А. Е. Анисимов, В. В. Пупышев

Сборник заданий по основаниям программирования

Оглавление

Предисловие

Вашему вниманию предлагается «Сборник задач по основаниям программирования». Этот задачник можно рассматривать, как приложение к книге Н.Н. Непейводы и И.Н. Скопина «Основания программирования» [5]. Целью задачника является практическое закрепление материала, изложенного в [5], освоение новых идей в программировании, их утверждения, дополнения или (даже!) опровержения. Авторы сборника постарались сохранить дух книги, цели и задачи, для которых книга предназначена. Также структура задачника полностью соответствует структуре указанной книги. Следует сказать, что издательство ИНТУИТ.РУ («Интернет-Университет Информационных Технологий») выпустило в свет учебные пособия Н. Н. Непейводы «Стили и методы программирования» [6] и И. Н. Скопина «Основы менеджмента программных проектов» [11], вместе с которыми данный сборник также может совместно использоваться.

В сборнике представлены более тысячи задач самой разной тематики и уровня сложности. Порядок следования задач в разделах не всегда соответствует субъективному уровню их сложности. Некоторые из задач являются несложными и, можно даже сказать, тривиальными. Однако, как и всюду в программировании, даже в простейших на первый взгляд задачах можно (и нужно!) ожидать массу «подводных камней». Поэтому, мы можем рекомендовать читателям, которые решают «простую» на их взгляд задачу, искать в первую очередь не только (и не столько) её решение, сколько рассуждать о возможных неявных сложностях, скрытых особенностях, которые могут возникнуть при реализации её решения, отладке и тестировании.

О тестах нужно сказать отдельно. Конечно, в идеале, любую задачу на программирование её автор должен снабжать системой тестов, позволяющих оценить качество решения в самых разных (в том числе — экстремальных) ситуациях. Кроме этого, тесты во многом дополняют и уточняют постановку задачи, могут подсказать идеи будущего решения, предостеречь от некоторых ошибок и ляпов. Вследствие этого, разработ-

ка тестов является подчас более сложной проблемой, чем создание самой задачи и даже её решения. Но, к сожалению, тексты тестов существенно увеличивает объём постановки задачи, и, как следствие размер всего задачника, поэтому авторы просто не в состоянии приводить здесь тесты для каждой задачи. Некоторые задачи сборника снабжены нами тестами (см. во второй части); для других задач авторы рекомендуют читателям самостоятельно разработать систему тестов, это будет еще одним, дополнительным и очень полезным упражнением.

Вопрос о существовании решения отдельных задач авторы оставляют открытым. То же самое относится к эффективности возможных алгоритмов. Некоторые задачи даются с единственной целью — заинтересовать читателя некоторыми новыми понятиями и технологиями.

В некоторых задачах приводится ограничение на время исполнения программы при определенных характеристиках вычислительной техники (тактовая частота). Безусловно, эти ограничения носят относительный характер и дают автору решения дополнительную информацию к размышлению о выборе и оптимизации алгоритма и структур данных.

Ряд задач сборника предлагался на различных олимпиадах по программированию от школьного, районного уровня до уровня международных олимпиад. Олимпиадные задачи имеют свою специфику. Во-первых, они ориентированы на особые, специальные условия, в которых участник строит свое решение: ограничение по времени, жесткие условия на входные и выходные данные, временны̀е характеристики программы и другие. Во-вторых, оценка решения производится группой экспертов (жюри), которая разрабатывает систему оценивания, основанную на собственных представлениях о качестве программного решения. Чаще всего оценка пропорциональна количеству и сложности пройденных программой тестов, разработанных автором задачи или жюри, хотя нередко используются и другие критерии при оценке программ. В-третьих, дух олимпиады предполагает соревновательность, соперничество между участниками, поэтому часто важнее создать не идеальную программу, находящую всегда правильное решение (что в олимпиадных условиях бывает сделать, как правило, невозможно), а программу, превосходящую в чем-то своих конкурентов. Авторы не стали (в большинстве случаев) как-либо адаптировать тексты олимпиадных задач для данного сборника, а привели их «как есть». Просим читателя учитывать всё вышеизложенное, и, в особенности, стиль и структуру постановки олимпиадных задач. Во второй части сборника для некоторых олимпиадных задач, помимо тестов, приведены так называемые ЧаВО (Часто задаваемые Вопросы и Ответы на них), помогающие при решении задачи.

Приведем здесь условные обозначения, которые авторы сборника при-

меняли для задач различных олимпиад по программированию:

- задачи районных олимпиад Удмуртской Республики;
- задачи заочно-очных олимпиад, проводимых среди студентов и школьников Удмуртским государственным университетом при спонсорской поддержке в виде призов; подробнее о правилах можно узнать на сайтах http://colymp.da.ru или http://colymp.udsu.ru;
- задачи олимпиад, помеченные как Б, автором которых является профессор, д. ф.-м. н. Анатолий Петрович Бельтюков;
- задачи олимпиад, помеченные как Н, автором которых является профессор, д. ф.-м. н. Николай Николаевич Непейвода; все особенности авторской стилистики полностью сохранены;
- задачи олимпиад, помеченные как О, автором которых является Олег Анатольевич Морозов;
- задачи из других источников; координаты источника указаны в самих задачах.

Дадим несколько полезных ссылок на интернет-ресурсы, содержащих большое количество материалов по задачам на программирование:

а) Олимпиады по информатике, СПбИТМО (http://neerc.ifmo.ru);
б) КОМП — конкурсы и олимпиады по машинному программированию (http://colymp.da.ru);
в) коллекция олимпиадных задач (http://algolist.manual.ru/olimp).

Авторы выражают благодарность за оказанную помощь, методическую и моральную поддержку Николаю Николаевичу Непейводе и Анатолию Петровичу Бельтюкову.

Сборник заданий предназначен для специалистов, преподавателей, студентов, школьников и всех тех, кто интересуется фундаментальными основами программирования и стремиться практически их освоить.

Авторы искренне надеются, что сборник поможет читателям практически освоить многие методики, стили программирования, более основательно закрепить теоретические знания, почерпнутые в [5, 6, 11]. Конечно, подобные издания не могут быть полностью ограждены от ошибок, поэтому, авторы будут благодарны всем, кто сообщит об обнаруженных опечатках, ошибках, некорректностях авторам по адресам pvv@uni.udm.ru или aae@ulm.uni.udm.ru.

Часть I

Задачи

Глава 1

Введение в систему понятий программирования

1.1. Короткий привет. Напишите самую короткую программу, печатающую строку "Hello World!". Используйте для этого самый лаконичный язык.

1.2. Длинный привет. Напишите самую длинную программу, печатающую строку "Hello World!". Программа должна быть такой, чтобы нельзя было убрать ни одного символа без изменения работоспособности. Можно использовать подходящие системы программирования.

1.3. Задача о волке, козе и капусте. Один мужик купил на базаре капусту, волка и козу. Чтобы добраться домой ему нужно перебраться через реку на лодке. В лодке могут поместиться только мужик и одна из «покупок». Нельзя оставлять без присмотра волка и козу или козу и капусту. Иначе волк съест козу или коза съест капусту. В каком порядке должен перевезти мужик свои «покупки», чтобы они все остались целыми.

1.4. Следы.

Будем изображать следы ног человека с помощью обычных символов

```
.oooO  Oooo.
(   ) (   )
 \ (   ) /
  \_) (_/
```

Рис. 1.1: Следы

на клавиатуре следующим образом (рис. 1.1).

Задача заполнить следами N человек, идущих рядом и проходящих путь в S полушагов. $S < 12$; $N < 12$.

Пример

Для $N = 3$ и $S = 5$.

```
.oooO     .oooO     .oooO
(   )     (   )     (   )
 \ ( Oooo. \ ( Oooo. \ ( Oooo.
  \_)(   )  \_)(   )  \_)(   )
.oooO ) / .oooO ) / .oooO ) /
(   )(_/  (   )(_/  (   )(_/
 \ ( Oooo. \ ( Oooo. \ ( Oooo.
  \_)(   )  \_)(   )  \_)(   )
.oooO ) / .oooO ) / .oooO ) /
(   )(_/  (   )(_/  (   )(_/
 \ (       \ (       \ (
  \_)       \_)       \_)
```

Начало движения всегда с левой ноги снизу вверх. Шаги и расстояния между людьми всегда такие, как в примере. Программа должна работать так, чтобы команда

ИМЯ ИСПОЛНЯЕМОГО МОДУЛЯ > foot.txt

записывала результат в файл *foot.txt*

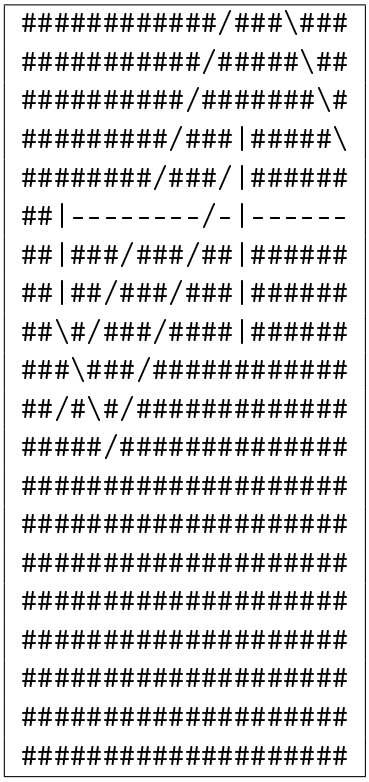
1.5. Простые дроби. Дано натуральное n. Вывести все различные обыкновенные несократимые дроби, принадлежащие интервалу от 0 до 1, знаменатели которых не превышают n.

1.6. ✠ Черепашья графика. На символьном экране 20×20 можно рисовать с помощью символов `'-'`, `'|'`, `'/'`, `'\'`. Командой является строка из латинских букв и цифр. Ниже приведен пример, дающий полное представление о командах и их возможностях.

Пример

`Вход:` `0310r13R8rR20l14Lr04L9rrr4LL1lRlRR3`

На экране:



— обозначение пробела.

Техническое требование

Написать программу, запрашивающую команду и рисующую на экране то, что описывает команда.

1.7. ✣ **Делители.** Для заданного N найти натуральное число, не превосходящее его и имеющее максимальное количество разных делителей. Если таких чисел несколько, вывести самое маленькое из них.

Техническое требование

N — натуральное не больше 2 000 000 000.

Пример

```
N= 100
Ответ: 60
```

1.8. ✣ **Линейное уравнение.** Задано арифметическое выражение с помощью операций '+','−', круглых скобок, целых чисел и переменной X. Задача, выяснить, при каких значениях X значение выражения будет равно 0.

Техническое требование

Выражение задаётся строкой с клавиатуры. Длина строки не более 80 символов. Сумма любых чисел в выражении не превышает 2 000 002 001.

Ответ выдавать в виде обыкновенной дроби со знаком, где числитель и знаменатель разделены символом '/'. Числитель и знаменатель не должны иметь общих делителей, кроме единицы.

