежуточного острова и от промежуточного до конечного; общая протяженность мостов при этом уменьшится.

Таким образом, исходная задача сводится к поиску *минимального остовного дерева*, т. е. к выделению некоторых ребер графа так, чтобы сумма весов была минимальна, а выделенные ребра образовывали связный граф на тех же вершинах. Данная задача хорошо известна и решается, например, *алгоритмами Краскала* или *Прима*.

Практический тур (день второй)

12. Атомный реактор

Задача решается методом *динамического программирования*.

Обозначим через $T(n, m, k)$ количество различных стопок, содержащих n пластин, в которых вставлено m датчи-

ков, причем после последнего датчика идут k пластин. Нас интересуют значения $T(n, m, k)$ при $1 \leqslant n \leqslant N$, $0 \leqslant m \leqslant M$, $0 \leqslant k \leqslant K$. Будем находить эти значения, вписывая результат в соответствующую трехмерную таблицу.

Очевидны следующие начальные условия:

- $T(n, 0, k) = 1$ для всех допустимых n и k и $0 \leqslant n \leqslant K$;
- $T(n, 0, k) = 0$ при $n > K$.

Вычисление остальных значений производится по следующим рекуррентным соотношениям:

- $T(n, m, 0) = \sum_{i=1}^{k} T(n, m-1, i)$. Эта ситуация соответствует добавлению датчика в конец стопки пластин;
- $T(n, m, k) = T(n-1, mk-1)$, $k > 0$ — эта ситуация соответствует добавлению пластины.

Окончательный результат вычисляется по формуле: $\sum_{i=1}^{k} T(n, m, i)$.

13. Грязные предвыборные технологии

Указанные в задаче ограничения позволяют хранить все данные в памяти вместе с рабочими массивами. Небольшое количество кандидатов (до 10) позволяет организовать перебор различных вариантов дискредитации (всего $2^9 = 512$); в каждом из них можно подсчитать, сколько избирателей необходимо подкупить для обеспечения победы своего кандидата, а затем выбрать минимальную стоимость проведения предвыборной кампании.

При переборе вариантов дискредитации можно использовать алгоритм перебора всех подмножеств, например, с помощью увеличения на 1 двоичного числа от 000000000 до 111111111 (каждому числу соответствует подмножество).

«Подкуп» избирателей при фиксированном наборе дискредитированных кандидатов можно осуществлять по одному, начиная со сторонников текущего лидера, — это одновременно дает голос нужному кандидату и отнимает голос у лидера. При этом после каждого шага необходимо пересчитывать лидера. Процесс заканчивается, когда единоличным лидером становится нужный кандидат.

Олимпиады 2000/01 учебного года

Городская олимпиада

Теоретический тур

1. Последовательность

Заметим, что описанное в задаче преобразование последовательности напоминает прибавление 1 к двоичному числу. Отличие состоит только в том, что меняются не единицы, а нули и изменение последовательности происходит не справа, а слева. Кроме того, могут изменяться не все нули, а только их часть. Посмотрим, чему соответствует описанная в условии операция при такой интерпретации последовательности. Рассмотрим последовательность как двоичную запись натурального числа, инвертируем ее (заменим нули единицами, а единицы — нулями) и будем рассматривать справа налево. Как мы уже заметили, если все нули, идущие сначала, заменить на единицы, а последующую единицу — на нуль, то это означает изменение всех идущих в конце числа единиц на нули, а последнего нуля — на единицу. Эта операция в точности совпадает с прибавлением единицы к двоичному числу. Если же в операции участвуют не все нули, то соответствующее число получается меньше исходного.

Итак, сначала у нас имеется последовательность, состоящая из единиц. Значит, соответствующее двоичное число равно нулю. В конце процесса мы должны получить последовательность из N нулей, что соответствует двоичной записи числа $2^N - 1$. Так как на каждом шаге мы либо уменьшаем число, либо увеличиваем его на 1, то для достижения цели нам потребуется не менее чем $2^N - 1$ шагов, — что и достигается, если на каждом шаге менять на единицы все идущие с начала последовательности нули.

2. Центральная симметрия

Алгоритм решения данной задачи может быть следующим. Занесем координаты точек в массив X размером $2 \times n$, где первая строка — абсциссы точек, а вторая — их ординаты. Упорядочим в порядке возрастания столбцы полученного массива по элементам первой строки, а если элементы первой строки равны, то упорядочение произведем по элемен-

там второй строки. Затем определим координаты центра симметрии Xs, Ys:

- если количество точек нечетно, то

$$\begin{cases} Xs = X(n \text{ div } 2 + 1,1); \\ Ys = X(n \text{ div } 2 + 1,2); \end{cases}$$

- если количество точек четно, то

$$\begin{cases} Xs = (X(1,1) + X(n,1))/2; \\ Ys = (X(1,2) + X(n,2))/2. \end{cases}$$

Множество точек является центрально-симметричным, если выполняются условия:

$$\begin{cases} (X(k,1) + X(n - k + 1,1))/2 = Xs; \\ (X(k,2) + X(n - k + 1,2))/2 = Ys, \end{cases}$$

где k меняется от 1 до n div 2.

3. Юстас и Алекс

Пусть $S(t)$ — количество различных сообщений, время передачи которых не превышает t секунд. Тогда $S(1) = 2$ (0 и A); $S(2) = 5$ (0, A, AA, $Б$, B); $S(3)=10$ (0, A, AA, AAA, $Б$, B, $AБ$, $БA$, AB, BA), где 0 означает пустое сообщение.

Предполагая, что известны $S(1)$, $S(2)$, ..., $S(n - 1)$, выведем рекуррентное соотношение для нахождения значения $S(n)$ при $n > 3$.

Сообщение может начинаться с сигнала A; таких сообщений с длиной не более n может быть ровно $S(n - 1)$, поскольку далее может следовать любое из сообщений длины не более $n - 1$. Сообщение может начинаться с $Б$ или B; таких сообщений $S(n - 2)$. Сообщение может начинаться с $Г$; таких сообщений $S(n - 3)$. Кроме того, возможно и пустое сообщение. Тогда $S(n) = S(n - 1) + 2 \cdot S(n - 2) + S(n - 3) + 1$.

Вычисляем последовательно $S(t)$:

$S(4)=10+10+2+1=23$;
$S(5)=23+20+5+1=49$;
$S(6)=49+46+10+1=106$;
$S(7)=106+98+23+1=228$;
$S(8)=228+212+49+1=490$;
$S(9)=490+456+106+1=1053$.

Таким образом, ответ в задаче — 9 секунд, так как существует более 1000 различных сообщений, длина которых не превышает 9 секунд, а количество сообщений, длина которых не превышает 8 секунд, — менее 1000.

4. Меткий стрелок

Алгоритм будет оптимальным, если после каждого выстрела площадь щита, где, возможно, находится шарик, будет сокращаться вдвое. Для этого первых два выстрела мы производим в два смежных угла щита. Далее мы проводим серединный перпендикуляр к отрезку, соединяющему точки выстрелов. Очевидно, что шарик находится за той половиной щита, выстрел в которую указан как ближайший; вторую половину щита из рассмотрения исключаем. Каждый последующий выстрел необходимо производить симметрично точке последнего выстрела относительно некоторой прямой, делящей фанерный щит пополам; при этом мы по очереди выбираем горизонтальные и вертикальные прямые, чтобы в итоге получился квадрат со стороной, не превышающей 10 см.

5. Гигантомания

Если обозначить расстояние (в миллиметрах) до земли от каждой лампочки, кроме первой и последней, за неизвестное, то описанное условие даст нам 1998 линейных уравнений с 1998 неизвестными:

$$x_i = \frac{x_{i-1} + x_{i+1}}{2} - 1. \qquad (*)$$

Естественно тогда обозначить заданные расстояния до земли первой и последней лампочек за x_1 и x_{2000}. Решать такие системы стандартными методами в общем виде — задача очень трудоемкая, однако в данном случае матрица системы уравнений оказывается весьма специфической. В каждой строке этой матрицы отличающихся от нуля коэффициентов будет только три, причем расположены они вдоль главной диагонали и рядом с ней. Такие матрицы называются *трехдиагональными*, а для решения соответствующих систем разработано несколько методов. Хорошо известны *методы прогонки* и *стрельбы*. Метод прогонки был

описан при решении задачи 88/89(19). В нашей же задаче не обязательно находить все расстояния, — достаточно лишь ответить на вопрос, являются ли все значения неизвестных положительными.

Для решения этой задачи лучше всего подходит *метод стрельбы*. Опишем его подробнее. Заметим, что если указано значение неизвестного x_2, то из формулы (*) мы находим

$$x_3 = 2x_2 - x_1 + 1. \qquad (**)$$

Аналогично можно найти значения следующих переменных. Однако последнее вычисленное нами значение x_{2000} может оказаться не равным заданному. Это означает, что указанное значение x_2 неверно. Отметим также, что при подстановке в (**) большего значения x_2 мы получим большее значение x_3, а значит и бóльшие значения всех последующих переменных. Следовательно, получив большее, чем заданное, значение x_{2000}, мы сделаем вывод, что x_2 должно быть меньше, и наоборот.

Отсюда мы получаем следующий метод решения задачи.

1. Предполагаем, что x_2 равно какой-то произвольной величине. (О ее выборе скажем чуть позже.)
2. Последовательно находим следующие по порядку неизвестные по формуле (**).
3. Если какое-то вычисленное значение оказывается меньше нуля, то это означает, что либо гирлянда коснется земли, либо x_2 взято меньшим, чем нужно. При этом надо перейти к п. 1, выбрав новое x_2, большее прежнего.
4. Если все значения переменных положительны, но x_{2000} получилось больше заданного, то x_2 следует подкорректировать в меньшую сторону.
5. Если все значения переменных больше нуля, а x_{2000} не больше заданного, то гирлянда земли не касается.

Название «метод стрельбы» описанный алгоритм получил благодаря следующей аналогии. Выбор x_2 похож на прицельный выстрел, после которого вычисленное значение x_{2000} играет роль результата стрельбы. Если произошел «перелет» или «недолет», то прицел корректируется в нужную сторону. Первые два «выстрела» часто производят исключительно для

«пристрелки», — в первом случае заведомо получая «недолет» (например, взяв $x_2 = 0$), а во втором — заведомо получая «перелет» (например, взяв x_2 равным заданному x_{2000}). Тогда последующие «выстрелы» определяются *методом дихотомии*.

6. Алгоритм

Данная программа реализует *алгоритм Евклида* для нахождения наибольшего общего делителя чисел a и b.

7. Буквы

Для решения задачи достаточно подсчитать, сколько раз каждая буква входит в первое и во второе слово. Если все значения счетчиков во втором случае не превосходят соответствующих значений в первом, то ответ — «Да»; иначе — «Нет».

Практический тур

8. Робот

Рассмотрим 5 векторов:

$$A_1(0;1);$$

$$A_2\left(-\sin\frac{2\pi}{5}, \cos\frac{2\pi}{5}\right);$$

$$A_3\left(-\sin\frac{4\pi}{5}, \cos\frac{4\pi}{5}\right);$$

$$A_4\left(-\sin\frac{6\pi}{5}, \cos\frac{6\pi}{5}\right);$$

$$A_5\left(-\sin\frac{8\pi}{5}, \cos\frac{8\pi}{5}\right).$$

Это векторы, по которым может двигаться робот. Исходная задача эквивалентна поиску неотрицательных целых коэффициентов a_i ($i = 1, 2, 3, 4, 5$), таких что расстояние от точки $a_1A_1 + a_2A_2 + a_3A_3 + a_4A_4 + a_5A_5$ до точки X будет минимальным (при $a_1 + a_2 + a_3 + a_4 + a_5 \leqslant K$).

Заметим, что сумма векторов $A_1 + A_2 + A_3 + A_4 + A_5$ равна нулю. Отсюда следует, что хотя бы один из коэффициентов a_i должен быть нулевым (иначе из всех a_i можно вычесть по единице, и сумма не изменится).

Таким образом, алгоритм заключается в следующем. Выбираем индекс коэффициента, который будет равен 0. Перебираем два из оставшихся коэффициентов. В процессе перебора мы попадаем в некоторую точку Y. До точки X нам нужно пройти расстояние $X - Y$. Вектор $X - Y$ можно разложить по двум оставшимся векторам. Пусть $X - Y = aA_i + bA_j$. Если топлива не хватает, то коэффициенты при A_i и A_j должны быть пропорциональны a и b (при этом мы не забываем, что коэффициенты должны быть целыми и положительными). Таким образом, оставшееся количество топлива мы делим между коэффициентами a_i и a_j в пропорции a и b. Так как коэффициенты должны быть целыми, мы должны перебрать несколько вариантов (из-за округления в большую и меньшую стороны).

В итоге из всех точек, полученных в процессе перебора, мы должны выбрать ближайшую к точке X.

9. Интерпретатор

Данная задача требует прежде всего аккуратного программирования.

Количество различных переменных в одной программе не превышает 1000, поэтому можно использовать таблицу имен переменных и таблицу их значений. Программа должна построчно считывать команды из исходного файла и разбирать их в соответствии с условиями задачи. При возникновении ошибок работа должна прерываться с выдачей необходимого сообщения. При обработке переменных должен осуществляться их поиск в таблице и при необходимости — добавление в таблицу нового имени.

Областная олимпиада

10. Раскраска ладей

Покажем сначала, что при любой расстановке ладей достаточно трех цветов. Действительно, если мы будем просматривать доску по строкам, а в каждой строке — слева напра-

во, то очередное поле может биться не более чем двумя из просмотренных (и уже покрашенных) ладей — не более одной слева и не более одной сверху. Значит, ладью, стоящую на этом поле, мы всегда можем покрасить в один из оставшихся цветов. Следовательно, ответом в задаче может быть только 1, 2 или 3 цвета. Пример, когда недостаточно двух цветов, — это множество из нечетного количества ладей, которые бьют друг друга по циклу, — например, если первые две строки исходного файла таковы: 111 в первой строке и 101 во второй.

В силу вышесказанного мы можем решать задачу следующим образом: мы просматриваем доску в естественном порядке (описанном выше) и красим каждую встретившуюся ладью в цвет с наименьшим возможным номером. Если ладьи не бьют друг друга, то мы все их закрасим в первый цвет. Раскраска в два цвета при данном алгоритме выбора цвета будет однозначной, если она вообще возможна. Если же в течение просмотра максимум станет равен трем, то мы сразу прерываем просмотр и выдаем ответ — 3 цвета.

Олимпиады 2001/02 учебного года

Городская олимпиада

Теоретический тур

1. Разделяющий элемент

Решение может быть найдено в результате однократного просмотра элементов массива. Будем вычислять суммы элементов от правого и левого краев массива, для чего заведем два указателя. Сначала левый указатель имеет значение 1, а правый $n + 1$; соответственно, левая сумма изначально равна первому элементу массива, а правая — нулю. Далее сравниваем суммы, сдвигаем указатель с той стороны массива, сумма с которой на данный момент меньше, и увеличиваем значение соответствующей суммы на величину нового элемента. Проделываем описанную операцию до тех пор, пока левый указатель меньше правого. Значения указателей в итоге будут равны индексу разделяющего элемента.

2. Вычисление суммы

За четыре обращения к функции S сумму всех натуральных чисел от N до M, не делящихся ни на 2, ни на 3, можно найти по *формуле включений и исключений*:

$$S(N, M, 1) - S(N + N \bmod 2, M, 2) - S(N + 3 - N \bmod 3, M, 3) + \\ + S(N + 6 - N \bmod 6, M, 6).$$

Однако существует алгоритм, использующий функцию S всего один раз! Заметим, что числа, составляющие искомую сумму, на единицу больше или меньше чисел, кратных шести. Обозначим искомую сумму через SS. Тогда алгоритм, отвечающий на вопрос задачи, выглядит так:

```
SS := 2*S(N + 6 - N mod 6, M, 6);
if N mod 6 = 1 then SS := SS + N;
if M mod 6 = 5 then SS := SS + M;
if N mod 6 = 0 then SS := SS + N + 1;
if M mod 6 = 0 then SS := SS - M - 1;
```

3. Вычеркивание

Для решения пункта (А) заметим, что трехзначных чисел — меньше 1000 штук. Поэтому легко организовать *переборный алгоритм*, который для каждого трехзначного числа проверяет, можно ли получить его из заданного числа вычеркиванием цифр. (Сначала находим, где в первый раз встречается первая цифра; начиная с этого места, ищем вторую цифру и т. д.)

Рассмотрим теперь решение пункта (Б), которое, конечно, подойдет и для пункта (А). Заведем k списков (вначале они будут пустыми). В i-м списке будут храниться все числа, содержащие i цифр, которые можно получить вычеркиванием из заданного. Будем просматривать цифры начиная с первой. Для очередной цифры корректируем списки в обратном порядке: начиная с последнего непустого списка приписываем к каждому элементу рассматриваемую цифру и добавляем в следующий список, если такого числа там еще нет; затем также заносим эту цифру в первый список, если ее там еще нет. Ответом на вопрос задачи будет количество элементов в k-м списке.

4. Клетки таблицы

Задачу можно решить с помощью алгоритма, описанного в 87/88(10). Однако можно применить и более эффективный алгоритм.

1. Если $m = n$, то переходим к п. 2, иначе — к п. 3.
2. Если m — четно, то последнее число будет записано в клетку с координатами $(m/2 + 1, m/2)$, иначе — в клетку $((m + 1)/2, (m + 1)/2)$. Завершаем работу.
3. Если $m < n$, то переходим к п. 4, иначе — к п. 5.
4. Если m — четно, то последнее число будет записано в клетку с координатами $(m/2 + 1, m/2)$, иначе — в клетку $((m + 1)/2, n - (m - 1)/2)$. Завершаем работу.
5. Если n — четно, то последнее число будет записано в клетку с координатами $(n/2 + 1, n/2)$, иначе — в клетку $(m - (n - 1)/2, (n + 1)/2)$. Завершаем работу.

5. Преобразование текста

Ответом на вопрос задачи является следующий список попарно неэквивалентных четырех слов: `пустое слово`; `a`; `b`; `c`. Это следует из двух утверждений:

1) любое слово с помощью данных преобразований можно свести к слову, состоящему не более чем из одной буквы;
2) указанные в ответе четыре слова нельзя перевести друг в друга.

Первое утверждение непосредственно следует из того, что любые две рядом стоящие буквы можно заменить одной (если они разные) или пустым словом (если они одинаковые). При этом длина слова уменьшается, т. е. за несколько таких операций слово «свернется» либо в пустое слово, либо в одну букву. Остается заметить, что отсутствующий переход `ca` ↔ `b` получается так: `ca` ↔ `ccb` ↔ `b`.

Для доказательства второго пункта введем функцию $f(A)$, которая сопоставляет слову A двоичное число 000, 100, 010 или 001 по следующим двум правилам:

1) первый бит результата есть четность количества букв `a` в слове A; второй бит — четность количества букв `b` в слове A; третий бит — четность количества букв `c` в слове A;
2) если в результате окажется больше одной двоичной единицы, то заменяем нули единицами, а единицы — нулями.

Легко видеть, что применение к слову A преобразования, заданного в условии задачи, не меняет значения функции f. Но так как функция f от указанных слов (пустое, a, b, c) имеет разные значения, то от одного из них к другому перейти нельзя.

Таким образом, мы доказали, что имеются четыре класса попарно неэквивалентных слов, и привели список представителей этих классов.

6. Алгоритм

Данный алгоритм находит произведение чисел a и b, используя двоичное разложение числа a.

Докажем, что величина I, равная $ad + c$, является *инвариантом*. Обозначим значения величин на i-м шаге через a_i, d_i, c_i, а на $(i+1)$-м шаге — через a_{i+1}, d_{i+1}, c_{i+1}. Тогда возможны два случая:

1) если $e = 0$,
   то $a_{i+1} \cdot d_{i+1} + c_{i+1} = a_i/2 \cdot d_i \cdot 2 + c_i = a_i \cdot d_i + c_i$;
2) если $e = 1$,
   то $a_{i+1} \cdot d_{i+1} + c_{i+1} = ((a_i - 1)/2) \cdot d_i \cdot 2 + c_i + d_i = a_i \cdot d_i + c_i$.

До исполнения алгоритма $I = ab$; после выполнения алгоритма $I = c$, следовательно после выполнения алгоритма $c = ab$.

7. Рождественские подарки

Сопоставим i-му цвету многочлен $x^0 + x^1 + \ldots + x^{ai}$. Получим k многочленов. Перемножим их. Ответом на вопрос задачи будет коэффициент при x^m.

Доказательство. Одночлен x^m получился после перемножения $x^m = x^{s1} \cdot x^{s2} \cdot \ldots \cdot x^{sk} = x^{s1+s2+\ldots+sk}$, где $0 \leqslant si \leqslant ai$. Биекция между наборами шариков и одночленами вида x^m очевидна (каждый набор соответствует своему x^m). После приведения подобных членов мы и получим количество различных наборов.

8. Ломаная

Данная задача сводится к задаче нахождения количества *граней плоского графа*. Пусть вершины графа — это вершины ломаной и точки пересечения ее звеньев, а ребра графа — отрезки ломаной, соединяющие вышеперечисленные вершины. Очевидно, что полученный граф — *связный* и *плоский*. Зная количество вершин графа (B) и количество его ребер (P), количество граней можно найти по формуле Эйлера: $2 - B + P$. Подсчитать же значение величин B и P достаточно легко.

Практический тур

9. Ограда

Известен факт, что наибольшая площадь многоугольника с заданными сторонами достигается, когда он — вписанный. Таким образом, задача сводится к поиску окружности, в которую можно вписать наш многоугольник.

Во-первых, надо определить, можно ли сложить многоугольник из заданных отрезков. Для этого достаточно проверить *неравенство многоугольника*: наибольшая его сторона должна быть меньше суммы остальных.

Далее с помощью *метода дихотомии* мы подбираем радиус искомой окружности. Для фиксированного радиуса окружности можно попробовать расположить отрезки на окружности; при фиксированном радиусе длина хорды однозначно определяет величину стягиваемой дуги. Если сумма дуг оказывается больше всей окружности (т. е. хорды не помещаются в окружность), то радиус надо увеличить; иначе — уменьшить.

10. Кубики

Необходимым и достаточным условием существования решения данной задачи является совпадение максимумов в первой и второй последовательности чисел. Необходимость этого условия очевидна. Докажем его достаточность и одновременно получим алгоритм вычисления наибольшего количества кубиков, которое могло пойти на строительство сооружения.

1. MAXSUM = 0.
2. Поставим по одному кубику в столбцах на пересечениях строк и столбцов, в которых находятся максимумы. Уменьшим все максимумы на единицу. Тогда $MAXSUM = MAXSUM + R \cdot S$, где R — количество максимумов в первой последовательности, а S — количество максимумов во второй последовательности.
3. Если максимальные значения в последовательностях еще больше нуля, то вернемся к п. 2.

Этот алгоритм конечен, так как максимумы в последовательностях каждый раз уменьшаются на единицу.

То, что полученная расстановка кубиков является максимальной, легко доказывается индукцией по величине максимума в последовательностях.

Нетрудно видеть, что, согласно приведенному алгоритму, в клетку с координатами (i, j) будет поставлено количество кубиков, равное минимуму из чисел, стоящих на i-м месте первой последовательности и j-м месте второй последовательности. Отсюда следует другое, более оптимальное решение. Пусть последовательности чисел хранятся в массивах A и B. Тогда

$$MAXSUM = \sum_{i=1}^{8} \sum_{j=1}^{8} \min(A_i, B_j).$$

Опишем процесс нахождения минимальной суммы. Для начала просто получим для нее оценку снизу. Действительно, каждое число, которое есть хотя бы в одной из последовательностей, должно совпадать с количеством кубиков на одной из клеток доски. Если же число встречается несколько раз, то кратность его появления на доске не меньше максимума из количества повторений этого числа в каждой из двух последовательностей. Например, для последовательностей $(5, 4, 4, 3, 2, 2, 2, 1)$ и $(5, 3, 3, 3, 2, 2, 1, 1)$ минимальная сумма будет не меньше $5 + 4 + 4 + 3 + 3 + 3 + 2 + 2 + 2 + 1 + 1 = 30$, а для последовательностей (5, 4, 4, 3, 3, 2, 2, 1) и (5, 4, 4, 3, 3, 2, 1, 1) минимальная сумма будет не меньше $5 + 4 + 4 + 3 + 3 + 2 + 2 + 1 + 1 = 25$ (заметьте, что за счет повторения чисел оценка улучшилась, хотя общая сумма чисел увеличилась).

Покажем, как можно обеспечить данное количество кубиков. Сначала для простоты изложения переставим строки и столбцы так, чтобы числа в обеих последовательностях шли по убыванию (понятно, что это можно сделать, не ограничивая общности). Первое число в обеих последовательностях одинаково; ставим соответствующее количество кубиков в клетку (1,1) и, вычеркнув первые числа из обеих последовательностей, переходим к рассмотрению последовательностей меньшей длины. Если первые числа в них одинаковые, то ставим соответствующее количество кубиков в клетку (2,2) и вычеркиваем первое число из каждой последовательности. Если же первое число в одной из последовательностей больше, то ставим столбик из этого количества кубиков рядом с последним поставленным столбиком кубиков в тот же ряд доски и вычеркиваем число из последовательности. Таким образом, мы поставим числа в нужные ряды доски, сэкономив при появлении одинаковых чисел, и добьемся совпадения суммы с полученной выше оценкой. Значит, это и есть одна из расстановок с минимально возможной суммой.

Областная олимпиада

11. Таблица

В этой задаче удобно считать, что нумерация строк и столбцов начинается с нуля, а не с единицы (иначе из заданных i и j нужно сначала вычесть 1).

При небольших i и j можно заполнить нужный фрагмент таблицы по определению.

При больших значениях i и j можно заметить следующий факт. На пересечении i-й строки и j-го столбца стоит число i `xor` j (где `xor` — операция побитового исключающего «или»). Докажем это утверждение индукцией по размеру квадрата (более точно, — будем рассматривать квадраты размера $2^i \times 2^i$).

База индукции: для $i = 0$ имеем квадрат размером 1×1; значение $A[0,0] = 0$ `xor` $0 = 0$.

Пусть для $i = k$ утверждение верно, т.е. $A[t,s] = t$ `xor` s при $t,s < 2^k$. Докажем утверждение для $i = k + 1$. Квадрат размера 2^{k+1} состоит из четырех квадратов размера 2^k:

1	2
3	4

Квадрат 1 является исходным квадратом размера 2^k, поэтому для него утверждение верно по предположению индукции.

Рассмотрим квадрат 2. В каждой строке квадрата 1 встречаются все числа от 0 до $2^k - 1$, поэтому в столбцах от 2^k до $2^{k+1} - 1$ эти числа находиться не могут. Минимальные числа, которые там могут быть, — это те же числа, что и в той же строке в квадрате 1, но с установленным битом номер k. Действительно, во всех числах квадрата 1 этот бит сброшен (по свойствам чисел i `xor` j). Минимальность же следует из предположения индукции (квадрат 1 минимален; мы к нему «дописываем» старший бит). Остается заметить, что если к числам квадрата 1 дописать старший бит, то это будет соответствовать t `xor` s для $0 \leqslant t < 2^k$ и $2k \leqslant s < 2^{k+1}$ в силу того, что у числа t этот бит сброшен, а у числа s — установлен.

Для квадрата 3 рассуждения аналогичны, только нужно рассматривать столбцы, а не строки.

Квадрат 4 будет совпадать с квадратом 1. Действительно, старший бит во всех числах квадратов 2 и 3 установлен, поэтому мы можем воспользоваться числами со сброшенным старшим битом. Эти числа и образуют квадрат 1. Его минимальность снова следует из предположения индукции. Этот квадрат соответствует t `xor` s для $2^k \leqslant t < 2^{k+1}$ и $2^k \leqslant s < 2^{k+1}$ в силу того, что у t и s эти биты установлены.

12. Умножение дроби

После считывания данных надо разбить дробь на два массива цифр — отдельно на непериодическую часть и период числа Q. После этого можно реализовать алгоритм умножения непериодической части дроби на число N так же, как выполняется умножение на одну цифру «столбиком» (только «в уме» при этом могут оказываться и многозначные числа). Умножение же периода на число N проводится не совсем аналогично. Важной особенностью умножения периодической дроби на число является добавление после окончания умножения числа, которое останется «в уме», не только к допериодической части, но и к полученному новому периоду. Потом надо проверить результат на периодичность. Заметим,

что длина периода не может увеличиться, но может уменьшиться, хотя дробь все равно останется периодической. Если же получается период, состоящий из одних девяток, то это значит, что получилась конечная дробь.

13. Система дорог

А) Для трех точек (для $N = 3$) решение изложено в задаче 92/93(8).

Б) Пусть $N = 4$. Сначала отметим, что минимальная система дорог должна быть системой прямолинейных отрезков, причем некоторые из них имеют общие концы. Эта система должна быть, по условию, связной; кроме того, в системе не должно быть (в силу ее минимальности) циклов, — в частности, треугольников. То есть это — некоторое *дерево*, включающее четыре исходные точки и, возможно, несколько дополнительных. *Висячими вершинами* этого дерева могут быть только исходные точки. Кроме того, степень любой вершины этого графа не превосходит 3. Действительно, если четыре отрезка имеют общую вершину, то хотя бы один из углов, образованных этими отрезками, не более 90°, а тогда возможна локальная оптимизация суммы расстояний, связывающих три точки этого угла (рис. 49): $FA + FD > AF' + FF' + DF'$. Этот факт использовался в решении задачи 92/93(8).

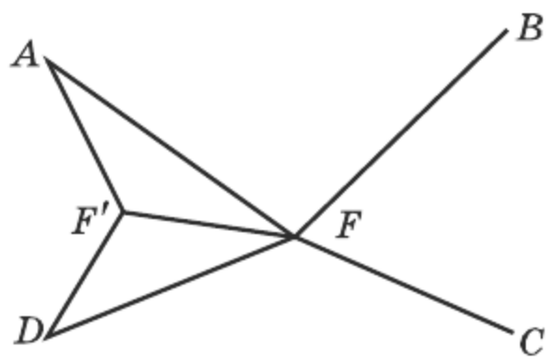


Рис. 49

Ясно, что для любой дополнительной точки степень не может равняться 1 (в этом случае точка просто не нужна) или 2 (в этом случае точку можно «обойти» более прямым путем), т. е. степень каждой дополнительной точки равна 3.

При всех перечисленных условиях дополнительных точек может быть не более двух, иначе висячих вершин будет более четырех.

Итак, для решения нам необходимо разобрать следующие случаи:

1) дополнительных точек нет — решением является *минимальное связующее дерево* с вершинами в заданных точках;
2) дополнительная точка одна — тогда она должна быть связана с тремя точками из четырех заданных и может быть найдена с помощью решения пункта (а) для этих трех точек;
3) дополнительных точек две — тогда каждая из них должна быть связана с другой и с двумя точками из заданных. Пусть, например, одна дополнительная точка связана с точками A и D, а другая — с точками B и C. Как доказывается в решении пункта (а), отрезки, выходящие из дополнительной точки, должны образовывать углы 120°.

Геометрические построения, позволяющие найти дополнительные точки в этом случае, представлены на рис. 50.

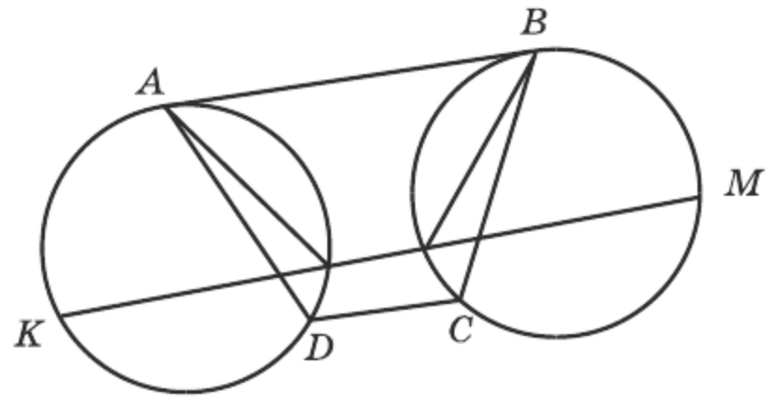


Рис. 50

На сторонах AD и BC строятся две дуги окружностей, из которых отрезки AD и BC видны под углом 120°. Отмечаются середины K и M дуг, дополняющих построенные дуги до полных окружностей. Пересечения отрезка KM с ранее построенными дугами дают искомые точки. Действительно, исходящие из них отрезки расположены под углами 120° друг к другу, — это легко увидеть, рассмотрев две пары вписанных углов, опирающихся на дуги в 120°.

Олимпиады 2002/03 учебного года

Городская олимпиада

Теоретический тур

1. Циклический состав

Квадратичный алгоритм, делающий $O(n^2)$ действий для решения данной задачи, выглядит следующим образом:

- идем по вагонам, каждый раз стираем то, что написано на доске, и пишем свое число (например, номер вагона); нумерацию начинаем с 1;
- если после k шагов встретили «1», то заменяем ее на $k + 1$ и возвращаемся на k шагов;
- если единица, которую мы написали сначала, изменилась на $k + 1$, то цикл найден; если нет, то снова идем вперед и продолжаем нумерацию;
- в конце концов мы либо выйдем из состава, либо найдем цикл (если он есть).

Оценим сложность этого алгоритма. Пусть n — количество вагонов. Тогда в худшем случае (если во всех вагонах стоят единички) мы сделаем $1 \cdot 2 + 2 \cdot 2 + 3 \cdot 2 + \ldots + n \cdot 2 = N \cdot (N + 1) = O(N^2)$ шагов.

Существует алгоритм решения этой задачи и линейной трудоемкости. Он будет выглядеть так:

- идем по вагонам, каждый раз стираем то, что написано на доске, и пишем свое число (как и в предыдущем алгоритме);
- если после k шагов мы встретили «1», то заменяем ее на $k + 1$ и идем еще на k шагов вперед, продолжая нумерацию; если при этом с $k + 1$ до $2k$ вагонов имеют на досках числа от 1 до k, а в вагоне с номером $2k + 1$ стоит число $k + 1$, то меняем его на $2k + 1$ и возвращаемся на k шагов назад;
- если после этого на доске вместо числа $k + 1$ стоит число $2k + 1$, то цикл найден; если нет, то снова идем вперед к вагону $2k + 1$ и продолжаем нумерацию;
- в конце концов мы либо выйдем из состава, либо найдем цикл.

Оценим сложность этого алгоритма. Пусть n — количество вагонов в поезде ($2^{n-1} \leqslant N < 2^n$) либо в цикле. Тогда в худшем случае (если на каждом шаге мы делаем возврат) количество пройденных шагов строго меньше $2 \cdot 2 + 4 \cdot 2 + 8 \cdot 2 + \ldots$ $\ldots + 2^n \cdot 2 = 2 \cdot (2^{n+1} - 2) = 2^{n+2} - 4$. Отношение количества пройденных шагов к количеству вагонов будет меньше, чем $2^{n+2} - 4/2^{n-1} = 8$. Таким образом, алгоритм — линейный.