Пример

```
Выражение: -(-(X+(-50))-(X-6))
X=+28/1
```

1.9. Сокращение выражения. Задано арифметическое выражение с помощью операций '+', '−', '∗' и круглых скобок. Записать выражение эквивалентное заданному, но без скобок и с минимальным количеством операций. Выражения эквивалентны, если при любых значениях одинаковых переменных в них они дают одинаковые значения. Заглавные и строчные буквы не различаются.

Пример

```
Исходное выражение: ((123-X)*(2+3))-(X-(-Y))
Результат: 615-6*X-Y
```

1.10. ✣ Обратный алгоритм. Задан алгоритм.

```
Вход:  In
  B = In;   i = 100;
  пока i > 0
  нц
    B = целая часть числа(101 * B / 100);
    i = i - 1;
  кц
  Res = B
Выход: Res
```

Написать программу, которая по выходным (Res) данным будет давать входные (In).

Техническое требование

Входные и выходные данные — натуральные числа не больше 2 000 000 000.

Пример

```
Res= 5330
In=2001
```

1.11. ❐ Превращение программы в почтовое сообщение. Написать программу, печатающую свой собственный исходный текст с добавлением в начало каждой строки символа '>'. Например, Ваша программа

была написана на каком-то языке программирования. Затем её оттранслировали. Результатом работы оттранслированной программы должен быть исходный текст Вашей программы с добавленными в начало каждой строки символами '>'.

Техническое требование

Программа должна работать без внешних модулей и файлов. Результат работы записывается в файл *.TXT в текущей директории, где * — имя Вашей программы. Например, программа INPUT.PAS должна выдавать файл INPUT.TXT

1.12. Функция Аккермана. Написать программу для вычисления функции:

$$A(m,n) = \begin{cases} n+1, & m = 0 \\ A(m-1,1), & n = 0, m \geqslant 0 \\ A(m-1, A(m,n-1)), & n, m > 0 \end{cases}$$

Данная функция работает очень медленно. Допустимые m и n определите сами.

Пример

```
A(1,2) = 4
```

1.13. Числа Фибоначчи. Вычислить значение функции от натурального x:
$f(1) = 1$,
$f(2) = 1$,
$f(x) = f(x-2) + f(x-1)$, если $x > 2$.

x будет такой, что $f(x) < 2\,000\,000\,000$.

Пример

```
x= 6
f(6)=8
```

1.14. Быстрая степень. Построить программу, вычисляющую число X в степени n. X и n — натуральные числа такие, что $X^n < 2\,000\,000\,000$.

Пример

```
X= 2
n= 10
1024
```

1.15. Выпуклая оболочка. На плоскости задано множество точек.

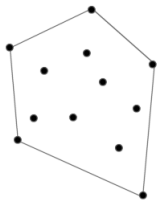


Рис. 1.2: Выпуклая оболочка

Нужно соединить некоторые из них так, чтобы получился замкнутый выпуклый многоугольник и все остальные точки лежали внутри этого многоугольника или на границе.

Пример

См. рисунок 1.2

1.16. Как Вы думаете, почему в языках программирования названия переменных нельзя строить из любых символов?

1.17. Как Вы думаете, почему в языках программирования названия переменных, обычно, не начинаются с цифры?

1.18. Подумайте, каковы: файловая система, редактор, транслятор, библиотеки периода трансляции, библиотеки периода исполнения, отладчик, пользовательские библиотеки, средства поддержки разработки программ, — для известных Вам систем программирования.

1.19. Что такое модель вычислений?

1.20. Какие основные отличительные характеристики модели вычислений Фон Неймана?

1.21. Стыкуются ли современные интерфейсы пользователя и модель Фон Неймана?

1.22. Какие ещё модели вычислений Вы знаете?

1.23. Почему существуют несколько разных моделей вычислений?

1.24. Прочитать строку. Напечатать её три раза в одной строке.

1.25. Прочитать три строки. Напечатать их в строку через пробел.

1.26. Написать программу, которая печатает дату дня рождения авто-

ра (Ваш).

Пример

```
Мой день рождения 30 февраля.
```

1.27. Следующее число. Сделать программу, которая читает целое число и печатает следующее целое.

Пример

```
Число: 15
Следующее число: 16
```

1.28. Бифштекс. Для приготовления одного бифштекса нужно 100 грамм мяса. Вопрос: сколько нужно мяса для приготовления N бифштексов?

Пример

```
Сколько хотите бифштекстов - 3
Нужно мяса (в граммах) - 300
```

1.29. Мороженное. Для приготовления одного мороженного нужно 3 литра молока. Вопрос: сколько полных порций мороженного получится, если у нас есть N литров молока?

Пример

```
У нас молока - 11
Штук мороженного - 3
```

1.30. Имеет ли смысл добавлять в программе несколько пробелов там, где достаточно одного?

1.31. Имеет ли смысл делать в программе несколько переходов на новую строку там, где достаточно одного?

1.32. Какие символы или группы символов Вам приходилось встречать для обозначения комментариев?

1.33. Могут ли быть комментарии внутри комментариев?

1.34. В чём разница подпрограммы и библиотеки?

1.35. Какие способы подключения библиотек Вы знаете?

1.36. На каком этапе подключаемая библиотека переводится в машинный код?

1.37. Пусть есть две подпрограммы выполняющие разные действия, но

названные одинаково. И обе они Вам нужны. Подпрограммы встроены в оттранслированные модули и исправить названия нет возможности. Есть ли возможность использовать их обе?

1.38. Какова максимальная длина имени переменной в Вашем любимом языке программирования?

1.39. Какие символы можно использовать для имени переменной в Вашем любимом языке программирования?

1.40. Приведите примеры языков программирования, в которых можно описывать переменные и подпрограммы с одинаковым именем.

1.41. Приведите примеры языков программирования, в которых можно описывать переменные с одинаковым именем, несколько раз.

1.42. Приведите примеры языков программирования, в которых можно не описывать переменные.

1.43. Что такое тип?

1.44. В некоторых языках программирования можно не указывать типы переменных. Действительно ли переменным можно присваивать значения различных типов?

1.45. Для чего нужны типы?

1.46. Чем отличаются прив**и**дение и прив**е**дение?

1.47. Пусть есть программа, которая читает два числа и печатает их сумму. Но эта же программа для некоторых чисел печатает их произведение. Например, для чисел 2 и 2 или 11 и 11/10. Сколько ещё таких пар чисел?

1.48. Приведите пример действия, которое не меняет значения переменных.

1.49. Приведите пример действия, которое вообще ничего не меняет.

1.50. Если x=1 и y=2, то какие пары значений могут получиться при параллельном вычислении x=y, y=x. Перечислите все варианты, их три.

1.51. В чём различие понятий *параллельное вычисление* и *совместное вычисление*?

1.52. Как организовать обмен двух переменных своими значениями так, чтобы результат был одним и тем же, как для последовательного, так и для параллельного вычисления всех действий?

1.53. Заданы три числа, выдать самое большое из них.

1.54. Разные цифры. Сколько разных цифр содержится в заданном четырёхзначном числе.

Пример

```
Число: 0121
Ответ: 3
```

1.55. Напишите программу, которая для заданного римского числа печатает следующее. Входные римские числа в диапазоне от I до M.

Глава 2

Синтаксис, семантика и прагматика языка программирования

2.1. Самая короткая программа. Для известных вам языков программирования напишите программы наименьшей длины.

2.2. Приведите пример программы, содержащей
а) синтаксическую ошибку;
б) семантическую ошибку, обнаруживаемую компилятором;
в) семантическую ошибку, не обнаруживаемую компилятором;
г) алгоритмическую ошибку.

2.3. Как по-Вашему, что значит следующая команда
sphere{ <1,2,3>, 2 pigment{color White}}
на языке Povray?

2.4. Что описывают данные правила
<значок>::= "@"
<ряд>::= <значок><значок>|<ряд><ряд>
?

2.5. Сформулируйте с помощью средств естественного языка следующие контекстные зависимости для какого-либо языка или системы программирования:
а) каждое имя в программе должно быть определено;
б) каждое имя в программе должно быть определено ровно один раз;
в) каждое имя в блоке (подпрограмме, сложном операторе) должно быть определено один раз;

г) допускается переопределение функции в программе с другим набором параметров;
д) допускается использование имени в программе только после его определения;
е) не допускается обращение к элементу массива по индексу, не принадлежащему диапазону допустимых значений этого индекса, если его значение может быть вычислено на этапе компиляции программы;
ж) допускается присваивать строковым переменным значения символьного типа;
з) допускается присваивать вещественным переменным значение целого типа;
и) при вызове подпрограммы список фактических параметров должен соответствовать по количеству и типам формальным параметрам;
к) среди определенных функций программы должна присутствовать функция с именем main.

2.6. Можно ли считать операторы read и write языка Паскаль «встроенными процедурами»? Почему?

2.7. Являются ли семантически эквивалентными следующие преобразования кода программ языков C/C++, Pascal (буквы A, B, C, D означают некоторые выражения, буквы x, y — переменные)? Если ответ отрицательный, то сформулируйте ограничения, при которых указанные преобразования действительно эквивалентны.

а) if (A) {B; C;} else {B; D;}
преобразовано в
B; if (A) C; else D;
б) if (A) {B; C;} else {D; C;}
преобразовано в
if (A) B; else D;
C;
в) if A then B else C;
преобразовано в
if not A then C else B;
г) A*B+A*C
преобразовано в
A*(B+C)
д) A+(B+C)
преобразовано в
A+B+C
е) x=A+B;
y=A+B;

преобразовано в
x=A+B;
y=x;

ж) x=A*A;
преобразовано в
x=sqr(A);

з) A and (B and C)
преобразовано в
A and B and C

и) for i:=1 to y do
begin x:=1; A end;
преобразовано в
x:=1;
for i:=1 to y do A;

к) while A do B;
преобразовано в
if A then repeat B until not A;

л) repeat B until A;
преобразовано в
B; while not A do B;

м) if (A) if (B) C; else D;
преобразовано в
if (A && B)C; else D;

2.8. Язык с многочленами. Допустим, в некотором языке имеется возможность записи конструкции «многочлен». Синтаксис записи многочлена:

```
<многочлен> ::= <одночлен>| <одночлен><знак><многочлен>
<знак>::=+|-
<одночлен>::=<вещест-число>|<вещест-число> x^ <целое-число>
```

Сформулируйте обычные семантические ограничения, накладываемые на «правильную» запись многочлена, которые нельзя отразить с помощью синтаксических правил (например, что в записи многочлена не должно быть одночленов с одинаковыми показателями степени, с нулевыми коэффициентами и др.). Какой недостаток имеет приведенная здесь грамматика?

2.9. Почему следующая программа вычисляет не то, что нужно:

```
#define sqr(x) x*x
...
a=1;        //1
```

```
b=sqr(a+1); //2*2=3 ?!!!
```

2.10. Напишите несколько программ, где по смыслу была бы нужна конструкция **if**. Но используйте только управляющие конструкции **while**. Например, напишите программу к задаче 1.53.

2.11. Укажите для какого-нибудь языка программирования средства, предназначенные для оптимизации создаваемого кода или учитывающие вычислительную среду, но, с точки зрения концептуальной целостности, являющиеся избыточными. Попробуйте сравнить по этому признаку два любых языка.