2. Тасовка

Алгоритм решения пункта (*А*) состоит из последовательности одинаковых шагов. На каждом таком шаге мы снимаем первую половину колоды и вставляем по одной карте из этой снятой части колоды через одну между картами оставшейся части колоды. Тогда после первого шага мы получим 2^{n-1} пар карт, упорядоченных по убыванию. Аналогично, после второго шага мы получим 2^{n-2} четверок карт, упорядоченных по убыванию. Несложно заметить, что через k шагов длина упорядоченных кусков будет равна $2k$, т. е. после n шагов все карты будут упорядочены.

Пример:

0 1 2 3 4 5 6 7 — исходное расположение карт;
4 **0** 5 **1** 6 **2** 7 **3** — после первого шага;
6 **4** 2 **0** 7 **5** 3 **1** — после второго шага;
7 **6** 5 **4** 3 **2** 1 **0** — конечное расположение.

Жирным шрифтом здесь выделены номера карт, которые были вставлены в колоду после очередного шага.

Решение пункта (*Б*). Пусть номера карт в исходном их расположении представлены в виде n-значных двоичных чисел от 00...0, до 11...1. Таких чисел будет ровно 2^n штук. Тогда в конечном расположении карт эти числа образуют какой-то другой порядок.

Назовем «*антитасовкой*» следующее действие с колодой. Вынимаем из колоды несколько (не более половины) карт и, не меняя их порядка, кладем эти карты поверх оставшейся колоды. Заметим, что каждой тасовке в точности соответствует какая-то «антитасовка», являющаяся обратной операцией по отношению к ней.

Покажем, что при помощи таких «антитасовок» можно отсортировать 2^n произвольно лежащих карт за n шагов. Это и будет означать, что при помощи соответствующих им тасовок (являющихся обратными операциями), примененных в обратном порядке, мы решаем поставленную задачу.

Оказывается, что нужной последовательностью «антитасовок» является такая, которая сортирует соответствующую колоде последовательность двоичных чисел, используя *поразрядную сортировку*. Покажем это.

Пусть имеется некоторая неупорядоченная последовательность n-значных двоичных чисел. Выберем все числа, у которых в последнем разряде стоит 0. «Вынем» эти числа из последовательности и, не меняя их порядка, поместим в начале последовательности. На втором шаге применим такую же операцию к числам, у которых 0 стоит в предпоследнем разряде, и т. д. Заметим, что после очередного, k-го шага каждое число A, у которого последние k разрядов меньше, чем у числа B, будет находиться левее числа B. В итоге за n шагов все числа окажутся записанными в порядке возрастания.

3. Шахматный турнир

Опишем классический алгоритм построения описанной в условии задачи цепочки (ряда) участников турнира (*гамильтонова пути в ориентированном графе*). Ключевым здесь является следующее соображение: если уже построена какая-то цепочка с нужным нам свойством, то еще одного шахматиста можно включить в эту цепочку так, чтобы ее основное свойство сохранилось. Действительно, если новый шахматист не выиграл у первого из цепочки, то мы включаем его в начало; если он не проиграл последнему — в конец; иначе — включаем его перед первым шахматистом в цепочке, которому он не проиграл. Таким образом, начав с цепочки из двух произвольных шахматистов, мы по очереди можем включить в нее всех остальных.

4. Барабан

Опишем последовательность действий, приводящих к получению приза. Еще раз отметим, что между действиями барабан вращается, и какие именно выключатели мы проверяем в очередной раз, мы не знаем.

1. Опускаем вниз два противоположных переключателя.
2. Опускаем вниз два соседних переключателя. Если дверь не открылась, то у нас ровно один переключатель направлен вверх.
3. Проверяем противоположные переключатели. При этом возможны две ситуации (рис. 51, проверяемые переключатели обведены рамкой). В первом случае опускаем вниз один переключатель, и мы выиграли. Во втором случае один из переключателей поднимаем вверх. Теперь у нас два соседних переключателя направлены вверх, а два других — вниз.

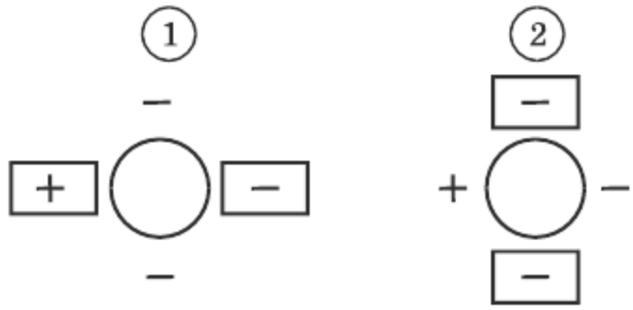


Рис. 51

4. Проверяем соседние переключатели. При этом возможны три ситуации (рис. 52). Переключаем оба проверяемых переключателя. В первом и во втором случае мы выиграли. В третьем случае мы получим ситуацию, когда противоположные переключатели имеют одинаковое положение.

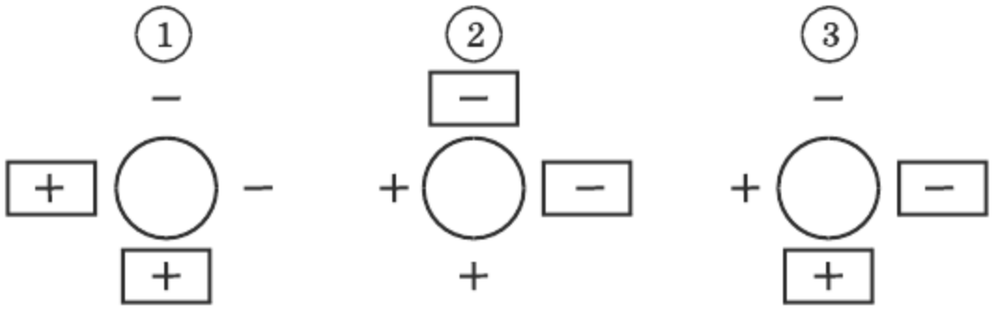


Рис. 52

5. Теперь достаточно переключить противоположные переключатели, и мы выиграли.

5. Функция

Вычислив несколько первых значений функции, мы увидим закономерность: минимумы равны 0 и расположены в точках вида 2^k, а максимумы достигаются в точках вида $3 \cdot 2^k$ и равны 2^k. Для доказательства того, что только эти точки являются точками экстремума, надо рассмотреть три соседних значения F. При этом в точке $2M$ минимум (максимум) достигается только в том случае, если аналогичным свойством обладает точка M.

6. Слова

Задача решается методом *динамического программирования*. Обозначим как $S(k)$ ответ на вопрос задачи (максимальное количество слов в разбиении) для подстроки из первых k символов. Заполним таблицу S по очереди для $k = 0, 1, 2, 3\ldots$: $S(0) = 0$; $S(1)$ равно 0 или 1. При вычислении же очередного значения $S(k + 1)$ мы берем максимум из таких значений $S(i)$, для которых подстрока с номерами символов от $i + 1$ до $k + 1$ образует слово, входящее в словарь. К этому максимуму мы добавляем 1 и записываем результат в $S(k + 1)$. Если же такого слова нет, то полагаем $S(k + 1)$ равным 0. В конце вычислений мы и получим ответ на вопрос задачи. Чтобы получить само разбиение, заведем еще один вспомогательный массив $N(k)$, в который будем при вычислениях записывать либо 0, либо номер i, при котором достигается описанный в рассуждении максимум. Этот номер и будет давать на каждом шаге конец предпоследнего слова.

7. Векторы

Векторы складываются по *правилу параллелограмма*. Если при этом длины сторон равны, то мы получаем ромб, а вектор суммы будет его биссектрисой. Обозначим вектор AB через $\boldsymbol{a}$, вектор AC через $\boldsymbol{b}$, а вектор, направленный вдоль биссектрисы угла BAC, — через $\boldsymbol{e}$ (рис. 53).

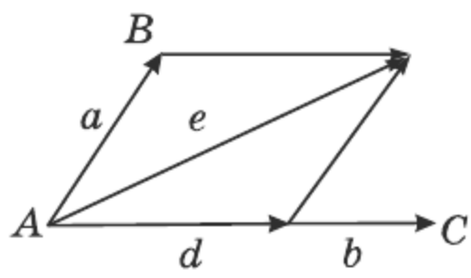


Рис. 53

Тогда первую часть задачи (пункт (А)) можно решить так:

МОДУЛЬ(***a***,x);	x — длина вектора ***a***;
МОДУЛЬ(***b***,y);	y — длина вектора ***b***;
ПРОИЗВЕДЕНИЕ(***b***,x/y,***d***);	вектор ***d*** направлен так же, как вектор ***b***, но имеет ту же длину, что и вектор ***a***;
СУММА(***a***,***d***,***e***).	вектор направлен по биссектрисе угла *BAC*

Программа для решения второй части (пункт (Б)) будет выглядеть так:

МОДУЛЬ(***a***,x);	x — длина вектора ***a***;
МОДУЛЬ(***b***,y);	y — длина вектора ***b***;
ПРОИЗВЕДЕНИЕ(***b***,x/y,***d***);	вектор ***d*** направлен так же, как вектор ***b***, но имеет ту же длину, что и вектор ***a***;
СУММА(***a***,***d***,***e***); ПРОИЗВЕДЕНИЕ(***b***,-1,***d***);	***d*** — вектор, противоположный ***b***;
СУММА(***a***,***d***,***c***); ПРОИЗВЕДЕНИЕ(***c***,-1,***c***)	***c*** — вектор третьей стороны *BC*;
МОДУЛЬ(***c***,z);	z — длина вектора ***c***;
ПРОИЗВЕДЕНИЕ(***c***,x/z,***g***);	вектор ***g*** направлен так же, как вектор ***c***, но имеет ту же длину, что и вектор ***a***;
СУММА(***a***,***g***,***f***);	вектор направлен по биссектрисе угла *ABC*;
РЕШЕНИЕ_СИСТЕМЫ(***e***,***f***,***a***,z,t); ПРОИЗВЕДЕНИЕ(***e***,z,***h***).	разложение вектора ***a*** по векторам ***e*** и ***f***; ***h*** — ответ.

Возможно и другое решение задачи, основанное на свойстве, по которому биссектриса угла делит противолежащую сторону на отрезки, длины которых пропорциональны длинам прилежащих сторон. Для этого варианта составление программы авторы оставляют читателям.

Практический тур

8. Перекраска

Сначала опишем, как при $N = 2^k$ фокусник может получить желаемый результат. В примере 1 к условию задачи написано, как этого добиться за одну команду при двух ячейках. Если у нас уже есть два куска одинакового размера, состоящих из фишек одинакового цвета (может быть, каждый — своего), то, называя по очереди номера первых, вторых и т. д. ячеек из этих кусков, фокусник добьется, что все фишки станут одного цвета.

Другим полезным приемом в данной задаче будет перекраска любой тройки одинаковых фишек в цвет четвертой. Например, если в первых четырех ячейках лежат фишки цветов 1, 2, 2, 2, то, называя ячейки 1 и 2, 1 и 3, 2 и 4, мы получим в результате 4 фишки первого цвета.

Опишем теперь сам алгоритм. Сначала мы делаем с одного края все фишки в количестве 2^k ($2^k \leqslant N < 2^{k+1}$) одного цвета (при $N = 2^k$ алгоритм на этом и заканчивается). Затем действуем таким же образом с другого края; при этом часть фишек из первого куска перекрасится, но для нас важно, что мы получим раскраску в два цвета (а если повезет, то и в один) и будем знать, сколько фишек — первого цвета (слева) и сколько — второго (справа). Если количество фишек первого цвета кратно 3, то с помощью второго из описанных приемов мы перекрасим их во второй цвет и завершим алгоритм. Если остатки от деления количества фишек первого и второго цветов на 3 одинаковые, то мы сначала один или два раза перекрасим по одной фишке каждого цвета в третий цвет, а затем с помощью второго приема выкрасим все в третий цвет. Остается случай, когда куски имеют размеры $3k + 1$ и $3m + 2$. Докажем, что алгоритма перекраски не существует, если общее число фишек кратно 3, и на этом решение задачи будет завершено. Действительно, как нетрудно заметить, разность количества фишек любых двух цветов по модулю 3 не изменяется при перекрасках (т. е. является *инвариантом*). Поэтому если общее число фишек кратно 3 и в начальной раскраске у количеств фишек разных цветов имелись всевозможные остатки от деления на 3, равные 0, 1 и 2, то получить все фишки одного цвета нельзя.

9. Поисковая система

Данная задача носит чисто технический характер. Сначала организуется и анализируется диалог с пользователем (ввод условий фильтрации). На следующем шаге, при необходимости, мы переводим название региона в код с помощью второго из заданных файлов. Затем, после инициализации, начинаем чтение данных, подсчет и запоминание (в файле) списка людей. Выделение данных про каждого человека и проверку условий осуществляем последовательно, а при первой же неудаче переходим к следующей записи.

Областная олимпиада

10. Разделение пограничных городов

Нетрудно показать, что серединный перпендикуляр к отрезку минимальной длины, соединяющему точки разных многоугольников, не пересекается с данными многоугольниками. Этот перпендикуляр и достаточно построить для решения задачи. Кратчайшее же расстояние между многоугольниками всегда достигается между вершиной одного из многоугольников и стороной другого многоугольника (см. задачу 99/00(11)).

11. Кубик на доске

Положение кубика на доске однозначно определяется клеткой доски и ориентацией кубика в пространстве. Ориентацию кубика легче всего задать, указав две его грани, например, нижнюю и левую.

Рассмотрим *граф*, где каждому возможному положению кубика (с учетом клетки и ориентации) сопоставлена вершина, а каждому возможному переходу сопоставлено ребро. Вес ребра при этом определяется числом, написанным на нижней грани кубика. Наличие ребра определяется следующим образом: если из данной клетки и данной ориентации кубика путем перекатывания кубик попадает в соседнюю клетку в соответствующей ориентации, то ребро есть.

Таким образом, задача свелась к поиску *кратчайшего пути* в графе между вершинами, соответствующими начальной клетке и всем возможным ориентациям кубика в конечной клетке.

Для нахождения кратчайшего пути во взвешенном графе можно использовать, например, *алгоритм Дейкстры*.

12. Экзамен в Африке

Пусть $A[i][j]$ обозначает минимальное количество задач для случая, когда имеется i студентов и диапазон сложности равен j. Под диапазоном сложности мы будем понимать $v - u + 1$, где u — сложность заведомо решаемой задачи, а v — сложность заведомо нерешаемой задачи.

Вначале заполним таблицу значением -1. Значения $A[1][j]$ при этом будут равны j, так как для только одного студента нужно проверить все сложности в порядке возрастания. Значения же $A[i][1]$ будут равны 0.

Значения $A[i][j]$ будем вычислять рекурсивно. Если значение еще не вычислено, то оно будет равно $\min(\max(A[i][j-k], A[i-1][k]+1))$, где максимум выбирается по $1 \leqslant k < j$.

Олимпиады 2003/04 учебного года

Городская олимпиада

Теоретический тур

1. Ряд из нулей и единиц

Как легко видеть, последовательно меняя на противоположные значения первого и второго, второго и третьего, ..., тысяча первого и тысяча второго символа последовательности, мы за 1001 ход можем переместить единицу с любого края в центр последовательности. Если мы докажем, что n не меньше 1001, то из условия задачи будет вытекать необходимое нам утверждение.

Заметим, что из условия задачи следует, что за $2n$ ходов мы можем перевести позицию с одной единицей в начале последовательности в позицию с одной единицей в конце. Пусть для этого нам потребовалось X_1 раз менять 1-й и 2-й символ на противоположные; X_2 раз — 2-й и 3-й символы; ... ; X_{2002} раз менять 2002-й и 2003-й символы. Поскольку первый символ в результате изменился с 1 на 0, то X_1 нечетно, а значит не меньше 1. Второй символ остался 0, значит сумма X_1 и X_2 четная, поэтому X_2 тоже не меньше 1. Аналогично доказывается (последовательно слева направо),

что и все остальные X_i тоже нечетные, т. е. не меньше 1. Это означает, что мы сделали не меньше 2002 ходов, или $2n \geqslant 2002$, т. е. $n \geqslant 1001$, что и завершает доказательство.

2. От края до края

Стандартный путь решения задачи — построение *компонент связности* с помощью *волнового алгоритма*. В процессе его работы каждая клетка доски помечается по следующим правилам:

1) выбираем очередную непомеченную клетку вдоль левого края доски и помечаем ее наименьшим не использованным пока натуральным числом K;
2) все непомеченные клетки того же цвета, граничащие с клетками, помеченными числом K, помечаем числом $K + 1$. Если помеченной оказалась клетка у правого края доски, то искомый путь существует. Если ни одна клетка не получила пометку $K + 1$, но на левом краю еще есть не помеченные клетки, переходим к п. 1, иначе констатируем, что искомого пути нет, и завершаем алгоритм;
3) увеличиваем K на 1 и переходим к п. 2.

Заметим, что после достижения правого края доски сам путь легко строится с помощью *обратного хода волнового алгоритма*: достаточно от конечной клетки вернуться в начальную по своему «следу», который можно запоминать вместе с очередной пометкой.

3. Словарь племени Мумбо-Юмбо

Обозначим количество слов длины t, начинающихся на букву А, через $A(t)$, а начинающихся на букву У — через $U(t)$. К каждому слову длины t можно приписать в начале букву А, получив слово длины $t + 1$. Поэтому:

$$A(t+1) = A(t) + U(t).$$

Букву У, по условию, можно приписать только к слову, начинающемуся на А. Поэтому:

$$U(t+1) = A(t).$$

Отсюда следует, что $A(t+1) = A(t) + A(t-1)$. Также отметим, что $A(1) = 1$, $A(2) = 2$. Очевидно, что эти формулы определяют хорошо всем известную *последовательность Фибоначчи*, значения которой легко вычислить.

Ответом на пункт (А) задачи является число $A(n) + U(n)$, т. е. $A(n+1)$.

Ответ на пункт (Б) тоже легко найти. Чтобы определить первую букву слова, стоящего на месте k, сравним k с $A(n)$. Так как в начале словаря идут слова, начинающиеся на букву А, то при $k \leqslant A(n)$ первой буквой является А, иначе — У. Для определения следующей буквы надо k уменьшить на $A(n)$, если слово начинается на У (т. е. «пропустить» $A(n)$ слов, начинающихся на А), уменьшить n на 1 и рекурсивно продолжать поиск следующей буквы.

4. 3 из 5

Для решения сразу обоих пунктов задачи заведем массив P из пяти элементов, в котором будем хранить пять заданных простых чисел, а также массив X целых чисел с номерами элементов от a до b включительно. С помощью элемента массива X с номером k будем подсчитывать, на сколько чисел из заданных пяти делится число k. Содержательный фрагмент программы при этом будет выглядеть так:

```
for i:=a to b do X[i]:=0;
for i:=1 to 5 do
 begin
    k:=p[i];
    while k<a do k:=k+p[i];
       while k<=b do
          begin
             X[k]:=X[k]+1;
             k:=k+p[i]
          end
 end;
 l:=0;
for i:=a to b do
 if X[i]=3 then l:=l+1;
```

5. Ломаная

Ключевым для решения этой задачи соображением является то, что функция $\alpha(|x-a|+x-a)$ слева от точки a равна 0, а справа — она линейная с угловым коэффициентом 2α. Поэтому мы будем добавлять подобные комбинации, перебирая точки слева направо. Каждое следующее слагаемое такого типа не будет «портить» ломаную, построенную слева от очередной точки, а коэффициент при нем мы будем подбирать так, чтобы сумма вместе с очередным слагаемым прошла через следующую из заданных точек. Первое слагаемое мы возьмем равным y_1, а последнее выберем таким, чтобы сумма всех коэффициентов стала равна нулю и, соответственно, последняя часть ломаной получилась горизонтальной (со значением y_n). Намек на предложенное решение давался в примере к условию задачи — в нем специально не были приведены подобные члены.

6. Память

Ответ: 1023,5 Мегабайт.

Очевидно, что 256 32-разрядных (т. е. 32-битных) регистров составляют $256 \cdot 4$ битов, т. е. ровно 1 Кб. Вместе с еще 511 килобайтами кэш-памяти это составит 512 Кб, или 0,5 Мб. Тогда на оперативную память остается $1024 - 0{,}5 = 1023{,}5$ Мб.

7. Нумерация по диагонали

Заметим, что количество элементов на любой диагонали при переходе к следующей возрастает на 1. Номер диагонали K вычисляется по номерам строки и столбца как $i + j - 1$. Поскольку на K-й диагонали имеется K чисел, то на всех диагоналях до K-й включительно будет $1 + 2 + 3 + \ldots + K = K(K + 1)/2$ чисел (так называемые *треугольные числа*), причем внутри диагонали номер числа совпадает с номером строки.

Тем самым, для решения пункта (А) задачи мы сразу находим ответ: $(i + j - 2) \cdot (i + j - 1) + i$.

Для решения пункта (Б) мы последовательно вычисляем треугольные числа и определяем номер диагонали, на которой должно быть расположено искомое число. Остаток покажет номер строки, а зная номер диагонали и номер строки, мы найдем и номер столбца.

8. Фрагменты программ

Фрагменты 1, 2 и 4 эквивалентны между собой и определяют, верно ли, что в массиве *A все* элементы положительны, а фрагменты 3 и 5 эквивалентны между собой и определяют, есть ли в массиве *A хотя бы один* положительный элемент.

Практический тур

9. Банк

Решение задачи определяется так называемым *жадным алгоритмом*. Обосновывать его в данном случае мы не будем, а просто опишем ежедневный процесс принятия решений.

Каждый день во второй половине дня мы сравниваем текущие и уже известные завтрашние курсы обеих валют. Далее действуют следующие правила. Если обе валюты падают, то переводим все в рубли. Если одна валюта падает, а другая растет, то переводим падающую валюту в рубли, а затем все рубли — во вторую валюту. Если обе валюты растут, то выбираем ту, в которой процентное отношение роста больше, и переводим все в нее. В конце периода переводим все обратно в рубли. При переводе не забываем округлять значения до двух знаков после запятой в пользу банка. Основная часть программы при этом выглядит так:

```
for i:=1 to n-1 do
  begin
    readln(fin,zkurs1,zkurs2);
    p1:=zkurs1/kurs1; p2:=zkurs2/kurs2;
    if (p1<=1) and (p2<=1) then
      begin
        if v2>0 then perevod21;
        if v3>0 then perevod31
      end;
    if (p1>p2) and (p1>1) then
      begin
        if v3>0 then perevod31;
        if v1>0 then perevod12;
      end;
```

```
      if (p1<p2) and (p2>1) then
          begin
             if v2>0 then perevod21;
             if v1>0 then perevod13;
          end;
      if (p1=p2) and (p1>1) then
             if v1>0 then perevod12;
      day:=day+1;kurs1:=zkurs1;kurs2:=zkurs2;
   end;
if v2>0 then perevod21;
if v3>0 then perevod31
```

10. Сумма

Для определения *возможности* составления нужной суммы заведем два массива размерами от 0 до $k - 1$ (по числу возможных остатков от деления на k). В них мы будем отмечать единицей числа, которые можно получить, расставляя знаки «+» и «−» между числами, которые мы прочитали из файла данных. Сначала оба массива обнулим и в первом отметим единицей единственную достижимую сумму 0, так как пока не обработано ни одного слагаемого. После считывания из файла числа с номером I будем заполнять второй массив следующим образом: для каждого достижимого числа S после считывания $I - 1$ чисел (оно отмечено единицей в первом массиве) мы отмечаем во втором массиве числа $S + Xi$ и $S - Xi$ (разумеется, операции производятся по модулю K). После заполнения второго массива мы переносим информацию из него в первый массив, второй массив обнуляем, а затем переходим к следующему числу.

Если после считывания всех чисел файла в нулевом элементе второго массива (соответствующего остатку 0 при делении на K) будет стоять 1, то ответ положительный, если 0 — то отрицательный.

Для нахождения нужной *расстановки* знаков нам потребуется еще два массива тех же размеров — в них мы будем накапливать расстановку знаков, с помощью которой достигается данный элемент. Тогда при заполнении во втором массиве ячеек с номерами $S + Xi$ и $S - Xi$ мы будем в соответствующие ячейки четвертого массива заносить строку из

третьего массива с номером S, дописывая «+» или «−» в ее конец. После обработки каждого элемента вторая пара массивов переприсваивается так же, как и первая, а в конце в ячейке с номером 0 будет храниться нужная расстановка знаков.

При больших значениях N и K для экономии памяти третий и четвертый массивы лучше сделать битовыми. Существенно может помочь сэкономить время и память, например, следующее соображение: если в какой-то момент станут достижимы все остатки (а при случайном выборе исходных данных это произойдет очень быстро), то процесс, описанный выше, можно прекратить, все оставшиеся слагаемые взять со знаком «+», а полученный при этом остаток от деления на K компенсировать правильным выбором знаков первых слагаемых. Возможны и другие подходы к оптимизации алгоритма, поиск которых мы оставляем читателям.

Областная олимпиада

11. Калькулятор

Данная задача нередко встречается как вспомогательная при решении сложных олимпиадных задач. Входные данные здесь заданы в строке, поэтому сначала нужно разделить ее на части, сохранив числа и знак операции. В первых подзадачах можно хранить операнды и в числовом формате, но смысл задачи — в организации «длинной арифметики», т. е. действий с числами, которые превышают максимальные размеры разрядной сетки, предусмотренные в числовых форматах данных. При этом числа хранятся либо в виде строк, либо в виде массивов, где каждый элемент хранит, например, одну цифру числа. В обоих случаях вместе с числом полезно хранить и количество значащих цифр в нем.

Второй полезный совет: удобнее хранить цифры в обратном порядке — на первом месте массива — последняя цифра числа и т. д., так как операции сложения, вычитания и умножения выполняются с конца числа (с младших разрядов). Все операции можно реализовать по правилам их выполнения «столбиком», однако умножение быстрее и удобнее реализовать как умножение многочленов, где для

вычисления K-й цифры потребуется вычислить сумму произведений M-й цифры первого числа и $(K-M)$-й цифры второго числа ($M = 0, 1, \ldots, K$) с переносом после этого в старший разряд соответствующего количества единиц.

12. Кладоискатель

Для решения этой задачи мы используем любой из *алгоритмов поиска кратчайшего пути в графе*. Правда, данная задача более сложна, чем классическая задача о поиске пути в лабиринте (или кратчайшего пути в произвольном графе). Во-первых, по исходным данным сначала надо построить сам граф, в котором мы будем искать кратчайший путь. Вершинами этого графа будут являться состояния системы, описанной в задаче, т. е. каждая вершина графа будет задаваться текущими координатами клада и кладоискателя. Ребра же графа будут задаваться возможными ходами (отметим, что из каждой вершины исходит не более четырех ребер). Во-вторых, конечных вершин построенного графа, в которые мы хотим попасть, много — это все вершины, у которых клад находится в нужной ячейке. И наконец, граф надо делать *взвешенным*, т. е. разным ходам (с грузом и без груза) надо придавать разные значения, поскольку в задаче требуется найти не минимальное число ходов, а минимальное число «толчков» (ходов с грузом). Из ограничений, указанных в условии задачи, несложно вывести, что вполне достаточно сделать вес ребра, соответствующего «толчку», в 1000 раз больше веса ребра, соответствующего ходу без груза.

Олимпиады 2004/05 учебного года

Городская олимпиада

Теоретический тур

1. Выборы

Для обоих пунктов задачи ответ положителен, если каждый из кандидатов набрал хотя бы один голос. Приведем общее решение. Назовем ситуацию, при которой за кандидата Ю будет подано k, а за Я — r голосов, *ситуацией* (k, r). Сначала покажем, как можно добиться ситуации, при которой у кандидатов останется по одному голосу: (1,1).

Алгоритм 1.

1. Признаем недействительными такое количество бюллетеней, чтобы аутсайдер остался с одним голосом: $(1,t)$.
2. Если у второго кандидата более одного голоса, то удвоим количество бюллетеней, поданных за аутсайдера, и обнулим у обоих по одному голосу: $(1,t) \to (2,t) \to (1,\ t-1)$.
3. Повторим п. 2 необходимое количество раз.

Теперь покажем, как с помощью обратных операций из ситуации (k,r) можно получить ситуацию (1,1). Пусть $k \leqslant r$. Обратными операциями, очевидно, являются добавление кандидатам одинакового числа бюллетеней и уменьшение вдвое количества бюллетеней, поданных за одного при условии четности этого числа.

Алгоритм 2.

1. Если k — четно и больше 0, то делим k на 2. Если k нечетно, то предварительно добавляем кандидатам по одному бюллетеню. Если $k = 0$, то просто добавляем по одному голосу.
2. Повторяем п. 1 до получения ситуации $(1,t)$.
3. Если t — четно, то делим t на 2. Если нечетно, то предварительно добавляем кандидатам по одному бюллетеню и делим на 2 число бюллетеней обоих кандидатов.
4. Повторяем п. 3 до получения ситуации (1,1).

В итоге пункт (А) решается алгоритмом 1 с последующим уменьшением на 1 количества голосов, отданных за каждого. Пункт (Б) решается последовательным применением алгоритма 1 и обратного к алгоритму 2.

2. Игра в монотонность

Докажем, что ответом задачи является число 4. Для этого приведем стратегию первого игрока, которая независимо от игры второго игрока обеспечивает ему получение монотонной последовательности длины 4, а также стратегию второго игрока, при которой первый не может построить более длинную монотонную последовательность.

Стратегия первого игрока. Своим первым ходом он ставит любое число в середину полоски. Если второй игрок ставит свое число рядом с числом первого или через одну клетку, то своим вторым ходом первый игрок может создать монотонную последовательность длины 3 с пустыми местами

по ее краям, а значит на следующем ходу он достигнет длины 4. Если же второй игрок своим первым ходом ставит число вдалеке от числа первого, то первый своим вторым ходом создает последовательность из двух одинаковых чисел, с каждого края от которой есть не менее двух свободных клеток. В этом случае с помощью второго игрока (за один ход) или без его помощи (тогда — за два хода) первый игрок может превратить два одинаковых рядом стоящих числа в монотонную последовательность длины 4.

Стратегия второго игрока. Второй игрок делит все клетки на пары соседних и эти пары через одну закрашивает в шахматном порядке (1,2), (5,6), (9,10), ..., (97,98) — в черный цвет, а остальные — в белый. Далее он каждым своим ходом ставит число во вторую клетку той же пары, что и первый. При этом он подбирает число так, чтобы в «черной» паре числа шли в порядке возрастания, а в «белой» паре — в порядке убывания. Нетрудно убедиться, что при этом не может возникнуть монотонной последовательности с длиной, большей 4, так как никакие две соседние пары клеток не могут в нее входить целиком.

3. Таксист

Данную задачу лучше решать на предварительно построенном графе. Вершинами в нем служат заказы, а ориентированные ребра обозначают возможность перехода к выполнению очередного заказа после выполнения предыдущего. Так как в этом графе нет циклов, можно применять как хорошо организованный *перебор*, так и метод *динамического программирования*.

4. Упрощаем условия

Вариантов решений здесь может быть несколько. Самый короткий из них имеет вид:

```
if <условие 3>
   then <серия 1>
   else if <условие 1>
        then if <условие 2>
             then <серия 1>
             else <серия 2>
        else <серия 2>
```

5. Числа по кругу

Поскольку мы переставляем числа без их изменения, то различных возможных ситуаций в нашей системе — конечное число. Если доказать, что состояния системы не могут повторяться, то задача будет решена. Для доказательства подобных утверждений в общем случае стараются найти некоторую характеристику системы, которая на каждом шаге монотонно изменяется, так называемый *полуинвариант*. В нашем случае в качестве полуинварианта можно выбрать сумму всех попарных произведений соседних чисел. Действительно, за один шаг сумма $\ldots + ab + bc + cd + \ldots$ превращается в сумму $\ldots + ac + cb + bd + \ldots$, т. е. увеличивается на $ac + cb + bd - ab - bc - cd = a(c - b) + d(b - c) = (a - d)(c - b)$, что, по условию задачи, больше нуля. На этом доказательство заканчивается.

6. Максимум

Обозначим количество чисел через N, сами числа через $A[i]$, а знаки между ними — через $Z[i]$. Для решения задачи применим метод *динамического программирования*.

Заведем квадратную матрицу B, элемент $B[i,j]$ которой будет показывать максимальное значение выражения, которое можно получить с помощью расстановки скобок на участке от числа с номером i до числа с номером j включительно. Тогда число $B[1,n]$ будет ответом к задаче.

Ясно, что чем короче кусочек последовательности, тем легче расставить в нем скобки оптимальным способом. Кроме того, например, для $j = i$, так же как и для $j = i + 1$, расставлять скобки бессмысленно. Поэтому заполнять матрицу B мы будем по диагоналям, начиная от главной диагонали матрицы. При этом элементы самой главной диагонали $B[i,i] = A[i]$, а элементы $B[i,i + 1]$ получаются после применения операции $Z[i]$ к числам $A[i]$ и $A[i + 1]$.

Пусть у нас уже заполнены K диагоналей матрицы B или, иными словами, мы уже решили задачу для всех кусочков нашей последовательности, длина которых не превосходит K. Покажем, как, используя эту информацию, найти ответ к задаче для любого куска последовательности длины $K + 1$.

При любой расстановке скобок одна из операций выполняется последней и разбивает рассматриваемый кусок на два более коротких. Про каждый из них мы можем сказать, какое максимальное значение можно получить с помощью расстановки скобок в нем, поскольку соответствующие элементы таблицы *B* уже заполнены. Остается только устроить перебор по всем операциям в куске, считая их последними, и выбрать максимальный результат.

7. Алгоритм

Первая часть данного алгоритма, известная как *решето Эратосфена*, расставляет единицы в клетках, номера которых являются составными числами. После ее работы нули останутся только в первой клетке и в клетках, номера которых являются простыми числами. Вторая часть алгоритма — это замаскированный подсчет количества нулей, не считая первой ячейки. *Общий ответ*: данный алгоритм вычисляет количество простых чисел в диапазоне от 1 до 10 000.

Практический тур

8. Путь через дорогу

Представим исходные данные задачи в виде графа. Роль вершин в нем будут играть «островки» и два «берега», а ребрами будут соединены те из них, расстояние между которыми меньше или равно величине шага. Это проверяется с помощью простых формул расстояния между двумя кругами и между кругом и прямой (тем более — параллельной одной из координатных осей). После заполнения матрицы смежности из условий задачи нам надо найти кратчайший путь между двумя заданными вершинами построенного графа. Это делается одним из известных алгоритмов, например, *алгоритмом Дейкстры*.

9. Поиск вируса

Для решения задачи существует несколько классических алгоритмов, например, *алгоритмы Рабина*, *Кнута—Морриса—Пратта*, *Бойера—Мура*, которые можно найти в многочисленной литературе и в Интернете.

Олимпиады 2005/06 учебного года

Городская олимпиада

Теоретический тур

1. Тестирование микросхем

Один из самых простых алгоритмов решения данной задачи состоит в том, чтобы каждую микросхему протестировать вместе со всеми остальными. Если загорится лампочка у половины микросхем или более, то данная микросхема неисправна, если меньше, то исправна. Можно сэкономить сравнения, если после обнаружения первой исправной микросхемы повторно протестировать с ней все остальные микросхемы и сделать вывод об их исправности.