2.12. Какое влияние оказывает прагматика вычисления логических выражений на процесс вызова функции f:
int n=1;
int f(){n++; return (n%2) && f();}

2.13. Выделите прагматику при работе с графикой в известных Вам языках программирования.

2.14. Постройте синтаксическое дерево вывода для каждой из следующих программ (или фрагментов программ):
а) void main(){return;}
б) void main(void){int i,j; i=j=1; return;}
в) int max(int a,b){return a>b?a:b;} void main(void){int x=max(1,2);}

2.15. Допустим, в некоторой системе программирования используется исключительно сокращенное вычисление логических выражений, то есть, когда при последовательном вычислении операндов конъюнкции $A_1 \wedge A_2 \wedge \ldots \wedge A_n$ встречается первое ложное значение, то оставшиеся операнды не вычисляются и результат конъюнкции объявляется ложным; симметрично — для дизъюнкции. Придумайте схему алгоритма, которая принуждает вычислять все операнды логического выражения.

2.16. Допустим, в некоторой системе программирования используется исключительно полное вычисление логических выражений. Придумайте схему алгоритма, при которой вычисляется минимальное количество первых операндов логического выражения, достаточное для вычисления всего результата.

2.17. Напишите программу, которая бы определяла, полное или со-

кращенное вычисление логических выражений применяется в системе программирования. Постарайтесь сделать её мобильной, то есть компилируемой в разных системах программирования с одним языком, но различной прагматикой.

2.18. Попытайтесь написать программу, которая бы определяла, контролируется или нет в системе программирования выход индекса массива за границу допустимых значений. Не пользуйтесь в программе прагматическими директивами!

2.19. Написать фрагмент программы, которая вводит два целых числа и выводит их сумму (разность), либо сообщает, что при выполнении операции сложения (вычитания) происходит переполнение разрядной сетки.

2.20. Решить задачу 2.19 для операции умножения двух целых чисел.

2.21. Написать программу, определяющую наименьшее положительное вещественное число (машинный эпсилон), представляемое в данном вещественном типе[1]. Проанализировать полученные результаты для разных вещественных типов языка программирования.

2.22. Напишите программу, определяющую максимально возможное значение в данном языке (системе) программирования
а) для любого из целых типов;
б) для любого из вещественных типов.

2.23. Во входном текстовом файле задано синтаксически правильное выражение; синтаксис выражения следующий:
```
<выражение> ::= <цифра>|(<цифра><знак><выражение>)
<знак> ::= +|-|*
<цифра> ::= 0|1|2|3|4|5|6|7|8|9
```
Построить по заданному выражению бинарное дерево выражения, вывести его на экран в виде дерева, найти значение выражения.

2.24. Во входном текстовом файле задано синтаксически правильное логическое выражение; синтаксис выражения следующий:
```
<лвыраж> ::= <лконстанта>|(<лконстанта><операция><лвыраж>)
<операция> ::= &|v|x
<лконстанта> ::= T|F
```
Построить по заданному логическому выражению бинарное дерево выражения, вывести его на экран в виде дерева, найти значение выражения.

[1]Точнее, машинный эпсилон — это наименьшее положительное вещественное число, которое при прибавлении к единице дает результат, отличный от единицы.

2.25. Во входном текстовом файле задан текст. Проверить, является ли текст правильной записью выражения согласно синтаксиса из задачи 2.23, и, если является, построить по нему дерево выражения, вывести дерево на экран, найти значение выражения.

2.26. Во входном текстовом файле задан текст. Проверить, является ли текст правильной записью логического выражения согласно синтаксиса из задачи 2.24, и, если является, построить по нему дерево выражения, вывести дерево на экран, найти значение выражения.

2.27. Во входном текстовом файле задан текст. Проверить, является ли текст записью правильного выражения согласно синтаксиса:

```
<выражение> ::= <терм>|<терм><знакпм><выражение>
<терм> ::= <целоечисло>|<целоечисло>*<терм>
<знакпм> ::= +|-
<целоечисло> ::= <цифра>|<цифра><целоечисло>
<цифра> ::= 0|1|2|3|4|5|6|7|8|9
```

и, если является, построить дерево выражения, вывести его на экран, вычислить значение выражения.

2.28. Высота дроби. Выражение записывается в соответствии с синтаксисом:

```
<выражение>::=<терм>|<терм>+<выражение>|<терм>-<выражение>
<терм>::=<простое>|<простое>*<терм>|<простое>/<терм>
<простое>::=<целое>|<имя>|(<выражение>)
<целое>::=<цифра>|<цифра><целое>
<цифра> ::= 0|1|2|3|4|5|6|7|8|9
<имя>::=<буква>|<буква><имя>
<буква> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
```

Выражение записывается в строку. Однако, если его записать по общепринятым правилам математики, то можно получить формулу с дробями, в том числе «многоэтажными». Например, выражение

```
(a+b)/(1/x)-(1/y)/(a+b)
```

может быть записано так:

$$\frac{a+b}{\frac{1}{x}} - \frac{\frac{1}{y}}{a+b}$$

Считаем, что в подобной записи высота букв в именах и цифр в числах равна l_1 пунктов, высота дробной черты равна l_2 пунктов. Вертикального расстояния между числителем и дробной чертой, а также между чертой

и знаменателем нет.

Даны натуральные l_1, l_2, правильная запись выражения. Найти высоту формулы (в пунктах), представляющей данное выражение.

2.29. Дополним язык, заданный в задаче 2.27, возможностью определения функций:

```
<ввод>::=<выражение>|<опр-функции>
<опр-функции> ::= def <имя-функ>(<форм-параметры>)=<выражение>
<форм-параметры> ::= <имя>|<имя>,<форм-параметры>
<выражение> ::= ...|<имя-функ>(<факт-параметры>)
<факт-параметры> ::= <выражение>|<выражение>,<факт-параметры>
```

Какими ещё правилами необходимо дополнить синтаксис? Опишите семантику данных синтаксических конструкций, сформулируйте семантические ограничения. Требуется написать программу, которая по входному тексту строит дерево выражения, вычисляет и выводит на экран значение выражения.

Пример

```
Введите: def sqr(x)=x*x
Введите: 1+sqr(2+sqr(2))
Значение=37
```

2.30. Лишние скобки. Дана запись арифметического выражения, в которое могут входить числа, имена переменных, знаки операций +, -, *, /, круглые скобки. Запись корректна, соответствует обычным правилам синтаксиса выражений со скобками. Требуется удалить из записи выражения все лишние пары скобок, не влияющие на результат вычисления.

Пример

```
Вход=(12+(4*(x1/2.0)))/(20-x)+(41)
Выход=(12+4*x1/2.0)/(20-x)+41
```

2.31. Язык с дробями. На базе языка целых чисел создадим язык выражений с рациональными числами:

```
<выражение>::=<терм>|<терм>+<выражение>|<терм>-<выражение>
<терм>::=<число>|<число>*<терм>|<число>%<терм>
<число>::=<целое>|<целое>/<целое>|<целое> <целое>/<целое>|
(<выражение>)
```

(Здесь знак "%"означает операцию деления, а знак "/"разделяет числитель и знаменатель обыкновенной дроби).

Сформулируйте

а) абстрактно-синтаксические диаграммы для указанных синтаксических конструкций;

б) семантические ограничения, налагаемые на цепочки языка (то есть

– на выражения с рациональными числами);

в) семантику выражений (описание смысла, приписываемого цепочке синтаксических конструкций, то есть способа вычисления выражения).

Создайте вычислитель значения выражения, построенного в соответствии с приведенным синтаксисом и семантикой.

2.32. Язык с множествами. Создадим язык описания и действий с множествами. Синтаксис:

```
<выражение>::=<терм>|<терм>+<выражение>|<терм>-<выражение>
<терм>::=<множество>|<множество>*<терм>
<множество>::=[<список-элементов>]|[]|(<выражение>)
<список-элементов>::=<элемент>|<элемент>,<список элементов>
<элемент>::=<целое-без-знака>|<диапазон>
<диапазон>::=<целое-без-знака>..<целое-без-знака>
<целое-без-знака>::=<цифра>|<цифра><целое-без-знака>
<цифра> ::= 0|1|2|3|4|5|6|7|8|9
```

Сформулируйте семантические ограничения и опишите семантику для данного синтаксиса. Напишите синтаксический анализатор выражений с множествами, вычислитель таких выражений.

Пример

Выражение: *([1..3,5,7]+[3,4,5]-[2])*[1..3,5..6,8..8]*

Ответ=[1,3,5]

2.33. Предложите абстрактно-синтаксическую структуру в виде дерева для следующих конкретно-синтаксических конструкций какого-либо языка (или языков) программирования

а) определение переменной;

б) определение константы;

в) определение массива;

г) определение записи (структуры);

д) определение класса;

е) цикл с предусловием;

ж) цикл с постусловием;

з) цикл с параметром;

и) оператор выбора.

2.34. Напишите наименьший по длине текста фрагмент программы на языке C/C++, эквивалентный данному:

а)
```
s:=0;
for i:=1 to 100 do
  if i mod 2 = 0 then begin
```

```
        s:=s+i;
        a[i]:=s;
      end;
```

б)
```
...
p,q:string;
...
q:='';
for i:=1 to length(p) do
  if p[i]<>' ' do
    q:=q+p[i];
```

2.35. Форматер программ. Написать программу, на вход которой подается текст программы на каком-нибудь языке программирования, на выходе — этот же текст, структурированный отступами слева с учетом вложенности операторов.

2.36. Написать программу, на вход которой подается текст программы на каком-нибудь языке программирования. Программа должна проанализировать текст и выдать предупреждение

а) о тех именах, которые не были определены в программе;
б) о тех переменных, которые определены, но не используются в программе;
в) о тех переменных, которые используются, но предварительно явно не были инициализированы.

2.37. Сформулируйте правила перехода от конкретно-синтаксического дерева программной конструкции (дерева вывода) к абстрактно-синтаксическому дереву для следующих конструкций какого-либо языка программирования:

а) ввод;
б) вывод;
в) присваивание;
г) последовательное выполнение операторов;
д) ветвление (условный оператор);
е) цикл;
ж) определение подпрограммы;
з) вызов подпрограммы.

Придумайте формализм для описания таких правил перехода.

2.38. Проэкспериментируйте, как влияет на временные характеристи-

ки программы явное и неявное преобразование типа. Например, в следующих трех операторах выполняемые действия эквивалентны (x – вещественная переменная):

```
x=1000;
x=1000.0;
x=double(1000);
```

Какова разница по времени исполнения этих операторов? Приведите другие примеры преобразований типа и проэкспериментируйте с ними.

2.39. Даны целые m_a, n_a, m_b, n_b ($m_a \leqslant n_a, m_b \leqslant n_b$). Допустим, типом переменной a является поддиапазон целых чисел $m_a..n_a$, а типом переменной b – поддиапазон $m_b..n_b$. Определить тип следующих выражений как минимально возможные поддиапазоны целого типа:

а) $a + b$;
б) $a - b$;
в) $a * b$;
г) $a\ div\ b$;
д) $a\ mod\ b$.

2.40. Нарисуйте блок-схему, которую нельзя перерисовать без пересечений соединительных линий.