Более быстрый, но более сложный алгоритм состоит в том, чтобы тестировать микросхемы попарно. Если хотя бы одна лампочка загорелась, то мы обе микросхемы откладываем в сторону. Хотя бы одна из них точно неисправна, поэтому среди оставшихся снова будет более половины исправных микросхем. Если лампочки не горят, то обе микросхемы одинаковы, и мы в дальнейшем работаем только с одной микросхемой из каждой пары. Таким образом, за $\frac{N}{2}$ сравнений мы уменьшим количество микросхем как минимум вдвое. Повторяя процесс, мы за N сравнений сможем определить хотя бы одну исправную микросхему. Далее с ее помощью мы еще за N сравнений остальных микросхем с исправной определим исправность всех оставшихся микросхем.

2. Набор гирь

Проведем поиск самой легкой монеты по *олимпийской системе*. В первом туре разобьем монеты на пары и взвесим их попарно (4 взвешивания). Оказавшиеся более легкими опять взвесим попарно (2 взвешивания), а затем две самые легкие сравним (1 взвешивание) и определим монету в 5 г.

Затем определим монету в 6 г. Для этого найдем самую легкую монету из трех, которые сравнивались с монетой в 5 г (2 взвешивания), — тяжелее других монет она оказаться

не могла. Проведем 4 взвешивания, положив на одну чашу весов монеты в 5 и 6 г (всего — 11 г), а на другую, по очереди, монеты, которые оказались более тяжелыми в первом туре. Среди них заведомо есть монета в 12 г: она легко определится, так как она — единственная, которая тяжелее 11 г. Если среди взвешиваемых окажется монета в 11 г, то она тоже определится, так как при ее взвешивании весы окажутся в равновесии. Если же она не определилась, значит она оказалась более легкой в первом туре, т. е. это та монета, которая сравнивалась с найденной монетой в 12 г.

Итого нам понадобилось 13 взвешиваний.

Отметим, что можно было бы определить две тяжелые монеты (также по олимпийской системе), но тогда суммарно потребовалось бы 14 взвешиваний. Такое решение оценивалось в 4 балла.

3. Королевский расчет

Ясно, что кратчайший путь короля не может идти в направлениях влево, вверх и влево-вверх.

Для решения задачи лучше всего применить метод *динамического программирования*, при котором мы последовательно заполняем поля шахматной доски, указывая на каждом из них количество кратчайших путей до него (рис. 54).

После первого хода

		1					
	0	1					
1	1	1					

После второго хода

			1				
		1	2				
	1	1	3				
1	1	1	2				
2	3	2	1				

Рис. 54

При этом каждое новое число получается как сумма нескольких уже записанных чисел в соседних клетках (одинаковым цветом фона на рис. 54 выделены результаты после очередного хода). Итоговая таблица будет иметь вид, показанный на рис. 55.

				1	4	14	
			1	3	10	30	
		1	2	6	16	45	
	1	1	3	7	19	51	
1	1	1	2	6	16	45	
2	3	2	1	3	10	30	
5	7	6	3	1	4	15	50
12	18	16	10	4	1	5	

Рис. 55

Ответ: 50.

4. Алгоритм

Заметим, что последовательность команд `b:=a+b`; `a:=b-a`; `b:=b-a` меняет местами значения переменных a и b. Внутренний цикл «перегоняет» первое число в конец массива, а весь алгоритм переставляет первую половину массива как единое целое в его конец.

5. Множество отрезков

Алгоритм решения этой задачи может быть следующим. Упорядочим точки по возрастанию их первых координат, а при равных первых координатах — по возрастанию их вторых координат. Теперь соединим первую точку со второй, третью с четвертой и т. д. Это и будет искомое построение, в чем нетрудно убедиться самостоятельно.

Возможны и другие решения, поиск которых остается за читателями.

6. Копирование файлов

Обозначим через V общий объем копируемых файлов (в байтах), а через $T_0, T_1, \ldots, T_{n-1}$ — записанные Васей прогнозы времени (в секундах). Вычислим оставшиеся объемы копируемой информации V_i и *мгновенные скорости* копирования v_i (байт/с) последовательно при $i = 0, 1, 2, \ldots, n-1$ по формулам: $V_0 = V$; $v_i = V_i / T_i$; $V_{i+1} = V_i - 5v_i$. Тогда минимальную и максимальную скорости мы найдем непосредственно из массива значений v_i, а расположение наиболее трудно (медленно) читаемой информации найдем как $V - V_{N\min}$, где $N_{\min}$ — номер минимума в массиве скоростей.

7. Разнобуквенные слова

Решение этой задачи требует только владения техникой программирования и аккуратности. Необходимо последовательно считывать слова до пробела или точки и проверять их на наличие одинаковых букв. Для этого можно использовать индикаторный массив, в котором отмечается вхождение в слово каждой новой буквы.

Практический тур

8. Автоморфные числа

Отметим важное соображение: если K-значное число автоморфно, то его последние $K-1$ цифр тоже образуют автоморфное число, хотя, возможно, и не $(K-1)$-значное, так как они могут начинаться с нуля. Это позволяет строить цепочки автоморфных чисел, дописывая к уже имеющимся первую цифру.

Исследуя однозначные числа, мы видим, что последней цифрой любых автоморфных чисел могут быть только 0, 1, 5 или 6. Варианты развития цепочек автоморфных чисел с нулем или единицей на конце отметаются уже на стадии двузначных чисел. Остаются числа 25 и 76.

При дальнейшем построении этих двух цепочек автоморфных чисел надо учитывать, что на каком-то этапе первой цифрой может быть и 0. Само построение заключается в переборе различных вариантов первой цифры и проверке числа на автоморфность с помощью *процедур длинной арифметики*.

Обе цепочки продолжаются до 100 знаков (и даже бесконечно). В записи отдельных чисел могут встречаться и нули, поэтому в большинстве случаев возможны два ответа, но при некоторых значениях N возможен всего один ответ. Поэтому для решения задачи необходимо строить обе цепочки.

9. Короткий набор SMS

Сначала считываем исходные данные и составляем таблицу соответствия цифр и букв. Затем считываем слова из словаря, преобразуем их в строку цифр, сравниваем с эталоном и, в случае надобности, выводим в файл ответов, одновременно подсчитывая их количество. После завершения просмотра словаря проделываем необходимые манипуляции с файлом ответов, чтобы в его начале стояло значение счетчика подходящих слов.

Олимпиады 2006/07 учебного года

Городская олимпиада

Теоретический тур

1. Лифт

Если бы было известно, какая кнопка отвечает за поездку вверх, но неизвестно значение N, то мы бы пометили исходный этаж, один раз нажали на кнопку вверх, а далее нажимали бы на кнопку вниз и считали этажи до тех пор, пока не вернулись бы на исходный этаж.

Если бы было известно ограничение на величину N (например, не более 1000), то можно было бы провести две серии испытаний. Сначала пометить этаж, один раз нажать на первую кнопку и до 1000 раз на вторую, а затем наоборот — пометить этаж, один раз нажать на вторую кнопку и до 1000 раз на первую. За это время мы обязательно встретим помеченный в начале серии этаж и узнаем как величину N, так и то, какая кнопка за что отвечает.

Для решения же задачи в общем случае надо проводить па́рные серии испытаний, аналогичных описанным выше, но вместо 1000 брать в очередной паре серий новое предполагаемое ограничение на N. Чтобы наверняка завершить алго-

ритм, на каком-то шаге это ограничение должно превысить N. Логично последовательно увеличивать его в два раза.

Описанный алгоритм является линейным по трудоемкости (можно показать, что количество нажатий на кнопки не превышает числа $8N$). Если же предполагаемые ограничения увеличивать на 1 вместо удвоения, то трудоемкость алгоритма станет квадратичной (порядка N^2).

2. Алгоритм

Алгоритм определяет, есть ли число M среди элементов массива A с номерами от 1 до K.

3. Когда функции равны?

Остатки от деления чисел A и B на число N совпадают тогда и только тогда, когда их разность делится на N. Поэтому число N является делителем разности A и B, а его наибольшее значение равно $|A - B|$. Отдельно надо рассмотреть случай, когда значения A и B совпадают и наибольшего значения не существует, поскольку остатки от деления чисел A и B на любое число N совпадают.

4. Порядок в кавалерии

Ответ: да, можно. Приведем один из возможных алгоритмов заполнения доски. Сначала разобьем всю доску на группы по 4 клетки, связанные ходом коня между собой. Пример такого разбиения половины доски показан на рис. 56.

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

Рис. 56

Теперь покажем, как сделать значение клеток одной четверки равными числам K, $K + 1$, $K + 2$, $K + 3$ при любом фиксированном значении K (ясно, что отсюда будет следовать

общее решение задачи). Для этого K раз увеличим значение первой пары клеток четверки, затем $K + 1$ раз увеличим значение второй пары клеток четверки, а затем еще 2 раза увеличим значение во второй и третьей клетках четверки.

5. Среднее число

Ответ: число X лежит в пределах от 18 до 38.

Пусть $A1$, $A2$, ..., $A11$ — средние числа в пятерках, упорядоченные по возрастанию, а $A6$ — интересующее нас число. Оно больше чисел $A1$, $A2$, ..., $A5$, а те, в свою очередь, каждое больше двух чисел в своих пятерках. Если учесть, что $A6$ больше двух чисел в своей пятерке, то оказывается, что $A6$ больше 17 других чисел. Аналогично, $A6$ меньше 17 других чисел. Пример разбиения чисел на пятерки, когда $A6$ равно 18, приведен на рис. 57.

1	4	7	10	13	16	19	20	21	22	23
2	5	8	11	14	17	24	25	26	27	28
3	**6**	**9**	**12**	**15**	**18**	**29**	**30**	**31**	**32**	**33**
34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55

Рис. 57

Пример, когда $A6$ равно 38, получается из приведенного примера заменой каждого числа n на $56 - n$.

6. Прямоугольники на экране

Существует несколько эффективных алгоритмов подсчета количества областей, использующих тот факт, что области имеют прямоугольный вид. Например, проходя по доске, можно считать все клетки, которые являются левыми верхними углами прямоугольников. Для этого у каждой синей клетки проверим соседние с ней слева и сверху клетки: если они обе белые, то счетчик прямоугольников увеличивается на 1.

7. Цепочка домино

На языке теории графов условие этой задачи означает, что из данного набора костей можно построить *эйлеров путь* (или даже *эйлеров цикл*) в графе, где вершинами являются цифры от 0 до N, а ребрами — кости домино, на которых указаны две соответствующие цифры. Сама теория графов начиналась именно с решения задачи о нахождении эйлерова цикла в графе. *Эйлеров цикл* существует, когда граф связный и степень всех его вершин четная, а *эйлеров путь* существует, если не более двух вершин имеют нечетную степень. (Основные определения и алгоритмы проверки связности можно найти в любой книге по теории графов.)

8. Электронное письмо

Для эффективного поиска пункта, в котором произошла ошибка, следует использовать метод *дихотомии*, или *бинарного поиска*. На каждом шаге мы имеем участок пути, где значения длины сообщения в концах различны (на первом шаге это весь путь). Позвоним в средний пункт этого участка (или одного из двух средних — при четной длине участка) и узнаем длину полученного в нем сообщения. Затем переходим на ту половину участка, где длины сообщений на концах — разные и рекурсивно продолжаем процесс, пока участок не будет состоять из двух соседних пунктов.

Практический тур

9. Зарплата

Для решения задачи сначала надо заполнить массив зарплат, читая данные из файла и выделяя из них нужные нам части. Затем надо найти суммарную и среднюю зарплату по фирме. После этого требуется пройти по массиву второй раз, подсчитывая количество и суммарную зарплату как высокооплачиваемых, так и низкооплачиваемых работников.

10. Вычеркивание цифр

Можно указать два способа решения этой задачи: *комбинаторный* и основанный на *методе динамического программирования*.

Первый способ. Каждая пара цифр 2 и 6 может образовывать число 2006 вместе с любыми двумя нулями, расположенными между ними. Всего таких пар — C_n^2, где n — количество таких нулей. Таким образом, чтобы получить ответ, достаточно для каждой пары (2, 6) подсчитать количество нулей между ними и добавить соответствующее слагаемое к искомой сумме. При хорошей организации вычислений все перечисленные действия требуют $O(n^2)$ действий.

Второй способ. Будем поочередно приписывать к исходной последовательности натуральные числа n начиная с 1. Обозначим через $T2$ количество двоек в построенной последовательности; через $T20$ количество способов получения числа 20 после вычеркивания из построенной последовательности всех цифр, кроме двух; через $T200$ количество способов получения числа 200 после вычеркивания из построенной последовательности всех цифр, кроме трех; через $T2006$ количество способов получения числа 2006 после вычеркивания из построенной последовательности всех цифр, кроме четырех.

При $n = 1$ значения всех введенных переменных равно нулю. Пусть до какого-то n эти значения вычислены. Приписывание очередного числа мы будем производить по одной цифре. При этом если это цифра 2, то увеличиваем значение $T2$ на единицу. Если это цифра 0, то увеличиваем число $T20$ на значение $T2$: это означает, что новый нуль может образовывать число 20 вместе с каждой идущей ранее двойкой. Кроме того, мы увеличиваем число $T200$ на значение $T20$: это означает, что новый нуль может образовывать число 200 вместе с каждой идущей ранее двадцаткой. А если это цифра 6, то увеличиваем число $T2006$ на значение $T200$: это означает, что шестерка может образовывать число 2006 вместе с каждой идущей ранее двухсоткой.

После приписывания к последовательности числа N мы и получаем ответ: это значение $T2006$. Легко заметить, что предложенный алгоритм требует однократного просмотра цифр исходной последовательности.

Областная олимпиада

11. Судоку

Данная задача относится к переборным, и для ее решения хорошо применять *алгоритм поиска с возвратом*, или *backtracking*. Для уменьшения объема перебора полезно использовать вспомогательный массив, в котором учитывать количество вариантов заполнения каждой клетки игрового поля. Если на каком-то шаге это значение обнулится, то это значит, что алгоритм зашел в тупик, и требуется *откат назад*. Продолжать поиск следует с заполнения клеток, в которых количество вариантов минимально.

12. Лифт

Данная задача требует аккуратного моделирования движения лифта по описанным в условии правилам. После реализации остается проверить, какой из возможных вариантов начала движения (их, по условию, не более двух) дает наилучший результат.

Олимпиады 2007/08 учебного года

Городская олимпиада

Теоретический тур

1. Интересное число

Дадим интерпретацию этой задачи в терминах *теории графов*. Пусть цифры от 1 до 9 соответствуют вершинам графа, а ориентированные дуги (стрелочки) указывают, какие пары цифр образуют двузначные числа, равные произведению однозначных. Так, в нашем графе будут присутствовать дуги $1 \to 2$ и $3 \to 2$, так как $12 = 3 \times 4$, $32 = 8 \times 4$, а дуг $1 \to 3$ и $3 \to 4$ не будет.

В построенном графе надо найти путь максимальной длины, проходящий через каждую вершину не более одного раза. Если таких путей несколько, то следует выбрать тот, в начале которого стоят бóльшие цифры.

Анализируя такой граф, несложно заметить, что цифра 9 не может стоять в начале или середине искомого числа,

а перед ней может стоять только цифра 4. Вторая из редко встречающихся в таблице умножения цифр — это цифра 7. И перед ней, и после нее может стоять только цифра 2. Итак, если путь, проходящий через все девять вершин (*гамильтонов путь*) существует, то он начинается с дуги $7 \to 2$, а заканчивается дугой $4 \to 9$. Приведенные соображения сильно сокращают перебор возможных вариантов и приводят к ответу: единственное девятизначное число с указанным в условии свойством — это 728 163 549.

2. Алгоритм

Данная рекурсивная функция вычисляет сумму цифр числа N.

3. Ядерный чемоданчик

Лучшее из известных решений требует 23 нажатий.

Разобьем все кнопки на две группы — первые 3 кнопки и последние 9.

Нетрудно убедиться, что следующие семь нажатий на первые три кнопки обеспечат все возможные комбинации состояний первых трех выключателей: 3, 2, 3, 1, 2, 3, 2. Например, если первые три выключателя были выключены (– – –), то их состояния после каждого нажатия были следующими: (– – –), (– – +), (– + +), (– + –), (+ + –), (+ – –), (+ – +), (+ + +). Если же среди них были изначально включенные, то те же самые состояния повторятся, но в другой последовательности.

Если среди последних девяти выключателей пять или больше находились в положении ВКЛ, то в момент, когда первые три будут включены, будет подан сигнал.

Теперь нажмем по одному разу на последние 9 кнопок. Если среди последних девяти выключателей не было пяти включенных, то пять или больше находились в положении ВЫКЛ, а значит, после наших нажатий они стали включенными. Теперь для подачи сигнала достаточно повторить вышеуказанную серию нажатий на первые три кнопки.

В итоге получаем, что серия нажатий: 3, 2, 3, 1, 2, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 3, 2, 3, 1, 2, 3, 2 обеспечит подачу сигнала тревоги.

4. Определение победителя

Заметим, что длина исходной строки ровно на 1 меньше, чем число N, поскольку после каждого боя выбывает ровно 1 боец. Это позволяет по данной строке определить количество бойцов в исходном строю. Дальше можно решать задачу по-разному, но проще всего промоделировать ход соревнований. Для этого заведем массив A с номерами участников (первоначально $A[i] = i$). Далее по турам (длина тура — N `div` 2, где N — количество оставшихся после очередного тура участников) если в i-й битве тура результат «+», то $A[i] = A[2i - 1]$, иначе $A[i] = A[2i]$. После конца тура меняем значение N на $(N + 1)$ `div` 2 и начинаем все сначала, пока не останется один участник — победитель. Его номер — значение первого числа в массиве A.

5. Блоха на кубе

Решим данную задачу *методом динамического программирования*. Для этого разобьем 8 вершин куба на 4 множества:

- $M1$ — исходная вершина;
- $M2$ — три соседние с ней по ребру;
- $M3$ — три вершины, соседние с исходной по граням, но не соседние по ребру;
- $M4$ — противоположная вершина куба.

Рассмотрим множество задач, похожих на исходную, — сколькими способами можно из исходной вершины за k прыжков попасть в какую-то вершину куба? Обозначим через $T(i,k)$ (где $i = 1, 2, 3, 4$; $k = 1, 2, 3, \ldots$) количество способов достичь вершины из множества с номером i за k прыжков. В силу симметрии куба не важно, о какой именно вершине из данного множества идет речь. В наших обозначениях в исходной задаче требуется найти $T(4,7)$. Ясно, что $T(2,1) = 1$; $T(3,1) = 0$; $T(4,1) = 0$; $T(1,1) = 0$, так как за один прыжок мы можем попасть только в вершины из множества $M2$, причем в каждую только одним способом.

Выведем серию *рекуррентных соотношений*, с помощью которых последовательно заполним таблицу T. Для нахождения $T(4,k)$ разобьем все искомые пути на 3 непересекающих-

ся класса в зависимости от того, прыжок из какой вершины был последним (это может быть любая из трех вершин множества $M3$). Попасть в любую из них можно $T(3,k-1)$ способами, поэтому $T(4,k) = 3T(3,k-1)$. Для нахождения $T(3,k)$ воспользуемся аналогичными соображениями. Все пути разобьем на три класса в зависимости от последнего прыжка. Так как для каждой вершины множества $M3$ он может быть сделан из одной вершины множества $M4$ или из двух вершин множества $M2$, то $T(3,k) = T(4,k-1) + 2T(2,k-1)$. Продолжая подобные рассуждения, получаем соотношения $T(2,k) = T(1,k-1) + 2T(3,k-1)$ и $T(1,k) = 3T(2,k-1)$. С их помощью последовательно заполним первые 7 строк таблицы T:

	$M1$	$M2$	$M3$	$M4$
1 прыжок	0	1	0	0
2 прыжка	3	0	2	0
3 прыжка	0	7	0	6
4 прыжка	21	0	20	0
5 прыжков	0	61	0	60
6 прыжков	183	0	182	0
7 прыжков				546

Ответом в задаче будет число 546.

6. ABBA

Для нахождения ближайшего следующего слова, составленного из этих же букв, нам надо оставлять на месте как можно больше первых букв. Поэтому сначала надо найти самую короткую последовательность последних букв, которая не является упорядоченной по убыванию (т. е. не имеет вида `BB...BAA...A`; для таких слов любые перестановки букв приводят к словам, которые стоят раньше их в словаре). Другими словами, нам надо найти последнее вхождение в наше слово пары букв `AB`. Если такой пары нет, то данное слово является последним в словаре среди всех слов длины $2K$. Если же последняя такая пара найдена, то до нее мы

оставляем слово без изменений, на место буквы A ставим букву B, а все оставшиеся после этого буквы располагаем в порядке AA...ABB...B. Тогда для слова ABBA следующим является слово BAAB.

Рассмотренная задача в другой формулировке является частью задачи *генерации всех сочетаний* C_{2k}^{k} *в лексикографическом порядке* — довольно важного частного случая из класса *переборных задач*.

7. Q Ку

Обозначим координаты данных точек через (x_1, y_1), (x_2, y_2), координаты центра окружности через (x_c, y_c), а ее радиус — через r.

Решение данной задачи может быть математическим или вычислительным.

Математически мы можем задать прямую, содержащую отрезок, параметрически с помощью формул:

$$\begin{cases} x = x_1 + t(x_2 - x_1), \\ y = y_1 + t(y_2 - y_1), \end{cases}$$

подставить эти выражения в уравнение окружности:

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

и решить полученное квадратное уравнение относительно переменной t. Если корней нет, то прямая (а отрезок тем более) не пересекается с окружностью. Если корни есть, то проверяем значения корней. Пересечение будет достигнуто в том и только в том случае, когда хотя бы один из корней лежит в диапазоне от 0 до 1.

Другой вариант математического решения — найти расстояния от точек до центра окружности:

$$\begin{cases} r_1 = \sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2}, \\ r_2 = \sqrt{(x_2 - x_c)^2 + (y_2 - y_c)^2}. \end{cases}$$

Первые два случая совсем просты: если оба расстояния меньше r, то отрезок целиком лежит внутри окружности, и они не пересекаются, а если $r_1 \leqslant r$, $r_2 \geqslant r$ или наоборот,

то отрезок и окружность пересекаются. В случае же, когда оба расстояния больше радиуса окружности, ответ может быть как положительным, так и отрицательным. Для его определения надо найти расстояние от центра окружности до прямой, содержащей отрезок, сравнить его с радиусом окружности и в случае, когда радиус больше (т. е. прямая пересекает окружность), определить, лежат ли точки пересечения внутри нашего отрезка.

При *вычислительном подходе* мы можем перебирать с некоторым шагом точки отрезка, вычислять расстояния от них до центра окружности и сравнивать с радиусом. Простые случаи при этом исследуются легко, а в сложном случае (когда отрезок пересекает окружность в двух точках) можно применять какой-либо из стандартных методов поиска экстремума (для нашей задачи — минимума) функции одной переменной (в нашей задаче — расстояния от точек отрезка до центра окружности) с нужной точностью. После нахождения экстремума делаем вывод и формулируем ответ.

Практический тур

8. Узкое шоссе

Сведем задачу к дискретной — к заполнению некоей таблицы *методом динамического программирования.* Для этого нам потребуются два соображения. Первое из них состоит в том, что каждой машине могут помешать только машины, которые едут медленнее нее: самой медленной машине никто не мешает ехать и она все время едет со своей скоростью; второй по скорости машине может помешать только самая медленная и т. п. Второе соображение заключается в том, что ситуация на шоссе меняется только в точках, где какая-то машина въезжает на шоссе или съезжает с него. Это позволяет нам разбить весь участок на части, отметив точки въезда и съезда машин с шоссе.

Тогда первая часть решения состоит в том, что мы упорядочиваем машины по скорости (не забывая, естественно, их исходную нумерацию) и упорядочиваем все координаты вперемешку (тоже запомнив, конечно, какая координата к какой машине относится). Для реализации этого удобно

использовать *индексные массивы*. Во второй части решения мы построчно, начиная с более медленных машин, заполняем рабочую таблицу размерами $N \times 2N$, где отмечаем время появления каждой машины в каждой из указанных точек (а возможно — и отрицательные величины, если машина не проезжает в данном месте трассы). Заполняя очередную клеточку таблицы, мы сначала рассчитываем время прибытия туда машины со своей скоростью, а потом производим проверки, не должно ли произойти на этом участке трассы каких-то обгонов этой машиной более медленных. В последнем случае мы записываем в заполняемую клеточку максимальное из времен прибытия в данный пункт тех машин, которые текущая машина на текущем участке должна была обогнать. Важно, что мы не рассчитываем точки промежуточных обгонов, поскольку нас не интересуют события на всей трассе, а важны только времена появления каждой машины в каждой из отмеченных точек трассы. Одно из чисел таблицы и является ответом к первой части задачи.

Для определения времени задержки старта первой машины можно просто многократно смоделировать описанный выше процесс с различными временами старта первой машины. При этом разумно использовать *метод дихотомии*, а ограничения на минимальный и максимальный ответ во второй части задачи взять естественные: минимальная задержка равна 0, а максимальная — времени движения первой машины, рассчитанному при решении исходной задачи.

9. Юбиляр

Для решения этой задачи требуется аккуратность и техника. Можно решить задачу за один проход по исходному файлу, но лучше разбить решение на две части. При первом проходе по файлу надо считывать строки, выделять дату, сравнивать ее с юбилейной датой и в случае совпадения выбирать максимальный возраст. После первого прохода мы либо выдаем отрицательный ответ, либо, имея сформированную (лучше — в виде строки) дату рождения старшего юбиляра, проходим по файлу второй раз и выводим в результирующий файл все нужные фамилии.

Олимпиады 2008/09 учебного года

Городская олимпиада

Теоретический тур

1. Поиск числа

Один из вариантов решения задачи состоит из двух этапов. Сначала с помощью *дихотомии* определяем место разрыва массива, затем с помощью сравнения x с первым элементом определяем, в какой части массива (до разрыва или после) надо искать, и ищем x, снова используя *метод дихотомии*.

2. Покупка

Ответ: покупку можно сделать на сумму, равную наибольшему общему делителю чисел m и k. Очевидно, что при любом обмене денег между покупателем и продавцом разность отданных ими денег будет кратна НОД(m,k). На вопрос, сколько монет надо отдать, а сколько — получить в качестве сдачи, чтобы разность составила НОД(m,k), ответ дается в курсе алгебры. Его можно получить перебором, а можно с помощью *алгоритма Евклида* для нахождения НОД.

3. Дорожные работы

Рассмотрим какое-то минимальное по количеству использованных плит покрытие и возьмем первую яму (яму с минимальной координатой). Если плита, покрывающая эту яму, начинается не с края ямы, а левее, то подвинем плиту вправо (в область больших координат) до совмещения с краем ямы на один или два метра. Если при этом сдвиге плита будет упираться в последующие плиты, то сдвинем, соответственно, и их. В результате получится новое минимальное покрытие всех ям, так как все покрытые ямы при сдвиге останутся покрытыми (хотя некоторые, возможно, — другими плитами). Таким образом, минимальное покрытие всегда можно начинать с укладки плиты левым краем на самую левую яму, а следующие непокрытые ямы покрываются аналогично.

4. Агоритм

Данный алгоритм вычисляет X^n.

5. Игральный автомат

Для пункта (А) задачи возможно несколько вариантов решений. Видимо, самым простым из них является последовательное нажатие 1-й и 2-й, 2-й и 3-й, 3-й и 4-й и т. д. кнопок.

Для пункта (Б) работает следующий алгоритм. На первом этапе нажимаем 1, 2, 3 и т. д. кнопки и обязательно находим приз, если его первоначальная координата была нечетной. Если же приз не найден, то значит, его первая координата была четной. В этом случае мы определяем текущую четность позиции приза и нажимаем все кнопки подряд, начиная с 1-й или 2-й.

6. Словарь

Всем известны традиционные словари, хранящие слова в *лексикографическом* порядке. В таком словаре поиск слов осуществляется относительно быстро хорошо известным *методом дихотомии* (деления пополам). Однако этот способ имеет и ряд недостатков, если речь идет о хранении и эффективном поиске слов на компьютере.

Поясним сказанное на примере. Пусть наш словарь содержит следующие слова:

`ар, араб, арба, ас, бал, бар, баркас, барс`

Эти слова можно побуквенно поместить в единый символьный массив, снабдив каждое слово дополнительным символом конца слова.

Для поиска методом дихотомии требуется обращаться к словам, находящимся в середине словаря, а это для слов различной длины является непростой задачей. Поиск значительно упростился бы, если бы все слова имели одинаковую длину, но тогда короткие слова потребуется дополнять незначащими символами (например пробелами), и память компьютера при этом используется недостаточно эффективно. Кроме того, поиск слова на каждом шаге должен сопровождаться побуквенным сравнением очередного слова из словаря со словом, которое мы ищем (с *эталоном*). Если при этом большое количество слов в словаре начинается с тех же самых букв, что и эталон, то приходится выполнять много сравнений, информация о результатах которых совершенно не используется в дальнейшем поиске.

Перечисленные недостатки можно устранить, применяя другой способ хранения слов, позволяющий при достаточно экономном использовании памяти организовывать их эффективный поиск. Этот способ связан с реализацией *иерархической структуры данных — деревьев*. Его суть заключается в следующем.

Для каждой буквы, с которой может начинаться слово, организуется отдельное дерево, корень которого соответствует этой букве. Вершины, являющиеся сыновьями этого корня, должны соответствовать вторым буквам слов и т. д. Для нашего примера такие деревья показаны на рис. 58.

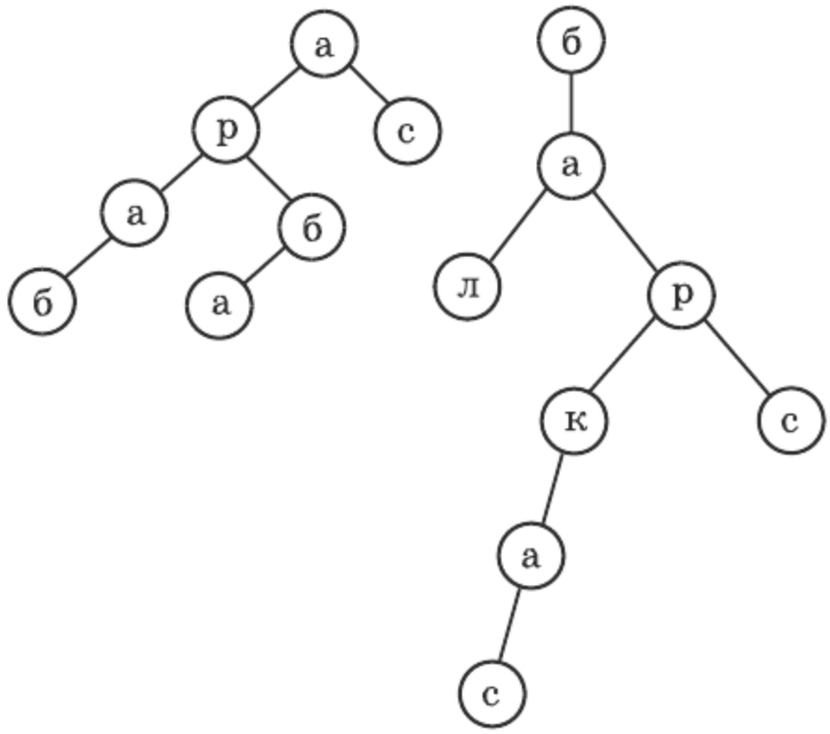


Рис. 58

При такой организации словам соответствуют *пути*, ведущие от корня соответствующего дерева к нижележащим вершинам. При этом слова, начинающиеся с одинаковых последовательностей букв, имеют общую часть пути, что экономит память и уменьшает количество сравнений при поиске.

Реализация таких деревьев в компьютере возможна различными способами, известными по многочисленной литературе о структурах данных. Опишем один из них.

Буквы слов будем записывать в один символьный массив S в следующем порядке:

- сначала записываем все возможные первые буквы слов; окончание этого списка отметим каким-либо «служебным» символом (например «\»);
- затем помещаем в массив списки вторых букв, которые могут следовать за каждой конкретной первой буквой, и т. д.

Параллельно заведем еще один массив I — указательный, в котором будут храниться индексы, указывающие, где именно в первом массиве (основном) надо продолжать поиск, если мы дошли до соответствующей буквы. Необходимо также отмечать, где в дереве заканчивается то или иное слово. Для этого в списке символов, которые могут следовать за данной буквой, надо указать символ, соответствующий концу слова (например «.»).

Таким образом, для нашего примера словаря должны быть организованы следующие массивы (в верхней строке указаны их индексы):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	а	б	\	р	с	\	а	\	.	а	б	\	.	\	л
I	4	7		9	13		15			18	20				22

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
S	р	\	б	\	а	\	.	\	.	к	с	\	\	.	\
I	24		29		31					33	35				

	31	32	33	34	35	36	37	38	39	40
S	.	\	а	\	.	\	с	\	.	\
I			37				39			

Обратим внимание на большое количество фрагментов, состоящих из точки и символа конца списка. Действительно, так заканчивается каждое слово, не являющееся началом другого слова. Для экономии памяти все такие фрагменты можно

объединить, т. е. оставить только в одном месте массива, а все ссылки на такие фрагменты сделать одинаковыми (на это место). Тогда, перенумеровав ячейки, мы получим следующий окончательный вид массивов для нашего словаря:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	а	б	\	р	с	\	а	\	.	а	б	\	.	\	л
I	4	7		9	13		15			18	20				13

	16	17	18	19	20	21	22	23	24	25	26	27	28	29
S	р	\	б	\	а	\	.	к	с	\	а	\	с	\
I	22		13		13			26	13		28		13	

Покажем, как в этом словаре можно найти слово бар.

1. В списке первых букв находим б (в позиции 2).
2. В указательном массиве узнаем, где искать вторые буквы (с позиции 7).
3. Находим букву а и далее продолжаем поиск с позиции 15.
4. Находим в позиции 16 букву р и продолжаем поиск с позиции 22, где находим точку, означающую, что слово найдено.

7. Уравнение

Нетрудно подобрать первые решения уравнения: (1,1), (2,3) и (5,8). Далее можно заметить, что при этом получаются *числа Фибоначчи*; другими словами, из решения (x, y) можно получить новое решение $(x_1 = x + y, y_1 = x + 2y)$. Для доказательства этого наблюдения можно новую пару (x_1, y_1) подставить в уравнение и убедиться, что получается тождество.

Полученные формулы для вычисления нового решения дают способ построения бесконечной последовательности решений.

Замечание. Полученная последовательность порождает все решения уравнения в натуральных числах, что можно доказать, применив соображения, обычно называемые *методом спуска*.

8. Инвестиции

Данная задача решается *методом динамического программирования* с конца таблицы. На i-м шаге мы берем последние i предприятий и находим максимальные прибыли при распределении между ними 0, 1, 2, ..., M тугриков. При заполнении же каждой ячейки мы используем уже заполненный следующий столбец.

Для нахождения самого искомого распределения можно завести еще одну таблицу, которую мы будем заполнять параллельно с таблицей прибылей и отмечать в ней *оптимальное управление* в каждой ситуации, т. е. количество денег, которое надо выделить i-му предприятию.