2.41. Одинаковые программы. Можно сократить текст программы, если убрать символы, не влияющие на её работу (лишние пробелы, скобки и т.п.). Заметим, что добавлять ничего нельзя. Две программы назовём *одинаковыми*, если после удаления описанного выше получится один и тот же текст. Приведите пример функционально эквивалентных программ[2], не являющихся одинаковыми.

2.42. Подобные программы. Назовём программы *подобными*, если есть такой способ поменять имена переменных в этих программах, чтобы они стали одинаковыми (определение в задаче 2.41). Приведите пример функционально эквивалентных программ, но не являющихся подобными.

2.43. Напишите программу проверяющую одинаковость программ (см. задачу 2.41).

[2]Две программы называются *функционально эквивалентными*, если для любых одинаковых входных данных они: либо выдают одинаковый ответ, либо зацикливаются.

2.44. Напишите программу проверяющую подобие программ (см. задачу 2.42).

2.45. Напишите программу проверяющую функциональную эквивалентность программ (см. задачу 2.41).

2.46. Нарисуйте схему для арифметических выражений с помощью *абстрактно-синтаксического домино.*

2.47. Запишите программу вычисления максимального из трёх чисел на нескольких языках программирования (например, C++, РЕФАЛ, Prolog, Lisp).

2.48. Разработайте фрагмент грамматики для некоторого языка, в котором в условном операторе if-then-else

а) может существовать проблема «кочующего else», то есть возможна неоднозначность, к какому именно из нескольких if соответствует ветвь else; кстати, именно поэтому необходимо дополнять такие синтаксические правила дополнительными семантическими соглашениями;

б) проблема «кочующего else» решена синтаксически.

2.49. Цепное правило. Синтаксическое правило контекстно-независимой грамматики называется цепным, если в его правой части находится одиночный нетерминальный символ. Разработайте алгоритм, преобразующий грамматику к эквивалентной без цепных правил.

2.50. Почему следующие два фрагмента программ не являются эквивалентными? Первый фрагмент:
for i:=t1 to t2 do A;
второй фрагмент:
i:=t1;
while (i<=t2) do
begin A; i:=i+1 end;

2.51. Напишите программу, которая считает количество слов в заданной строке. Определите все грани семантики вашего языка, знание которых необходимо для решения данной задачи.

Глава 3

Стили программирования, или программирование с птичьего полета

3.1. Программист Иванов использует всегда названия идентификаторов из одной латинской буквы и двух цифр (например, A01, X99). Какие достоинства и какие недостатки у такого стиля программирования?

3.2. Почему скорость работы программы не является первоочередным критерием для программиста?

3.3. В естественном языке есть слова и выражения, именуемые жаргонами. Приведите примеры жаргонов в языках программирования.

3.4. Дайте определение стиля программирования.

3.5. Шестнадцатеричное число. Является ли строка шестнадцатеричным числом.

Пример

```
Вход: 1A2B4C6D8E0F0
Выход: ДА
```

3.6. $\mathbf{A} + \mathbf{X} = \mathbf{B}$. Решить уравнение вида: $A + X = B$ в целых числах.

Техническое требование

Уравнение задаётся в виде строки. A, B — целые числа. Между любыми символами строки возможны пробелы.

Пример

```
Уравнение: -1␣2␣3+␣X␣=␣-␣1␣␣2␣␣␣3␣␣␣␣4␣
X = -1111
```

3.7. Lex-Flex. Решите предыдущую задачу (3.6) с помощью языков lex или flex.

3.8. 10 целых AR. Записать программу, которая печатает 10 целых чисел не меньших A и делящихся на R нацело. A и R — запрашивает программа.

Пример

```
A=11
R=2
12 14 16 18 20 22 24 26 28 30
```

3.9. Напишите программу, решающую следующую задачу, используя только простые условия (без «и», «или» и других логических связок). Используйте только циклы и ветвления. Попробуйте обойтись без операторов перехода (условного и безусловного).

Задача. Задана строка. Проверить, что после каждой единицы всегда идёт ровно один ноль или строка заканчивается.

Например, 101010101010101 — верная запись, а 1010110, 102010, 01010 и 1010010 — нет.

3.10. Логический калькулятор. Запрограммируйте вычисление значения логического выражения классической логики высказываний. $\top$ — обозначает истину, $\bot$ — ложь.

Пример

Выражение: $\top \wedge (\top \Rightarrow \bot) \vee (\bot \Leftrightarrow \top)$

$\bot$

3.11. ✣ Сжатие строки. Один из простых способов сжатия информации заключается в поиске одинаковых частей, идущих подряд. Вместо нескольких повторений можно записать их количество и всего одну часть.

Задача.

По заданной строке выдать её максимальное сжатие. Длина получившейся строки считается вместе с числами, обозначающими количества повторений. Длина числа — количество цифр в десятичной записи.

Техническое требование

Задана строка из латинских больших и маленьких букв. Длина входной строки не более 90 символов. Если вариантов сжатия несколько, то выдать любой.

Пример

Строка: *abababababABabAbbccccccccccc*

Сжатие: 4ab1ABabAbb11c

3.12. Решить числовой ребус

$$(ABCD + CDEF) * ABEF = X.$$

Одинаковые латинские буквы обозначают одинаковые десятичные цифры, разные — разные. X — натуральное число, вводится. Ответ выдавать в виде ребуса, где вместо букв подставлены цифры. Если решения нет, сообщите об этом.

Пример

```
X= 5890640
(1234+3456)*1256 = 5890640.
```

3.13. Деловые переговоры. Путь одной фирме требуется на деловые переговоры отправить сотрудника. Одинаково для этой роли подходят: Иванов, Петров, Васильев и Сидоров. Руководитель фирмы не стал лично их искать искать, а поручил это сделать пяти своим заместителям в следующей форме: «Встретите Иванова, Петрова, Васильева или Сидорова, пошлите его на эту встречу».

Какие проблемы могут возникнуть при таком подходе?

Как правильно организовать отправку на переговоры, чтобы избежать этих проблем?

3.14. Что можно сказать о программировании событий в визуальных интерфейсах пользователя с точки зрения стиля?

3.15. Приведите пример использования механизма событий, но не для взаимодействия с человеком.

3.16. Тетрис. Запрограммируйте известную игру тетрис.

3.17. Рассмотрим следующую программу (синтаксис см. в [6] §13.2):

```
a:  PRIO PA+1   IF ExtrData   THEN PA=PA+1;   F1;   AWAKE a;
b:  PRIO PA     IF Data       THEN PA=PA-1;   F2;   AWAKE b;
```

Что будет если событие Data будет случаться значительно чаще, чем ExtrData?

3.18. В чём принципиальная разница между программированием от событий и программированием от приоритетов?

3.19. Приведите пример, когда программирование от приоритетов является наилучшим методом.

3.20. Что вычисляет данная программа

F(x, y) <= **if** y = 0 then 0 else F(F(x, y), F(y, x)) **fi**
?

3.21. Каков теоретически минимальный набор операций для работы с функциями как с неделимым объектом?

3.22. Вычислить $A(1,3) + A(2,3) + \ldots + A(n,3)$, где A — функция Аккермана (см. задачу 1.12).

3.23. Какие системы, использующие объектно-ориентированное программирование (ООП), Вы знаете?

3.24. Программист Консерваторов говорит, что ООП ничего принципиально нового не даёт. Ведь данные есть и так. Методы — те же подпрограммы. Все это уже есть, нужно только умело пользоваться. Попробуйте что-нибудь возразить.

3.25. Назовите два–три недостатка ООП.

3.26. Путь у Вас есть модули на все случаи жизни. То есть для решения любой задачи есть подходящий модуль, но только один. Есть доказательство того, что каждый модуль работает верно. Больше ни чего об этих модулях неизвестно. Можно ли гарантировать построение любой программы из этих модулей?
Подсказка: подумайте об эффективности — память и время.

3.27. Приведите пример, когда есть необходимость использовать уже разработанные модули.

3.28. **Шашки.** Напишите спецификации для программ, которые должны делать только один очередной ход в русских шашках. Нельзя ограничивать средства программирования. То есть программа — чёрный ящик.

Напишите программу, которая устраивает турнир между этими программами.

3.29. Иногда при программировании применяют шаблоны со свободными местами. При подстановке в эти места определённого программного кода или результата действий получается готовая программа. Приведите примеры таких систем.

3.30. Какова роль использования шаблонов технологии SSI в Web-программировании?

3.31. Специализируйте программу для решения задачи 1.14 при $n = 0$.

Какие новые полезные свойства появились в специализированной программе?

3.32. Специализируйте программу для решения задачи 1.14 при $X = 1$. Как повысилась эффективность полученной программы?

3.33. В чём сила специализирующего программирования?

3.34. Есть программа поиска совпадающих элементов в линейном массиве. Как она специализируется, если массив окажется упорядоченным? А если в массиве содержаться только цифры (0, 1, ..., 9)?

3.35. Бегущий символ. Если на текстовый экран выводить много строк, то возникает эффект движения изображения вверх. Пользуясь этим эффектом заставьте непрерывную линию из символа '*' двигаться по экрану вниз и в стороны. Чтобы линия казалась непрерывной нужно чтобы позиция символа в следующей строке отличалась от позиции символа в предыдущей не более чем на 1. Обратите внимание, что символ может попытаться выйти за края экрана справа и слева, не допустите этого.

Количество строк (шагов бега) программа должна запрашивать с клавиатуры.

Пример

```
Длина дистанции: 12
   *
  *
   *
   *
    *
     *
      *
     *
      *
     *
    *
   *
```

3.36. Бегущий символ 2. Теперь решите задачу 3.35 для двух символов: '*' и '#'. Но теперь признаком окончания движения будет пересечение путей символов.

Пример

```
      *          #
```

```
*            #
 *          #
 *         #
  *       #
   *     #
    *   #
    *  #
     * #
    * #
   * #
   *#
```

3.37. Бегущий символ 3. Теперь решите задачу 3.36 для трёх символов: '*', '@', и '#'.

Пример

```
Длина дистанции: 12
   *          #    @
  *            #   @
   *          #     @
   *         #       @
    *       #       @
     *     #       @
      *   #       @
      *  #         @
       * #         @
      * #          @
      *#           @
```

3.38. Бегущие символы. Решите задачу 3.37. Но для любого количества символов от 2 до 20. Символы и их количества (длина строки) задаются строкой.

Пример

```
Символы: ABA*C
 A B   A  *   C
 A  B  A  *    C
 A B  A   *     C
 A  B A  *      C
 A B   A *     C
 A  B   A*    C
```

3.39. Вспомните как решались задачи 3.35, 3.36, 3.37, 3.38. Не показа-

лось ли Вам, что более общую задачу 3.38 решать легче чем 3.37?

Глава 4

Понятие жизненного цикла программного обеспечения и его модели

4.1. Цель жизни. Нарисуйте иерархию целей вашей жизни. Определите главную цель (это будет корень дерева иерархии). Затем сформулируйте, каких целей необходимо добиться для её достижения (это будет более низкий уровень), и так далее до самых простых и конкретных целей.

4.2. Найти любое значение X при котором выражение (A*X+B)*X+C примет значение 0. A, B, C, X — целые числа. A, B, C — вводятся, X — выводится. Если такого числа нет, то выдать "Ответа нет".