Практический тур

9. Покраска забора

Назовем первое число в каждой заявке «синим», а второе — «красным». Отсортируем все числа, указанные в заявках подрядчиков. Последовательно просматривая отсортированный массив, будем подсчитывать, сколько «синих» и сколько «красных» чисел нам встретилось. Если первых все время больше, чем вторых, то это означает, что каждую доску (включая данную) есть кому красить. Если же количества чисел оказались одинаковыми, то следующий номер должен отличаться от предыдущего не более чем на 1, иначе следующую доску красить некому. Заметим, что в начале просмотра и «синих» и «красных» чисел поровну (ни одного), — значит, следующая (первая) доска должна присутствовать в какой-то заявке.

10. Система распознавания автомобильных номеров

Для решения этой задачи требуется посимвольно сравнивать считанный автомобильный номер с каждым номером из базы данных. При этом количество позиций, где символы не совпадают, не должно превышать заданного значения.

Заметим, что если бы база данных с автомобильными номерами была отсортирована, то был бы возможен более быстрый поиск. (Как это можно было бы сделать — догадайтесь сами.)

ТЕМАТИЧЕСКИЙ УКАЗАТЕЛЬ

Раздел	Номера задач
Анализ данных	1996/97(8), 1996/97(11), 1998/99(9), 1999/00(9), 2002/03(9), 2005/06(6), 2007/08(9), 2008/09(10)
Арифметика	1987/88(9), 1988/89(3), 1988/89(16), 1989/90(3), 1989/90(14), 1991/92(7), 1999/00(3), 2003/04(11)
Геометрия	1987/88(5), 1987/88(12), 1988/89(7), 1988/89(20), 1989/90(4), 1989/90(8), 1989/90(13), 1989/90(17), 1989/90(18), 1989/90(20), 1990/91(5), 1990/91(7), 1990/91(10), 1990/91(11), 1991/92(1), 1992/93(6), 1992/93(8), 1992/93(13), 1993/94(7), 1993/94(8), 1994/95(8), 1995/96(3), 1995/96(12), 1995/96(17), 1996/97(7), 1998/99(4), 1998/99(13), 1999/00(11), 2000/01(2), 2000/01(8), 2001/02(8), 2001/02(9), 2001/02(13), 2002/03(7), 2002/03(10), 2004/05(8), 2005/06(5), 2007/08(7)
Головоломки	1994/95(2), 2002/03(4)
Графы	1989/90(21), 1991/92(18), 1993/94(13), 1994/95(12), 1995/96(8), 1997/98(10), 1999/00(2), 1999/00(11), 2002/03(3), 2002/03(11), 2003/04(12), 2004/05(3), 2004/05(8), 2006/07(7), 2007/08(1)
Динамическое программирование	1988/89(12), 1991/92(2), 1991/92(18), 1992/93(9), 1992/93(12), 1993/94(18), 1994/95(12), 1995/96(3), 1995/96(8), 1996/97(10), 1997/98(4), 1998/99(15), 1998/99(18), 1999/00(12), 2001/02(3), 2002/03(6), 2002/03(11), 2002/03(12), 2003/04(2), 2003/04(3), 2003/04(10), 2003/04(12), 2004/05(3), 2004/05(6), 2004/05(8), 2005/06(3), 2006/07(10), 2008/09(8)

Раздел	Номера задач
Жадные и однопроходные алгоритмы	1988/89(5), 1991/92(2), 1995/96(10), 1995/96(12), 1996/97(10), 1998/99(7), 1998/99(14), 1999/00(13), 2000/01(10), 2001/02(1), 2003/04(9), 2008/09(2), 2008/09(3)
Игры	1990/91(8), 1990/91(13), 1993/94(17), 1994/95(17), 2004/05(2)
Инвариант	1990/91(6), 1993/94(14), 1998/99(1), 2000/01(6), 2001/02(5)
«Клетчатые» задачи	1987/88(3), 1987/88(6), 1987/88(11), 1987/88(13), 1988/89(6), 1988/89(12), 1988/89(14), 1989/90(5), 1989/90(22), 1990/91(17), 1992/93(7), 1992/93(9), 1992/93(11), 1992/93(12), 1994/95(15), 1994/95(17), 1995/96(1), 1995/96(6), 1997/98(1), 1998/99(15), 2000/01(10), 2001/02(4), 2001/02(10), 2001/02(11), 2002/03(11), 2003/04(2), 2003/04(7), 2005/06(3)
Коды	1989/90(2), 1995/96(2), 1998/99(9)
Комбинаторика:	
• перечисление	1991/92(11), 1992/93(3), 1993/94(11), 1994/95(5), 1995/96(17), 1996/97(2), 1996/97(4), 1996/97(9), 2000/01(3), 2007/08(6)
• подсчет	1995/96(2), 2001/02(3), 2001/02(7), 2003/04(3), 2006/07(10), 2007/08(5)
Конструкции	1991/92(12), 1992/93(16), 1995/96(11), 1999/00(6), 2003/04(5), 2006/07(4)
Логика	1989/90(19), 1990/91(1), 1993/94(12), 1998/99(5), 1998/99(6), 1999/00(2), 1999/00(5), 2001/02(5), 2001/02(10), 2005/06(1), 2005/06(6), 2006/07(1), 2006/07(5)

Раздел	Номера задач
Моделирование	1991/92(5), 1991/92(10), 1991/92(15), 1992/93(1), 1993/94(13), 1994/95(1), 1994/95(9), 1995/96(7), 1995/96(10), 1998/99(16), 2003/04(9), 2006/07(12), 2007/08(4), 2007/08(8)
Монте-Карло	1993/94(4)
Нестандартные идеи и алгоритмы	1987/88(7), 1989/90(7), 1989/90(10), 1989/90(16), 1990/91(1), 1992/93(11), 1993/94(5), 1994/95(11), 1995/96(11), 1995/96(14), 1997/98(3), 1997/98(5), 1998/99(10), 1999/00(5), 2000/01(1), 2001/02(2), 2001/02(11), 2002/03(1), 2002/03(2), 2002/03(8), 2002/03(12), 2003/04(1), 2004/05(1), 2005/06(1), 2005/06(2), 2006/07(1), 2006/07(4), 2007/08(3), 2008/09(5)
Параллельные алгоритмы	1989/90(11), 1992/93(18), 1995/96(11)
Перебор	1987/88(11), 1988/89(5), 1988/89(6), 1991/92(8), 1993/94(9), 1996/97(1), 1997/98(7), 1997/98(10), 1998/99(8), 1999/00(13), 2000/01(8), 2003/04(10), 2005/06(9), 2006/07(11), 2007/08(1)
Полуинвариант	1988/89(13), 1989/90(6), 1991/92(14), 1993/94(1), 2004/05(5)
Программирование	
• для начинающих	1987/88(2), 1987/88(6), 1987/88(10), 1987/88(13), 1988/89(10), 1988/89(14), 1991/92(5), 1999/00(7), 2000/01(7), 2001/02(4), 2003/04(7), 2006/07(9)
• для более опытных программистов	1988/89(16), 1989/90(9), 1989/90(21), 1991/92(7), 1991/92(10), 1992/93(2), 1994/95(1), 1994/95(8), 1996/97(8), 1997/98(8), 1998/99(17), 1999/00(8), 1999/00(9), 2000/01(9), 2002/03(9), 2003/04(11)

Раздел	Номера задач
Распознавание образов	1990/91(18), 1991/92(16), 1994/95(7), 1995/96(16)
Рекурсия	1987/88(8), 1988/89(1), 1988/89(11), 1991/92(6), 1992/93(15), 1993/94(3), 1993/94(14), 1993/94(16), 1994/95(4), 1995/96(4), 1995/96(5), 1995/96(6), 1997/98(6), 1998/99(2), 1998/99(11), 1999/00(1), 2000/01(6), 2002/03(5), 2003/04(3), 2006/07(2), 2008/09(4)
Системы счисления	1989/90(3), 1992/93(10), 1993/94(6), 1995/96(2), 1998/99(2), 1999/00(3), 2000/01(1)
Сортировка и поиск	1988/89(17), 1989/90(21), 1990/91(2), 1991/92(9), 1993/94(13), 1993/94(15), 1995/96(9), 1996/97(8), 1996/97(11), 1999/00(10), 2002/03(2), 2004/05(9), 2005/06(2), 2006/07(8), 2008/09(1), 2008/09(9)
Структуры данных:	
• строки	1991/92(8), 1991/92(11), 1993/94(9), 1993/94(16), 1995/96(8), 1999/00(3), 1999/00(8), 2000/01(7), 2002/03(6), 2004/05(9), 2005/06(7), 2005/06(9)
• массивы	1987/88(10), 1994/95(3), 1995/96(9), 1997/98(4), 1998/99(3), 1998/99(12), 2007/08(4), 2008/09(9)
• стеки, очереди	1988/89(4), 1990/91(12), 1991/92(3), 1991/92(13), 1993/94(2), 1994/95(16)
• списки	1991/92(4), 1994/95(10), 1995/96(3), 1995/96(9), 1996/97(4)
• деревья	1992/93(17), 1994/95(16), 1995/96(5), 1996/97(3), 1998/99(18), 2008/09(6)

Раздел	Номера задач
Тестирование	1992/93(14), 1999/00(4)
Хэш-функции	1987/88(3), 1993/94(5)
Числа и цифры	1988/89(15), 1988/89(18), 1990/91(8), 1991/92(7), 1991/92(16), 1991/92(17), 1991/92(18), 1995/96(13), 1997/98(7), 1998/99(8), 2001/02(12), 2005/06(8)
Численные методы, оптимизация	1988/89(9), 1988/89(19), 1989/90(15), 1991/92(6), 1992/93(7), 1992/93(8), 1993/94(8), 1994/95(15), 1995/96(13), 1996/97(9), 2000/01(4), 2000/01(5)
Шифры	1988/89(10), 1989/90(22), 1990/91(1), 1995/96(16), 1997/98(5)
Эффективность алгоритмов	1987/88(1), 1987/88(7), 1990/91(15), 1990/91(16), 1992/93(11), 1993/94(7), 1994/95(10), 1998/99(7), 1998/99(12), 1998/99(14), 1999/00(6), 1999/00(10), 2001/02(1), 2001/02(2), 2001/02(3), 2002/03(1), 2002/03(8), 2004/05(9), 2005/06(7), 2006/07(1), 2006/07(6), 2006/07(8), 2008/09(1), 2008/09(6), 2008/09(9)

СОДЕРЖАНИЕ

Предисловие 3
Предисловие к книге «Ярославские олимпиады по программированию» 1990 г. 5
Предисловие к книге «Ярославские олимпиады по информатике» 1995 г. 6

Олимпиады 1987/88 учебного года 9
Городская олимпиада 9
Теоретический тур 9
1. Совершенные числа 9
2. Номер дня 9
3. Квадратики 9
4. Последовательность 9
5. Точка в треугольнике 9
6. Соседние клетки 10
Областная олимпиада 10
Теоретический тур 10
7. Цикловоз 10
8. Алгоритм 11
9. Период дроби 11
10. Печать по спирали 11
11. Обход конем 11
Практический тур 11
12. Паук и муха 11
13. Под боем 12

Олимпиады 1988/89 учебного года 13
Городская олимпиада 13
Теоретический тур 13
1. Алгоритм 13
2. Взаимная простота 13
3. Длинное деление 13
4. Без скобок 13
5. Не фунт изюма 14
6. Совпадение 14
7. Выпуклая оболочка 14
8. Произведение многочленов 14
Практический тур 14
9. Кубическое уравнение 14
10. Дешифровка 15

Областная олимпиада 16
Теоретический тур 16
11. Что получится? 16
12. Путь по клеткам 16
13. Конечность алгоритма 16
14. Покрытие 17
15. Нормализация 17
16. Вычисление выражения 17
17. Упорядочение 18
Практический тур 18
18. Римская арифметика 18
19. Арбузы 19
20. Площадь многоугольника 19

Олимпиады 1989/90 учебного года 21
Городская олимпиада 21
Теоретический тур 21
1. Множество 21
2. Минимальное кодирование 21
3. Двоичное умножение 21
4. Клетки в круге 22
5. Робот 22
6. Новобранцы 23
7. Фрагмент программы 23
Практический тур 23
8. Точка в тетраэдре 23
9. Телеграфный перевод 24
10. Перестановка 25
Областная олимпиада 25
Теоретический тур 25
11. Мультипроцессорная система 25
12. Преобразовние таблицы 26
13. Торт 27
14. Сложение с помощью логических операций 27
15. Минимум функции 27
16. Взаимодействие ЭВМ 28
17. Диагональ 29
18. Вершина прямоугольника 29
19. «Странное» свойство ЭВМ 29
Практический тур 29
20. Прямоугольники 29
21. Фрагменты последовательности 30
22. Шифр Кардано 31

Олимпиады 1990/91 учебного года 33
Городская олимпиада 33
Теоретический тур 33
1. Записка Бормана 33
2. Сортировка вагонов 33
3. Вложенный цикл 34
4. Корень 34
5. Четырехугольник и точка 34
6. Фрагмент программы 35
Практический тур 35
7. Часы 35
8. Игра 36
Областная олимпиада 36
Теоретический тур 36
9. Циклы 36
10. Письмо 37
11. Пересечение отрезков 37
12. Скобки 37
13. Игра «Даты» 37
14. Алгоритм 37
15. Последовательность 38
16. Многочлен 38
Практический тур 38
17. Игра «7» 38
18. Имена 40

Олимпиады 1991/92 учебного года 41
Городская олимпиада 41
Теоретический тур 41
1. Бильярд 41
2. Роман 41
3. Польский калькулятор 41
4. Список 42
5. Билеты 42
6. Алгоритм 42
Практический тур 43
7. Умножение «столбиком» 43
8. Кроссворд 43
Областная олимпиада 44
Теоретический тур 44
9. Порядок 44
10. Бензовоз 45
11. Слово 45
12. Схема 45
13. Польская запись 46

14. Алгоритм 46
15. Спички 47
Практический тур 47
16. Числительные 47
17. Степень двойки 47
18. Цепочка 48

Олимпиады 1992/93 учебного года 49

Городская олимпиада 49
Теоретический тур 49
1. Лидеры 49
2. Циклы 49
3. Фестиваль 50
4. Программа 50
5. Сумма квадратов 50
6. Треугольник и точки 50
7. Робот 51
Практический тур 51
8. Минимум 51
9. Путь коня 52

Областная олимпиада 53
Теоретический тур 53
10. Алгоритм 53
11. Таблица 53
12. Маршрут 53
13. Триангуляция 54
14. Клавиатура 54
15. Родственники 55
16. Схема 55
Практический тур 55
17. Родословная 55
18. Параллельные вычисления 56

Олимпиады 1993/94 учебного года 58

Городская олимпиада 58
Теоретический тур 58
1. Перестановки 58
2. Химия 58
3. Фибоначчи 58
4. Вычисление 59
5. Дни в месяце 59
6. Признак делимости 59
7. Кольцевая дорога 59
Практический тур 60
8. Точки в круге 60
9. Ребусы 60

Областная олимпиада 61
Теоретический тур 61
10. Лототрон 61
11. Латинский квадрат 61
12. Дешифровка 61
13. Кулинария 61
14. Алгоритм 62
15. Джонсон & Джонсон 62
16. Подстрока 62
Практический тур 63
17. Игра 63
18. Сумма простых 63

Олимпиады 1994/95 учебного года 65
Городская олимпиада 65
Теоретический тур 65
1. Капитал 65
2. Вирус 65
3. Библиотека алгоритмов 65
4. Альфа 66
5. Сейф 66
6. Алгоритм Маркова 67
Практический тур 67
7. Арифметика по-русски 67
8. Колесо 68

Областная олимпиада 69
Теоретический тур 69
9. «ЧЕГОНАДО» 69
10. Список 69
11. УХТЫ-01 69
12. Лабиринт 70
13. Дисперсия 70
14. Алгоритм 70
15. Сапер 71
Практический тур 71
16. Уравнение по-польски 71
17. Шахматы 72

Олимпиады 1995/96 учебного года 73
Городская олимпиада 73
Теоретический тур 73
1. Хорошая таблица 73
2. Краевая перфорация 73
3. Ковер и моль 74

4. Кольцевая дорога 74
5. Шайка 74
6. Робот 75
7. Кинотеатр 75
Практический тур 76
8. Как из мухи сделать слона 76
9. Перестановка книг 77
Областная олимпиада 79
Теоретический тур 79
10. Автобус 79
11. Сеть 79
12. Прожекторы 79
13. Ранг числа 80
14. Сломанный калькулятор 80
15. НОК 80
Практический тур 80
16. Расшифровка текста 80
17. Фигурки из правильных треугольников 81

Олимпиады 1996/97 учебного года 83
Городская олимпиада 83
Теоретический тур 83
1. Машины 83
2. Смежные перестановки 83
3. Генеалогическое древо 83
4. Гири 84
5. Рациональный умножитель 84
6. Алгоритм 84
7. Три круга 84
Практический тур 85
8. Поиск 85
9. Табло 85
Областная олимпиада 87
10. МПС 87
11. База данных 88

Олимпиады 1997/98 учебного года 90
Городская олимпиада 90
Теоретический тур 90
1. Восстановление таблицы 90
2. Программа 90
3. Роботы-парашютисты 90
4. Удаление элементов массива 91
5. Шифровка 91
6. Алгоритм 91

Практический тур 91
7. «Приятное» число 91
8. Множества 92

Областная олимпиада 93

Практический тур 93
9. Прогрессия из простых чисел 93
10. Пробирки 94

Олимпиады 1998/99 учебного года 95

Городская олимпиада 95

Теоретический тур 95
1. Доска 95
2. Алгоритм 95
3. Матричный порядок 95
4. Зеркало 96
5. Теннисный турнир 96
6. Максимальное произведение 96
7. Максимальная сумма 96
Практический тур 97
8. «Пестрое» число 97
9. Электронное письмо 97