Пример

```
A= 1
B= 3
C= 2
Ответ: -2
```

4.3. Почему технологический процесс разработки программной системы называется "цикл"?

4.4. Найти все значения X при котором выражение (A*X+B)*X+C примет значение 0. A, B, C — вводятся, X — выводится. Напоминание: задача 4.2 уже должна быть решена.

Пример

```
A= 1
B= 3
C= 2
```

```
X={-1, -2}
```

4.5. Если процесс разработки системы — цикл, то каково условие выхода из него?

4.6. Приведите пример процесса разработки системы, где общепринятая модель подходит не точно.

4.7. На каком этапе разработки жизненного цикла программы появляется возможность создания системы тестов? В чем преимущество, когда на этапе тестировании программного кода участвуют (или даже его полностью проводят) люди, не программировавшие данный продукт (тестеры)?

4.8. На каких этапах жизненного цикла могут участвовать люди, не являющиеся специалистами в сфере информационных технологий? Когда имеет смысл привлекать людей, вообще очень далеких от компьютерных технологий?

4.9. Приведите пример системы, для моделирования разработки которой не подходит классическая итерационная модель.

4.10. Найти любое значение X при котором выражение (A*X+B)*X+C примет значение 0. A, B, C — действительные числа. A, B, C — вводятся, X — выводится. Обратите внимание, что решение есть всегда при A или B не равных 0.

Напоминание: задача 4.4 уже должна быть решена.

4.11. Приведите пример процесса разработки системы, где классическая итерационная модель подходит не точно.

4.12. Предусловие и постусловие блока. Предусловием и постусловием программного блока будем называть логические высказывания, описывающие состояние вычислительной среды соответственно до и после выполнения блока. Истинность предусловия гарантирует истинность постусловия (если программный блок реализован «правильно»). То есть, если до выполнения блока вычислительная среда готова и соответствует требованиям предусловия, то исполнение блока гарантированно приведёт к ожидаемому результату. Например, для блока

```
x:=1/y;
```

предусловие может быть таким: «значением y является вещественное число, отличное от нуля», а постусловие таким: «значением x являет-

ся вещественное число, обратное величине y с учетом погрешности машинной операции деления». Таким образом, совокупность предусловия и постусловия могут входить в спецификацию программной единицы (блока, подпрограммы, модуля, класса и др.). Составьте пред- и постусловия для подпрограмм, решающих следующие задачи:

а) решение квадратного уравнения $ax^2 + bx + c = 0$, где $a \neq 0$;
б) полное решение уравнения $ax^2 + bx + c = 0$;
в) бинарный поиск в линейном вещественном массиве;
г) любой другой задачи на программирование.

4.13. Сформулируйте пред- и постусловия (в самом общем виде) для следующих этапов жизненного цикла программного обеспечения:

а) определение требований (планирование очередной итерации);
б) анализ выполнимости требовний;
в) моделирование интерфейса пользователя;
г) построение иерархии понятий;
д) конструирование;
е) программирование;
ж) тестирование;
з) оценка результата.

4.14. В каскадной модели жизненного цикла есть возвраты на предыдущие уровни. Приведите примеры для каждого из возвратов.

4.15. Найти любое значение X при котором выражение (A*X+B)*X+C примет значение 0. A, B, C — комплексные числа. A, B, C — вводятся, X — выводится. Обратите внимание на ввод.
Напоминание: задача 4.10 уже должна быть решена.

4.16. Приведите пример процесса разработки системы, где каскадная модель подходит не точно.

4.17. Визуализация решения задачи. Даны вещественные a_1, a_2, b_1, b_2, c_1, c_2, x, y. Определить, принадлежит ли точка с координатами (x, y) треугольнику, вершинами которого являются точки (a_1, a_2), (b_1, b_2) и (c_1, c_2) (точка считается принадлежащей, если она лежит на границе или во внутренности треугольника). Решение представить в графическом виде.
Указание: подобрать масштаб единиц декартовой системы координат и экранные координаты её центра таким образом, чтобы треугольник и точка имели бы на экране «оптимальное» положение и размеры. Определите, по каким стадиям развивается решение задачи?

4.18. Еще визуализация. Представить решение визуально (графически), как в задаче 4.17, для

а) задачи 7.16;
б) задачи 7.39;
в) задачи 11.16.

4.19. Гистограмма. Даны два ряда данных: $x_1, x_2, \ldots, x_n$ числового или строкового типа и $y_1, y_2, \ldots, y_n$ числового типа. Вывести столбчатую диаграмму (гистограмму), показывающую зависимость y_i от x_i. Подобрать разумный масштаб графического вывода и обеспечить наличие обычных элементов диаграммы (оси, подписи данных, легенда и другие.)

4.20. В модели фазы-функции есть этап «анализ осуществимости». Привести примеры, когда результат этапа отрицательный. Примеров должно быть не менее трёх и причины «неосуществимости» в них должны быть различны.

4.21. Приведите пример процесса разработки системы, где модель фазы-функции подходит не точно.

4.22. Планирование итераций разработки. Пусть требуется создать простейший графический редактор, позволяющий создавать цветные точечные изображения, сохранять и загружать картинки, выполнять графические преобразования и др. Спланируйте, на какие итерации можно разбить разработку? На каких этапах каждой итерации появляется возможность начала следующей итерации? Как наращивается функциональность программы от итерации к итерации?

4.23. В традиционных олимпиадах по программированию, когда за ограниченное время нужно решить как можно больше задач, учитываются только результаты тестов. Чем больше тестов прошли программы, тем больше баллов получает участник. Как здесь может помочь итеративный подход?

4.24. Решите задачу 4.22 для простейшего табличного процессора (электронной таблицы).

4.25. Программисты системы для решения поставленной задачи предлагают сначала написать код, а затем задокументировать, аргументируя это тем, что сразу получим реализуемое описание и не нужно будет думать о возможности или невозможности того, что спроектировано. Чем плох такой подход?

4.26. Если есть чёткие спецификации и каждый модуль написан и протестирован на соответствие своим спецификациям, то для чего нужно комплексное тестирование?

4.27. Допустим, требуется создать программу из задачи 4.22 (или задачи 4.37, 4.24) коллективом разработчиков (не менее 2 человек). Распределите части программного проекта таким образом, чтобы каждый разработчик имел возможность развивать свою часть проекта независимо от других частей максимально возможное время.

4.28. При разработке программ небольшой сложности бытует такой подход: сначала программируется пользовательский интерфейс, а затем основная логика программы. Какие преимущества и какие недостатки есть у данного подхода?

4.29. Инвариант цикла. Инвариант цикла — это утверждение (высказывание), которое истинно до выполнения цикла, после выполнения цикла, а, также, после каждой итерации цикла. Например, для цикла

```
s=0; p=1; i=0;
while (i<=4)
{i++; p=p*2; s=s+p;
}
```

инвариант может быть таким: $(p = 2^i)$ и $(s = \sum_{k=1}^{i} 2^k)$. В инвариант цикла могут входить высказывания, выраженные как на математическом языке, так и на естественном (например, русском). Постройте инварианты циклов для решений задач 7.37, 7.17, 7.33.

4.30. Допустим, требуется смоделировать (спланировать) развитие некоторого программного проекта в итеративной модели. Сформулируйте инвариант такого циклического процесса. В качестве примера возьмите какой-то конкретный проект или программу.

4.31. Сформулируйте инвариант спирального развития какого-либо объектно-ориентированного проекта.

4.32. В чем особенность инварианта параллельного или совместного циклов? Какие части тела следующего цикла можно реализовать параллельно (совместно) и каков в таком случае будет инвариант цикла:

```
s=0; p=1;
```

```
for(int i=1;i<=5; i++)
  {s=s+i; p=p*i;}?
```

4.33. Задача: поиск корней квадратного уравнения (см. 4.15). Что можно делать параллельно с реализацией интерфейса, а что нельзя?

4.34. Дорисуйте спиральную модель для случая, когда необходимо поддерживать две версии системы, например, для разных платформ.

4.35. Приведите пример момента в разработке системы, когда не помогает даже итеративное наращивание возможностей системы.

4.36. Функциональность. Постройте иерархию функциональности для какой-либо программной системы (например, текстового редактора, браузера, системы программирования и др.). В корне иерархии помещается описание самых общих функциональных возможностей, ниже — более конкретно и детально определяются вышестоящие возможности и так далее.

4.37. Сформулируйте требования и спецификации (функциональность) для разработки следующих программных систем:

а) простейший текстовый редактор;
б) простейший графический редактор;
в) простейший файловый менеджер;
г) простейший браузер;
д) архиватор;
е) калькулятор;
ж) простейший почтовый клиент;
з) простейшая система программирования.

Слово «простейший» означает минимально необходимый уровень функциональности для решения прикладных или системных задач.

4.38. Сформулируйте требования и спецификации данных и операций программного модуля, реализующего структуру стека. Предложите несколько различных вариантов реализации стека при одних и тех же спецификациях.

4.39. Решите задачу 4.38 для следующих информационных структур:

а) очередь;
б) бинарное дерево;
в) бинарное дерево поиска;
г) бинарное дерево выражения;

4.40. Постройте иерархическую абстракцию понятий для структуры данных «список», (использую, например, такие понятия, как стек, очередь, дек и другие). Опишите спецификации процедур обработки указанных структур. Как наследуются спецификации свойств структур и процедур их обработки?

4.41. Решите задачу 4.40 для информационной структуры «дерево».

4.42. Какие проблемы управления требованиями возможны при реализации простого текстового редактора? Перечислите по две на каждый из следующих пунктов:

- много источников;
- неочевидность требований;
- невыразимость словами некоторых требований;
- множественность типов;
- множественность уровней детализации;
- взаимосвязанность;
- взаимозависимость;
- противоречивость;
- уникальность;
- компромисс в требованиях;
- изменение требований;
- зависимость от времени.

4.43. Последовательность требований. Ниже приведена последовательность требований. Попробуйте выполнить трассировки для них. Считайте, что эти требования поступают по одному и в указанной ниже последовательности.

1. Написать программу для решения уравнения $a + x = b$.
2. a и b — натуральные числа.
3. x — число целое.
4. a и b — действительные числа.
5. Если решения найти нельзя, вывести строку: «В данном случае решений нет и быть не может».
6. Если есть несколько решений, выдать все.
7. a — строка символов.
8. x — действительное число.
9. a, b и x — строки символов.

А теперь классифицируйте эти требования.

4.44. Составьте глоссарий проекта для разработки простого текстового редактора. Он должен содержать не менее 20 понятий.

4.45. Постройте последовательность программ для последовательности требований задачи 4.43. Проанализируйте последовательность шагов жизненного цикла в этом случае.

4.46. Система на выброс Как соотносится первая итерация и понятие «система на выброс» описанное у Ф. П. Брукса [3]?

4.47. Решить задачу 4.45, но теперь при построении последовательности постарайтесь писать программы так, чтобы минимизировать затраты при переходе к новым требованиям.

4.48. Опишите модель жизненного цикла для решения задач по программированию «очень средним» студентом вуза. Так называемого «решения на сдано».

4.49. Переход к новому проекту. Опишите фазу завершения при переходе к новому проекту.

4.50. Вспомните Ваши действия при написании программ для задач 4.2, 4.4, 4.10, 4.15. Какие этапы жизненного цикла Вы проходили при написании этих программ? А в каких моделях они присутствуют?

4.51. Вспомните 2-3 системы, реализующие поддержку разных моделей жизненного цикла. Опишите подробно, какие модели и этапы поддерживаются в них.

4.52. CASE. Приведите примеры 3-4 современных CASE-систем.

4.53. Влияет ли модель жизненного цикла на стиль программирования?

4.54. Влияет ли стиль программирования на модель жизненного цикла?

4.55. В каком месте жизненного цикла выбирается стиль программирования?

4.56. Приведите пример процесса разработки, который бы не подходил ни под одну модель жизненного цикла описанного в [5, 11].

4.57. Приведите пример системы, при разработке которой, без ущерба качеству и эффективности разработки, можно пользоваться любой стандартной моделью.

4.58. Способы тестирования. Приведите примеры нескольких различных способов тестирования.

4.59. Структурное тестирование. Приведите пример структурного тестирования.

4.60. Сборочное тестирование. Приведите пример сборочного тестирования.

4.61. Функциональное тестирование. Приведите пример функционального тестирования.

4.62. Регрессионное тестирование. Приведите пример регрессионного тестирования.

4.63. Нагрузочное тестирование. Приведите пример нагрузочного тестирования.

4.64. Стрессовое тестирование. Приведите пример стрессового тестирования.

4.65. Качество системы. Что такое качество программной системы?

Глава 5

Выражения

5.1. Определите, какие именно неявные приведения типов при присваивании значения применяются в вашей системе программирования.

5.2. Здравствуй. Написать программу, которая будет здороваться со своим автором. Например, если программу составит Пупышев Вячеслав Викторович, то программа будет выдавать:
`Здравствуй, Вячеслав!`

5.3. Поглядывающий. Написать программу, которая печатает:
`--@,@--`.

5.4. Пингвин. Написать программу, которая рисует:

```
   _~_
  (o o)
 // V \\
/(  _  )\
  ^^ ^^
```

5.5. Даны значения двух переменных a и b. Поменять местами значения этих переменных.

5.6. Даны значения трех переменных a, b и c. Поменять местами значения этих переменных так, чтобы a получило бы значение b, b – значение c, а c – исходное значение a.

5.7. Арифметика. Написать программу, которая печатает результаты выражений:

а) $5 + 9$; б) $1987 + 3215$;
в) $12345 - 91$; г) $37 - 19998$;
д) $53 \cdot 81$; е) $-554 \cdot 7$;
ж) $-127 \cdot (-156)$; з) $(23 \cdot 2 - 30 \cdot 3) + 987$.

5.8. Инициалы. Написать программу, которая запрашивает фамилию, имя и отчество, а выводит фамилию и инициалы.

Пример

```
Фамилия: Пупышев
Имя: Вячеслав
Отчество: Викторович
Пупышев В.В.
```

5.9. Всем поровну. У N школьников есть мешок карамели «Чупачупс». По сколько, максимум, конфет можно раздать школьникам, чтобы оставшихся в мешке всем досталось поровну?

Техническое требование

Будут заданы натуральные числа: количество школьников и количество карамелек.

Пример

```
Школьников:  10
Конфет:  43
Каждому по: 4
```

5.10. А ну-ка, подели! У N школьников есть мешок карамели «Чупачупс». Сколько, минимум, конфет нужно убрать из мешка, чтобы всем школьникам досталось поровну?

Техническое требование

Будут заданы натуральные числа: количество школьников и количество карамелек.

Пример

```
Школьников: 10
Конфет: 43
Нужно убрать: 3
```

5.11. Дано число a. Найти:
а) a^4 за две операции умножения;
б) a^6 за три операции умножения;
в) a^5 за три операции умножения;
г) a^{10} за четыре операции умножения;

д) a^{21} за шесть операций умножения;
е) a^{28} за шесть операций умножения;
ж) a^{32} за пять операций умножения;
з) a^{40} за шесть операций умножения;
и) a^{48} за шесть операций умножения;
к) a^{36} за шесть операций умножения;
л) a^{64} за шесть операций умножения;
м) a^{104} за восемь операций умножения.

5.12. Дано вещественное x. Найти значение выражения $16x^4 - 6x^3 + 4x^2 - x + 1$. Разрешается использовать не более 5 операций умножения.

5.13. Дано вещественное x. Найти значение выражения $x^4 - 2x^3 + x^2 - 2x + 1$. Разрешается использовать не более 4 операций умножения.

5.14. Дано вещественное x. Найти значение выражения $8x^7 + 4x^4 + 2x^3 - 2x^2 + x - 1$. Разрешается использовать не более 5 операций умножения.

5.15. Дано время N в секундах. Выразить это время в сутках, часах, минутах и секундах.

5.16. Даны целые неотрицательные h_1, m_1, h_2, m_2 ($h_1, h_2 \in [0, 23]$; $m_1, m_2 \in [0, 59]$). Поезд вышел первого числа со станции отправления в h_1 часов m_1 минут и был в пути до станции назначения h_2 часов m_2 минут. Определить число и время прибытия поезда на станцию назначения.

5.17. Даны натуральные n и a. Построить выражение, принимающее значение номера дня недели n-го числа месяца, у которого первое число имеет номер дня недели a (понедельник — первый день недели, воскресенье — седьмой).

5.18. Пусть a — номер дня недели (от 1 до 7) 1 января невисокосного года. Построить выражение, принимающее значение номера дня недели, на который выпадает дата d числа m месяца этого года.

5.19. Построить выражение, принимающее значение номера дня недели, на который выпадает дата d числа m месяца y года (в пределах дат григорианского календаря).

5.20. Дано целое трехзначное число a. Найти число b, полученное из a перестановкой цифр в обратном порядке.

5.21. Решить задачу 5.5 для случая двух вещественных переменных.

Ограничение: не использовать дополнительных переменных, использовать только присваивание.

5.22. Решить задачу 5.5 для случая двух логических переменных.
Ограничение: не использовать дополнительных переменных, использовать только присваивание.

5.23. Решить задачу 5.5 для случая двух строковых переменных.
Ограничение: не использовать дополнительных переменных, использовать только присваивание.

5.24. Решить задачу 5.6 для случая трех целых переменных.
Ограничение: не использовать дополнительных переменных, использовать только присваивание.

5.25. Даны вещественные a и b. Найти наибольшее значение из a и b.
Ограничение: разрешается использовать *только* оператор присваивания.

5.26. Даны вещественные a и b. Найти наименьшее значение из a и b. Ограничение то же, что и в задаче 5.25.

5.27. Даны различные вещественные a, b и c. Найти то значение из них, которое лежит на числовой оси между двумя другими. Ограничение то же, что и в задаче 5.25.

5.28. Дано целое трехзначное число a. Найти наименьшее число b, полученное из a перестановкой цифр. Ограничение то же, что и в задаче 5.25.

5.29. Свечки для торта. Родители купили ребенку набор свечей для торта, количество свечей в наборе — N (натуральное число). Когда ребенку исполнился один год, на торте была одна свечка, два года — две, и т.д. На сколько лет хватит приобретенного набора свечей?

5.30. Множество натуральных чисел $\{1, 2, 3, \ldots\}$ разобъем на последовательные упорядоченные наборы: в первом наборе один элемент $\{1\}$, во втором — два $\{2, 3\}$, в третьем — три $\{4, 5, 6\}$ и т.д. Дано натуральное N, принадлежащее одному из наборов. Найти номер этого набора и номер, под каким элемент входит в этот набор.

5.31. Множество натуральных чисел разбито на наборы, как в задаче 5.30. Даны два натуральных числа $k, l (l \leqslant k)$. Какое значение имеет l-ый

элемент набора с номером k?

5.32. Частное любых чисел. Написать программу, которая запрашивает два целых числа и выводит их частное.
Внимание: при вводе 2 и 0 возникает аварийная ситуация!

Пример

```
Первое число: 2
Второе число: 0
На ноль делить НЕЛЬЗЯ !!!
```

5.33. Две разные цифры. Сколько разных цифр содержится в заданном двухзначном числе.

Пример

```
Число: 01
Ответ: 2
```

5.34. В строке поменять первый и последний символ местами, если они там есть.

Пример

```
Строка: abcd
dbca
```

5.35. Вывести результаты деления 100 на заданное число и на число на 1 меньшее заданного.

Пример

```
Число: 3
33.3333
50.0000
```

5.36. Дано целое неотрицательное N. Построить логическое выражение, принимающее значение «ИСТИНА», если N — високосный год, и «ЛОЖЬ» — в противном случае. В григорианском календаре високосным считается год, если он делится на 4, за исключением тех годов, которые делятся на 100, но не делятся на 400.

5.37. Даны целые x_1, y_1, x_2, y_2 ($x_1, y_1, x_2, y_2 \in [1;8]$, причем пары (x_1, y_1) и (x_2, y_2) различны). Поля шахматной доски задаются координатами (a, b). Присвоить логической переменной t значение «ИСТИНА», если ферзь, находящийся на поле (x_1, y_1), бьет фигуру, находящуюся на поле (x_2, y_2), и «ЛОЖЬ» — в противном случае. Ограничение то же, что и в задаче 5.25.

5.38. Даны целые x_1, y_1, x_2, y_2 ($x_1, y_1, x_2, y_2 \in [1;8]$, причем пары

(x_1, y_1) и (x_2, y_2) различны). На поле шахматной доски (x_1, y_1) находится слон, на поле (x_2, y_2) — конь. Присвоить логической переменной t значение «ИСТИНА», если одна фигура бьет другую, и «ЛОЖЬ» — в противном случае. Ограничение то же, что и в задаче 5.25.

5.39. Сформулировать на языке программирования логическое выражение, истинное при выполнении следующего условия, и ложное – в противном случае:

а) $a = max(x, y, z)$;
б) целые m и n имеют одинаковую четность;
в) целые p и q могут быть соответственно числителем и знаменателем правильной несократимой обыкновенной дроби;
г) уравнение $ax^2 + bx + c = 0$ имеет ровно один вещественный корень;
д) система из двух линейных уравнений имеет ровно одно решение;
е) переменные a и b имеют одинаковые логические значения;
ж) ровно две из трех переменных a, b и c имеют одинаковые логические значения;
з) из клетки (a_1, b_1) шахматной доски можно попасть в клетку (a_2, b_2) за один ход

- ферзя;
- коня.

5.40. Расставить скобки. Расставьте скобки в выражении, приведенном справа, так, чтобы последовательность выполняемых операций соответствовала записи математического выражения (слева):

а) $\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{3+\cfrac{1}{4}}}}$ $\quad 1/1 + 1/2 + 1/3 + 1/4$

б) $\dfrac{a+b}{x-y}c - y \quad a + b * c/x - y - y$

5.41. Определим приоритеты. Арифметическое выражение строится с использованием знаков четырех операций $+, -, *, /$ в соответствии с синтаксисом:

```
<выражение>::= <первичное>|<первичное><знак><выражение>
<первичное>::= <целаяконстанта>|(<выражение>)
<знак>::=+|-|*|/
```

Будем считать,что операция "/" означает деление двух чисел нацело. Скобки в этом выражении означают, что их содержимое должно быть

вычислено в первую очередь. Дано арифметическое выражение, целое значение n. Для каждой из четырех арифметических операций указать номер приоритета таким образом, чтобы значение вычисленного выражения было бы равно n. (Номера приоритетов – натуральные числа, не превосходящие 4, номера приоритетов двух или более операций могут совпадать. Выше тот приоритет операции, номер которого больше. Если две последовательные операции в выражении имеют одинаковый приоритет, то они вычисляются слева направо.)