Областная олимпиада 99

Теоретический тур 99
10. Слово 99
11. Алгоритм 100
12. Прогрессия 100
13. Точки в круге 100
14. Невозможная сумма 100
15. Пути коня 101
16. Перевозки 101
Практический тур 101
17. Ноты 101
18. Колье 103

Олимпиады 1999/00 учебного года 105

Городская олимпиада 105

Теоретический тур 105
1. Алгоритм 105
2. Шестеренки 105
3. Делимость 105
4. Система тестов 105
5. Фляга 106
6. Перестановки 106
7. Стадион 107

Практический тур 107
8. Частотный словарь 107
9. Электронная таблица 108
Областная олимпиада 109
Практический тур (день первый) 109
10. Упорядочивание файла 109
11. Архипелаг в океане 109
Практический тур (день второй) 111
12. Атомный реактор 111
13. Грязные предвыборные технологии 111
Олимпиады 2000/01 учебного года 113
Городская олимпиада 113
Теоретический тур 113
1. Последовательность 113
2. Центральная симметрия 113
3. Юстас и Алекс 113
4. Меткий стрелок 113
5. Гигантомания 114
6. Алгоритм 114
7. Буквы 114
Практический тур 114
8. Робот 114
9. Интерпретатор 115
Областная олимпиада 117
10. Раскраска ладей 117
Олимпиады 2001/02 учебного года 118
Городская олимпиада 118
Теоретический тур 118
1. Разделяющий элемент 118
2. Вычисление суммы 118
3. Вычеркивание 118
4. Клетки таблицы 118
5. Преобразование текста 119
6. Алгоритм 119
7. Рождественские подарки 119
8. Ломаная 120
Практический тур 120
9. Ограда 120
10. Кубики 121
Областная олимпиада 121
11. Таблица 121
12. Умножение дроби 122
13. Система дорог 123

Олимпиады 2002/03 учебного года 124
Городская олимпиада 124
Теоретический тур 124
1. Циклический состав 124
2. Тасовка 124
3. Шахматный турнир 124
4. Барабан 124
5. Функция 125
6. Слова 125
7. Векторы 126
Практический тур 126
8. Перекраска 126
9. Поисковая система 127
Областная олимпиада 130
10. Разделение пограничных городов 130
11. Кубик на доске 131
12. Экзамен в Африке 132

Олимпиады 2003/04 учебного года 134
Городская олимпиада 134
Теоретический тур 134
1. Ряд из нулей и единиц 134
2. От края до края 134
3. Словарь племени Мумбо-Юмбо 134
4. 3 из 5 135
5. Ломаная 135
6. Память 135
7. Нумерация по диагонали 135
8. Фрагменты программ 136
Практический тур 137
9. Банк 137
10. Сумма 139
Областная олимпиада 140
11. Калькулятор 140
12. Кладоискатель 141

Олимпиады 2004/05 учебного года 143
Городская олимпиада 143
Теоретический тур 143
1. Выборы 143
2. Игра в монотонность 143
3. Таксист 144
4. Упрощаем условия 144
5. Числа по кругу 144

6. Максимум 144
7. Алгоритм 145
Практический тур 145
8. Путь через дорогу 145
9. Поиск вируса. 146

Олимпиады 2005/06 учебного года 147
Городская олимпиада 147
Теоретический тур 147
1. Тестирование микросхем. 147
2. Набор гирь 147
3. Королевский расчет. 147
4. Алгоритм 147
5. Множество отрезков 148
6. Копирование файлов 148
7. Разнобуквенные слова. 149
Практический тур 149
8. Автоморфные числа 149
9. Короткий набор SMS 149

Олимпиады 2006/07 учебного года 151
Городская олимпиада 151
Теоретический тур 151
1. Лифт 151
2. Алгоритм 151
3. Когда функции равны? 151
4. Порядок в кавалерии 152
5. Среднее число 152
6. Прямоугольники на экране 152
7. Цепочка домино 152
8. Электронное письмо 152
Практический тур 153
9. Зарплата 153
10. Вычеркивание цифр 154
Областная олимпиада 154
11. Судоку 154
12. Лифт 156

Олимпиады 2007/08 учебного года 158
Городская олимпиада 158
Теоретический тур 158
1. Интересное число 158
2. Алгоритм 158
3. Ядерный чемоданчик 158

4. Определение победителя 158
5. Блоха на кубе 159
6. ABBA 159
7. Q ку 159
Практический тур 159
8. Узкое шоссе 159
9. Юбиляр 160

Олимпиады 2008/09 учебного года 162
Городская олимпиада 162
Теоретический тур 162
1. Поиск числа 162
2. Покупка 162
3. Дорожные работы 162
4. Алгоритм 163
5. Игральный автомат 163
6. Словарь 164
7. Уравнение 164
8. Инвестиции 164
Практический тур 165
9. Покраска забора 165
10. Система распознавания автомобильных номеров . . . 166

Решения и рекомендации к решениям 168
Олимпиады 1987/88 учебного года 168
Городская олимпиада 168
Теоретический тур 168
1. Совершенные числа 168
2. Номер дня 168
3. Квадратики 169
4. Последовательность 169
5. Точка в треугольнике 170
6. Соседние клетки 170
Областная олимпиада 170
Теоретический тур 170
7. Цикловоз 170
8. Алгоритм 172
9. Период дроби 172
10. Печать по спирали 173
11. Обход конем 173
Практический тур 175
12. Паук и муха 175
13. Под боем 175

Олимпиады 1988/89 учебного года 176

Городская олимпиада 176

Теоретический тур 176

1. Алгоритм 176
2. Взаимная простота 176
3. Длинное деление 176
4. Без скобок 176
5. Не фунт изюма 177
6. Совпадение 177
7. Выпуклая оболочка 177
8. Произведение многочленов 177

Практический тур 178

9. Кубическое уравнение 178
10. Дешифровка 179

Областная олимпиада 179

Теоретический тур 179

11. Что получится? 179
12. Путь по клеткам 179
13. Конечность алгоритма 180
14. Покрытие 180
15. Нормализация 180
16. Вычисление выражения 181
17. Упорядочение 181

Практический тур 182

18. Римская арифметика 182
19. Арбузы 182
20. Площадь многоугольника 183

Олимпиады 1989/90 учебного года 184

Городская олимпиада 184

Теоретический тур 184

1. Множество 184
2. Минимальное кодирование 184
3. Двоичное умножение 185
4. Клетки в круге 186
5. Робот 187
6. Новобранцы 187
7. Фрагмент программы 188

Практический тур 188

8. Точка в тетраэдре 188
9. Телеграфный перевод 189
10. Перестановка 189

Областная олимпиада 189

Теоретический тур 189

11. Мультипроцессорная система 189
12. Преобразовние таблицы 192
13. Торт 193
14. Сложение с помощью логических операций . . . 193
15. Минимум функции 194
16. Взаимодействие ЭВМ 195
17. Диагональ 195
18. Вершина прямоугольника 196
19. «Странное» свойство ЭВМ 196

Практический тур 197

20. Прямоугольники 197
21. Фрагменты последовательности 198
22. Шифр Кардано 198

Олимпиады 1990/91 учебного года 199

Городская олимпиада 199

Теоретический тур 199

1. Записка Бормана 199
2. Сортировка вагонов 201
3. Вложенный цикл 202
4. Корень 202
5. Четырехугольник и точка 203
6. Фрагмент программы 203

Практический тур 203

7. Часы 203
8. Игра 204

Областная олимпиада 205

Теоретический тур 205

9. Циклы 205
10. Письмо 206
11. Пересечение отрезков 207
12. Скобки 207
13. Игра «Даты» 208
14. Алгоритм 209
15. Последовательность 209
16. Многочлен 210

Практический тур 211

17. Игра «7» 211
18. Имена 212

Олимпиады 1991/92 учебного года 212
Городская олимпиада 212
Теоретический тур 212
1. Бильярд 212
2. Роман 213
3. Польский калькулятор 214
4. Список 215
5. Билеты 215
6. Алгоритм 216
Практический тур 216
7. Умножение «столбиком» 216
8. Кроссворд 216
Областная олимпиада 217
Теоретический тур 217
9. Порядок 217
10. Бензовоз 218
11. Слово 219
12. Схема 220
13. Польская запись 221
14. Алгоритм 223
15. Спички 223
Практический тур 224
16. Числительные 224
17. Степень двойки 224
18. Цепочка 226
Олимпиады 1992/93 учебного года 226
Городская олимпиада 226
Теоретический тур 226
1. Лидеры 226
2. Циклы 227
3. Фестиваль 228
4. Программа 229
5. Сумма квадратов 229
6. Треугольник и точки 230
7. Робот 230
Практический тур 231
8. Минимум 231
9. Путь коня 233
Областная олимпиада 233
Теоретический тур 233
10. Алгоритм 233
11. Таблица 234
12. Маршрут 238
13. Триангуляция 239

14. Клавиатура 240
15. Родственники 241
16. Схема 241
Практический тур 242
17. Родословная 242
18. Параллельные вычисления 242

Олимпиады 1993/94 учебного года 247

Городская олимпиада 247
Теоретический тур 247
1. Перестановки 247
2. Химия 247
3. Фибоначчи 248
4. Вычисление 249
5. Дни в месяце 250
6. Признак делимости 250
7. Кольцевая дорога 251
Практический тур 252
8. Точки в круге 252
9. Ребусы 253

Областная олимпиада 253
Теоретический тур 253
10. Лототрон 253
11. Латинский квадрат 254
12. Дешифровка 254
13. Кулинария 255
14. Алгоритм 256
15. Джонсон & Джонсон 257
16. Подстрока 257
Практический тур 258
17. Игра 258
18. Сумма простых 259

Олимпиады 1994/95 учебного года 260

Городская олимпиада 260
Теоретический тур 260
1. Капитал 260
2. Вирус 260
3. Библиотека алгоритмов 262
4. Альфа 263
5. Сейф 263
6. Алгоритм Маркова 265
Практический тур 266
7. Арифметика по-русски 266
8. Колесо 266

Областная олимпиада 267
Теоретический тур 267
9. «ЧЕГОНАДО» 267
10. Список 268
11. УХТЫ-01 270
12. Лабиринт 270
13. Дисперсия 271
14. Алгоритм 272
15. Сапер 272
Практический тур 275
16. Уравнение по-польски 275
17. Шахматы 276

Олимпиады 1995/96 учебного года 276

Городская олимпиада 276
Теоретический тур 276
1. Хорошая таблица 276
2. Краевая перфорация 277
3. Ковер и моль 279
4. Кольцевая дорога 280
5. Шайка 282
6. Робот 282
7. Кинотеатр 282
Практический тур 284
8. Как из мухи сделать слона 284
9. Перестановка книг 284
Областная олимпиада 285
Теоретический тур 285
10. Автобус 285
11. Сеть 286
12. Прожекторы 287
13. Ранг числа 287
14. Сломанный калькулятор 288
15. НОК 288
Практический тур 288
16. Расшифровка текста 288
17. Фигурки из правильных треугольников 289

Олимпиады 1996/97 учебного года 290

Городская олимпиада 290
Теоретический тур 290
1. Машины 290
2. Смежные перестановки 291
3. Генеалогическое дерево 292
4. Гири 293

5. Рациональный умножитель 294
6. Алгоритм 294
7. Три круга 294
Практический тур 295
8. Поиск 295
9. Табло 296
Областная олимпиада 297
10. МПС 297
11. База данных 298

Олимпиады 1997/98 учебного года 298

Городская олимпиада 298
Теоретический тур 298
1. Восстановление таблицы 298
2. Программа 300
3. Роботы-парашютисты 300
4. Удаление элементов массива 301
5. Шифровка 301
6. Алгоритм 302
Практический тур 302
7. «Приятное» число 302
8. Множества 303
Областная олимпиада 303
Практический тур 303
9. Прогрессия из простых чисел 303
10. Пробирки 304

Олимпиады 1998/99 учебного года 304

Городская олимпиада 304
Теоретический тур 304
1. Доска 304
2. Алгоритм 304
3. Матричный порядок 305
4. Зеркало 305
5. Теннисный турнир 306
6. Максимальное произведение 306
7. Максимальная сумма 307
Практический тур 308
8. «Пестрое» число 308
9. Электронное письмо 308

Областная олимпиада 309
Теоретический тур 309
10. Слово 309
11. Алгоритм 309
12. Прогрессия 310

13. Точки в круге 310
14. Невозможная сумма 311
15. Пути коня 312
16. Перевозки 312
Практический тур 313
17. Ноты 313
18. Колье 313

Олимпиады 1999/00 учебного года 313

Городская олимпиада 313
Теоретический тур 313
1. Алгоритм 313
2. Шестеренки 314
3. Делимость 314
4. Система тестов 315
5. Фляга 316
6. Перестановки 316
7. Стадион 317
Практический тур 317
8. Частотный словарь 317
9. Электронная таблица 318
Областная олимпиада 318
Практический тур (день первый) 318
10. Упорядочивание файла 318
11. Архипелаг в океане 319
Практический тур (день второй) 319
12. Атомный реактор 319
13. Грязные предвыборные технологии 320

Олимпиады 2000/01 учебного года 321

Городская олимпиада 321
Теоретический тур 321
1. Последовательность 321
2. Центральная симметрия 321
3. Юстас и Алекс 322
4. Меткий стрелок 323
5. Гигантомания 323
6. Алгоритм 325
7. Буквы 325
Практический тур 325
8. Робот 325
9. Интерпретатор 326

Областная олимпиада 326
10. Раскраска ладей 326

Олимпиады 2001/02 учебного года 327
Городская олимпиада 327
Теоретический тур 327
1. Разделяющий элемент 327
2. Вычисление суммы 328
3. Вычеркивание 328
4. Клетки таблицы 329
5. Преобразование текста 329
6. Алгоритм 330
7. Рождественские подарки 330
8. Ломаная 331
Практический тур 331
9. Ограда 331
10. Кубики 331
Областная олимпиада 333
11. Таблица 333
12. Умножение дроби 334
13. Система дорог 335
Олимпиады 2002/03 учебного года 337
Городская олимпиада 337
Теоретический тур 337
1. Циклический состав 337
2. Тасовка 338
3. Шахматный турнир 339
4. Барабан 340
5. Функция 341
6. Слова 341
7. Векторы 341
Практический тур 343
8. Перекраска 343
9. Поисковая система 344
Областная олимпиада 344
10. Разделение пограничных городов 344
11. Кубик на доске 344
12. Экзамен в Африке 345
Олимпиады 2003/04 учебного года 345
Городская олимпиада 345
Теоретический тур 345
1. Ряд из нулей и единиц 345
2. От края до края 346
3. Словарь племени Мумбо-Юмбо 346
4. 3 из 5 347
5. Ломаная 348

6. Память 348
7. Нумерация по диагонали 348
8. Фрагменты программ 349
Практический тур 349
9. Банк 349
10. Сумма 350
Областная олимпиада 351
11. Калькулятор 351
12. Кладоискатель 352

Олимпиады 2004/05 учебного года 352

Городская олимпиада 352
Теоретический тур 352
1. Выборы 352
2. Игра в монотонность 353
3. Таксист 354
4. Упрощаем условия 354
5. Числа по кругу 355
6. Максимум 355
7. Алгоритм 356
Практический тур 356
8. Путь через дорогу 356
9. Поиск вируса 356

Олимпиады 2005/06 учебного года 357

Городская олимпиада 357
Теоретический тур 357
1. Тестирование микросхем 357
2. Набор гирь 357
3. Королевский расчет 358
4. Алгоритм 359
5. Множество отрезков 359
6. Копирование файлов 360
7. Разнобуквенные слова 360
Практический тур 360
8. Автоморфные числа 360
9. Короткий набор SMS 361

Олимпиады 2006/07 учебного года 361

Городская олимпиада 361
Теоретический тур 361
1. Лифт 361
2. Алгоритм 362
3. Когда функции равны? 362
4. Порядок в кавалерии 362
5. Среднее число 363

6. Прямоугольники на экране 363
7. Цепочка домино. 364
8. Электронное письмо 364
Практический тур 364
9. Зарплата 364
10. Вычеркивание цифр 364
Областная олимпиада 366
11. Судоку 366
12. Лифт 366

Олимпиады 2007/08 учебного года 366

Городская олимпиада 366
Теоретический тур 366
1. Интересное число. 366
2. Алгоритм 367
3. Ядерный чемоданчик. 367
4. Определение победителя 368
5. Блоха на кубе 368
6. ABBA. 369
7. Q Ку 370
Практический тур 371
8. Узкое шоссе 371
9. Юбиляр 372

Олимпиады 2008/09 учебного года 373

Городская олимпиада 373
Теоретический тур 373
1. Поиск числа. 373
2. Покупка. 373
3. Дорожные работы 373
4. Алгоритм 373
5. Игральный автомат 374
6. Словарь 374
7. Уравнение 377
8. Инвестиции 378
Практический тур 378
9. Покраска забора 378
10. Система распознавания автомобильных номеров 378

Тематический указатель 379

Серия «Развитие интеллекта школьников»

Учебное издание

Волчёнков Сергей Геннадьевич,
Корнилов Петр Анатольевич,
Белов Юрий Анатольевич,
Дашниц Наталия Леонидовна,
Никулин Вадим Андреевич,
Заводчикова Надежда Ивановна

ЯРОСЛАВСКИЕ ОЛИМПИАДЫ ПО ИНФОРМАТИКЕ

Сборник задач с решениями

Ведущий редактор *Д. Усенков*
Художники *С. Инфантэ, Н. Новак*
Художественный редактор *О. Лапко*
Технический редактор *Е. Денюкова*
Корректор *Л. Макарова*
Компьютерная верстка: *С. Янковая*

Подписано в печать 21.01.10. Бумага офсетная
Формат 60 x 90 1/16. Гарнитура Школьная. Усл. печ. л. 25,5
Печать офсетная. Тираж 2 000 экз. Заказ

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