Пример.

*3-2*3+(12/5), n=5*

`+4, -3, *2, /1`

5.42. Б **Гиперсумма.** Написать программу для вычисления остатка от деления

$$(k+1)!\sum_{i_1=0}^{n}\sum_{i_2=0}^{i_1}\cdots\sum_{i_k=0}^{i_{k-1}} i_k$$

на число m, где n, k, m — натуральные числа от 1 до 32000. Числа n, k, m вводятся в указанном порядке с клавиатуры. Результат выводится на экран. Время счета — не более 10 секунд.

Пример

При вводе 1, 1, 3 программа должна выдать 2.

Глава 6

Разветвление вычислений

6.1. Даны целые числа N, D, R. Проверить, делится ли число N на D с остатком R?

6.2. Даны две символьные строки A и B. Проверить, верно ли, что строка A короче строки B в 2 раза?

6.3. Дано целое число A. Верно ли, что число A делится на 7?

6.4. Даны два числа X, Y. Проверить, верно ли, что число X больше числа Y?

6.5. Даны две символьные строки A1, A2. Проверить, верно ли, что строки A1 и A2 имеют одинаковые длины?

6.6. Даны две символьные строки A, B. Проверить, верно ли, что строки A и B имеют разные длины?

6.7. Даны две символьные строки A, B. Проверить, верно ли, что строка A длиннее строки B?

6.8. Даны два целых числа S и T. Верно ли, числа S и T имеют разные чётности?

6.9. Даны три числа A, B и Max. Проверить, верно ли, что Max равно максимальному из чисел A и B?

6.10. Больше фруктов. Одно яблоко весит 100 граммов, а один апельсин — 150 граммов. Нам дают несколько яблок либо несколько апельсинов. Хочется взять то, что весит побольше. Скажите, что взять: яблоки или апельсины?

Пример

```
Яблок: 7
Апельсинов: 5
Бери апельсины!
```

6.11. Копирование игры. Вам предлагают переписать новую игру. Из носителей информации у Вас есть: компакт-диск объёмом 650Мб, дискета объёмом 1Мб и лента объёмом 1000Мб. Перечислить все носители на которые игра поместится полностью. Размер игры задаётся в мегабайтах.

Пример

```
Размер игры (в Мб): 3
Войдёт на:
компакт-диск
ленту
```

6.12. Счастливое четырёхзначное. Четырехзначное число называется счастливым, если сумма первых двух его цифр равна сумме двух последних. Ваша программа должна читать четырехзначное число и определять, счастливое ли оно.

Пример

```
Число: 9889
Счастливое.
```

6.13. Мегабайты. Перевести килобайты в полные мегабайты и оставшиеся килобайты.

Пример

```
Килобайты: 3000
2Мб 952Кб
```

6.14. Ещё больше фруктов. Одно яблоко весит 100 граммов, один апельсин — 150 граммов, а банан — 170 граммов. Нам дают несколько яблок или апельсинов или бананов, взять можно только одно. Хочется взять то, что весит побольше. Скажите, что взять: яблоки, апельсины или бананы?

Пример

```
Яблок: 7
Апельсинов: 5
Бананов: 4
Бери апельсины!
```

6.15. Даны значения трех переменных a, b и c, отличные друг от друга. Обменять значения переменных с наибольшим и с наименьшим значениями.

6.16. Даны целые неотрицательные a, b, c, X, Y, Z. Известно, что в каждый новогодний подарок необходимо положить a конфет, b яблок и c груш. Какое максимальное количество подарков можно скомлектовать из X конфет, Y яблок и Z груш.

6.17. Запись игры. Вам предлагают переписать новую игру размером 2000 килобайт. Из носителей информации у Вас есть три дискеты со свободным местом A, B и C килобайт. Стирать и переносить информацию с дискет запрещено. Сколько дискет нужно, чтобы записать эту игру?

Пример

```
Свободное место на первой дискете: 1440
Свободное место на второй дискете: 570
Свободное место на третьей дискете: 1400
Дискет достаточно: 2
```

6.18. Больше объёма. Девочке записывали программу на компакт-диск. Если игра не входит, то девочка получает ещё один носитель для сохранения игры. Носителей два вида: мини-диск и компакт-диск. Мини-диск имеет размер 200Mb, компакт-диск — 700Мб. Всегда записывают на компакт-диски, а мини-диск используют только, если записать придётся не больше 200Мб. Какие носители получит девочка, если известен размер программы. Размер программы не больше 1400Мб.

Пример

```
Размер игры: 900
Девочка получит компакт-диск и мини-диск
```

6.19. ФИО. Написать программу, которая читает строку и определяет, что это: Ваше имя, фамилия или отчество. Если ни то, ни другое и не третье, то пишет: «НЕ ЗНАЮ». Заметим что, у каждого автора программы свои имя, фамилия и отчество и он должен включить их в программу.

Пример

(для Пупышева Вячеслава Викторовича)

```
Строка: Викторович
Викторович - Ваше отчество.
```

6.20. Название оценки. Сделать программу, которая читает число

от 1 до 5 и печатает название оценки.

Пример

```
Оценка: 1
Очень плохо.
```

6.21. Оценка ли? Решить задачу 6.20, но можно вводить любое целое число. Если число не является оценкой, печатать: «Не оценка».

Пример

```
Оценка: -3
Не оценка.
```

6.22. Цифра ли? Сделать программу, которая читает строку и сообщает «цифра», если строка – десятичная цифра. Если строка не является цифрой, печатать: «не цифра».

Пример

```
Оценка: -0
не цифра.
```

6.23. Линейное уравнение. Даны вещественные значения b и c. Решить уравнение $bx + c = 0$.

6.24. Квадратное уравнение. Даны значения a, b и c ($a \neq 0$). Решить квадратное уравнение $ax^2 + bx + c = 0$.

6.25. Система линейных уравнений. Даны вещественные a_1, a_2, b_1, b_2, c_1, c_2. Решить систему уравнений

$$\begin{cases} a_1 x + b_1 y &= c_1 \\ a_2 x + b_2 y &= c_2 \end{cases}$$

6.26. Даны числа x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4. Определить, лежит ли точка (x_1, y_1) внутри или вне треугольника, с вершинами (x_2, y_2), (x_3, y_3), (x_4, y_4).

6.27. Даны координаты точки плоскости (x, y). Определить область, которой принадлежит эта точка: номер четверти или имя оси или начало координат.

6.28. Даны вещественные положительные r, a и b. Определить, можно ли окружность радиуса r разместить внутри ромба с диагоналями a и b.

6.29. Даны вещественные положительные числа x, y, a, b, c. Определить, пройдёт ли кирпич с рёбрами a, b и c в прямоугольное отверстие в стене со сторонами x и y.

6.30. Даны координаты центров и радиусы двух окружностей, задаваемые уравнениями $(x-x_1)^2+(y-y_1)^2={r_1}^2$ и $(x-x_2)^2+(y-y_2)^2={r_2}^2$. Определить, лежит ли целиком одна окружность внутри другой.

6.31. Даны вещественные положительные a, b, c, d, e, f. Считаем, что пары чисел a и b, c и d, e и f обозначают размеры первого, второго и третьего прямоугольника соответственно. Выяснить, можно ли внутри одного из прямоугольников уместить два других, не накладывая один на другой.

6.32. Задача жестянщика. Даны вещественные положительные r, a, b, c и d. Определить, можно ли из круглой заготовки радиуса r вырезать две пластины прямоугольной формы с размерами $a \times b$ и $c \times d$.

6.33. Студент добирался домой в другой город: вначале на автобусе t_1 часов со скоростью v_1, затем на велосипеде t_2 часов со скоростью v_2, затем пешком t_3 часов со скоростью v_3. На половине пути он делал остановку в придорожном кафе. На чем он тогда передвигался?

6.34. Даны значения a, b и c ($a \neq 0$). Решить биквадратное уравнение $ax^4+bx^2+c=0$.

6.35. Даны значения a, b и c. Решить уравнение $ax^4+bx^2+c=0$.

6.36. Дано значение a. Решить уравнение $\cos^2 x-(a-2)\cos x+4a+1=0$.

6.37. Дано значение a. Решить уравнение $a^2\sin^2 x+(2a-1)\sin x+1=0$.

6.38. Даны вещественные числа a, b и c в десятичной непериодической записи. Найти точные вещественные корни уравнения $ax^2+bx+c=0$ (то есть если результат выполнения некоторой операции нельзя записать в виде десятичной непериодической дроби, то эту операцию выполнять не нужно).

Пример

```
a=1 b=-1 c=-1
x1=0.5- sqrt(5)/2
x2=0.5+sqrt(5)/2
```

6.39. Даны вещественные $x_1, y_1, r, x_2, y_2, x_3, y_3$. Определить, существуют ли общие точки у круга с центром в точке (x_1, y_1) и радиусом r и области, ограниченной прямоугольником, стороны которого параллельны или перпендикулярны осям координат, а (x_2, y_2) и (x_3, y_3) – координаты

его противоположных вершин.

6.40. Даны четыре точки на числовой прямой A, B, C и D. Найти длину пересечения отрезков AB и CD.

6.41. Даны вещественные $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$. Определить, имеет ли отрезок с концами (x_1, y_1) и (x_2, y_2) общие точки с отрезком с концами (x_3, y_3) и (x_4, y_4).

6.42. Даны вещественные $x_1, y_1, x_2, y_2, x_3, y_3$. Найти количество точек треугольника с вершинами (x_1, y_1), (x_2, y_2), (x_3, y_3), принадлежащих осям координат (возможен ответ «бесконечное множество»).

6.43. Даны координаты двух точек плоскости $A(x_1, y_1), B(x_2, y_2)$. Определить названия координатных осей, пересекаемых отрезком AB.

6.44. Если в числе есть хотя бы две значащие цифры, напечатать их по одной в строке.

6.45. Напечатать 3 первых символа введенной строки.

6.46. Сортировка четырёх. Написать программу, которая запрашивает четыре числа и выводит их в порядке возрастания.

Пример

```
3 8 19 11
3 8 11 19
```

6.47. Дешевая смесь. Пусть v_1, v_2 — объемы двух растворов, k_1, k_2 — их концентрации, p_1, p_2 — их цены за единицу объема. Пусть требуется получить смесь объемом v_3 с концентрацией k_3. Найти такие объемы смешиваемых расворов v_1' и v_2' ($v_3 = v_1' + v_2', v_1' \leqslant v_1, v_2' \leqslant v_2$), чтобы стоимость требуемой смеси была бы наименьшей, либо сообщить причину, почему это нельзя сделать.

6.48. Число прописью. Дано целое положительное число $N \leqslant 999$. Записать это число в виде числительного

а) русского языка;

б) английского языка.

6.49. О рублях. Дано целое неотрицательное число $N \leqslant 999$. Просклонять числительное, обозначающее денежную сумму в N рублей, по падежам.

6.50. Даны координаты четырех различных точек плоскости $A(x_1, y_1)$,

$B(x_2, y_2)$, $C(x_3, y_3)$ и $D(x_4, y_4)$, никакие три из которых не лежат на одной прямой. Определить, каким является четырехугольник $ABCD$: выпуклым, ромбом, параллелограммом, прямоугольником, квадратом, трапецией или ни одной из указанных фигур. Сформулируйте решение с помощью оператора выбора Дейкстры и таблицы решений (см. §6.2 [5] или §14.3 [6]).

6.51. Покер.Даны пять чисел. Если все они равны друг другу, то вывести число 1, если одинаковы ровно четыре числа, то вывести 2, если равны друг другу три числа и равны друг другу два других числа, то вывести 3, иначе если одинаковы три, то вывести 4, иначе если одинаковы два и два числа, то вывести 5, иначе если одинаковы только два числа, то вывести 6, во всех остальных случаях вывести 7.

6.52. Снова квадратное уравнение. Написать программу решения квадратного уравнения $ax^2 + bx + c = 0$ $(a \neq 0)$, используя наименьшее число арифметических операций и с учетом затрат времени на их выполнение: считать, что сложение/вычитание выполняется быстрее, чем умножение/деление, а извлечение корня выполняется медленнее всех. Квадрат числа вычисляется быстрее, чем умножение этого числа на само себя.

Глава 7

Циклические вычисления

7.1. Числа от 10 до 100. Написать программу, печатающую все целые числа от 10 до 100 включительно.

7.2. Прочитать N строк ($N < 100$) и напечатать их в обратном порядке, каждую с новой строки.

Пример

a
b
c
Результат:
c
b
a

7.3. Прочитать N строк ($N < 100$) и напечатать их через пробел.

Пример

aa
bb
cc
dd
Результат:
aa bb cc dd

7.4. Прочитать N строк ($N < 100$) и напечатать их через запятую. В конце должна быть точка.

Пример

aa bb cc
Результат:
aa, bb, cc.

7.5. Чётные от 2 до 100. Сделать программу, печатающую все чётные числа от 2 до 100 включительно.

7.6. Славься, автор. Записать программу, которая печатает имя автора программы столько раз, сколько указано.

Пример

```
Сколько? 3
Вячеслав
Вячеслав
Вячеслав
```

7.7. Цифры числа. Записать программу, которая печатает все цифры заданного числа по одной в строке начиная с последней. Число от 0 до 1 000 000 000

Пример

```
Число: 123457890
0
9
8
7
6
5
4
3
2
1
```

7.8. Напечатать числа $\underbrace{1\ 2\ 4\ 8\ 16\ \ldots}_{\text{N штук}}$

Пример

```
N=6
1 2 4 8 16 32
```

7.9. Напечатать N символов "*" подряд.

Пример

```
Количество звёздочек: 10
**********
```

7.10. Спрашивать свою фамилию до тех пор, пока не наберут правильно.

Пример

```
Фамилия: Иванов
Фамилия: Петров
```

```
Фамилия: пупышев
Фамилия: Пупышев
```

7.11. Напечатать

```
*
**
***
...
(20 рядов)
```

7.12. Наклонная. Для заданной высоты нарисовать наклонную линию.

Пример

```
Высота: 5
 *
 |*
 ||*
 |||*
 ||||*
```

7.13. Напечатать рисунок.

```
*\*
**\\***
***\\\*****
****\\\\********
...
```

Задаётся N — количество строк

7.14. Сколько, сколько? Дано натуральное $n (n \leqslant 9)$. Найти количество натуральных чисел, десятичная запись которых содержит n значащих цифр и все эти цифры различны.

7.15. Добавление запятых. Записать программу, которая печатает запятую и пробел после каждого символа заданной строки, а в конце ставит точку.

Пример

```
Строка: Заданная␣строка
З, а, д, а, н, н, а, я,  , с, т, р, о, к, а.
```

7.16. Дано множество n точек координатной плоскости (x_1, y_1), $(x_2, y_2), \ldots, (x_n, y_n)$. Найти координаты центра круга наименьшего радиуса, содержащего в себе все точки множества.

7.17. Схема Горнера. Дано вещественное x, целое $n, n > 0$. Найти значение многочлена n-ой степени $P(x)$ в точке x по схеме Горнера

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = (\cdots (a_n x + a_{n-1}) x + \cdots) x + a_0$$

Значения коэффициентов взять равными:

а) $a_i = 1$ $(i = 0, 1, 2, \ldots, n)$;
б) $a_i = i$ $(i = 0, 1, 2, \ldots, n)$;
в) $a_i = 2^i$; $i = 0, 1, 2, \ldots, n$;
г) $a_i = i!$ $i = 0, 1, 2, \ldots, n$;
д) коффициенты последовательно вводятся с клавиатуры (массивы не использовать!).

7.18. Дано целое положительное n. Найти
$\sqrt{\cdots \sqrt{\sqrt{3n} + 3(n-1)} + \cdots + 3}$.

7.19. Напечатать рисунок. Задаётся N — количество блоков При N=2 рисунок получается такой:

```
*| /
| /*
 /*|
/*|
*| /
| /*
 /*|
/*|
```

7.20. Гора. Задаётся N — высота горы. При n=3 рисунок получается такой:

```
  *
 ***
*****
```

7.21. Ромб Для заданной высоты ромба нарисовать его из символов

'*'

Пример

```
Высота: 4
  *
 ***
 ***
  *
```

7.22. Антиромб Для заданной высоты ромба нарисовать его из символов '*'

Пример

```
Высота: 4
*****
** **
*   *
*   *
** **
*****
```

7.23. Дано целое положительное n. Найти

$$\cfrac{1}{2^0+\cfrac{1}{2^1+\cfrac{1}{\vdots\cfrac{}{2^n+\cfrac{1}{2^{n+1}}}}}}$$

7.24. Дано целое $n > 0$, вещественное x. Среди чисел вида $a_i = e^{\cos x^{2i}} \cdot \sin x^{3i}, i = 1, 2, \ldots, n$ найти наибольшее, наименьшее и ближайшее к какому-либо целому.

7.25. Даны целые n, m $(0 \leqslant m \leqslant n \leqslant 12)$. Найти коэффициент бинома Ньютона $C_n^m = \frac{n!}{m!(n-m)!}$. Как зависит выбор алгоритма от используемого типа данных? Усложнение. Решите эту задачу для входных данных, больших 12.

7.26. Напечатать таблицу истинности бинарных логических операций *and*, *or*, *xor*.

7.27. Даны натуральные n, m $(2 \leqslant n \leqslant 10, 1 \leqslant m \leqslant 5)$. Напечатать все m-значные натуральные числа в n-ичной системе счисления.

7.28. Даны натуральные $N, M\,(N \leqslant M \leqslant 10^6)$. Найти все простые числа p, удовлетворяющие неравенствам $N \leqslant p \leqslant M$.

7.29. Нахождение квадратного корня. Даны положительные x, d. Найти номер и значение первого из членов последовательности a_i, построенной по правилу $a_1 = \frac{x+1}{2}$, $a_i = \frac{1}{2}\left(a_{i-1} + \frac{x}{a_{i-1}}\right)$, $i = 2, 3, \ldots$, квадрат которого отличается от числа x не более, чем на d (Формула *Герона Александрийского*).

7.30. Вычислить $1 + 2 + 3 + ... + N$

Пример

```
N=5
15
```

7.31. Если на написание цифры нужна одна секунда, сколько секунд понадобится для того чтобы выписать все целые числа от 1 до N (включительно)?

Пример

```
N=1000
1107
```

7.32. Если на написание цифры нужна одна секунда, сколько разных целых положительных чисел можно успеть выписать полностью за N секунд?

Пример

```
N=100
54
```

7.33. Вычислить значенение функции $f(x)$в точке x, $(|x| < 1)$ с помощью разложения её в бесконечный ряд

$$f(x) = \sum_{i=0}^{\infty} a_i$$

с точностью $\varepsilon > 0$. Требуемая точность считается достигнутой, если найдена сумма некоторого количества первых слагаемых, а последующее слагаемое оказалось по модулю меньше, чем ε. Полученный результат сравнить со значением, найденным с использованием соответствующих стандартных функций;

а) $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$;

б) $\dfrac{e^x - e^{-x}}{2} = \sum\limits_{i=0}^{\infty} \dfrac{x^{2i+1}}{(2i+1)!}$;

в) $\sin x = \sum\limits_{i=1}^{\infty} \dfrac{(-1)^{i-1} x^{2i-1}}{(2i-1)!}$;

г) $2 - e^{-x^2} - \cos x = \sum\limits_{i=1}^{\infty} (-1)^{i+1} x^{2i} \left(\dfrac{1}{i!} + \dfrac{1}{(2i)!} \right)$;

д) $(1 + 2x^2) e^{x^2} = \sum\limits_{i=0}^{\infty} \dfrac{2i+1}{i!} x^{2i}$;

е) $\dfrac{x - \sin x}{x^2} = \sum\limits_{i=1}^{\infty} (-1)^{i+1} \dfrac{x^{2i-1}}{(2i+1)!}, x \neq 0$;

ж) $2 \sin^2 x = \sum\limits_{i=1}^{\infty} (-1)^{i+1} \dfrac{(2x)^{2i}}{(2i)!}$;

з) $\cos \sqrt{x} - e^{-x} = \sum\limits_{i=1}^{\infty} (-1)^{i+1} x^i \left(\dfrac{1}{i!} - \dfrac{1}{(2i)!} \right), x \geqslant 0$;

и) $\sin x^2 - x^2 \cos x^2 = \sum\limits_{i=1}^{\infty} (-1)^{i+1} \dfrac{(2i) x^{4i+1}}{(2i+1)!}$;

к) $2x - x e^{-x^2} - \sin x = \sum\limits_{i=1}^{\infty} (-1)^{i+1} x^{2i+1} \left(\dfrac{1}{i!} + \dfrac{1}{(2i+1)!} \right)$;

л) $x + 1 - \dfrac{\pi^2}{12} - \ln(x+1) = \sum\limits_{i=1}^{\infty} (-1)^{i+1} \dfrac{(i+1) x^{i+1} + 1}{(i+1)^2}$;

м) $2x \cdot \operatorname{arctg} x - 2 \ln \sqrt{1 + x^2} = \sum\limits_{i=1}^{\infty} (-1)^{i+1} \dfrac{x^{2i}}{i(2i-1)}$.

7.34. Дано натуральное число $N \leqslant 2\,000\,000\,000$. Представить N в виде суммы квадратов натуральных чисел, содержащей наименьшее число слагаемых.

7.35. Найти значение коэффициента бинома Ньютона (см. задачу 7.25) за наименьшее количество операций умножения.

7.36. Дано натуральное число $N \leqslant 2\,000\,000\,000$. Представить N в виде суммы факториалов натуральных чисел, содержащей наименьшее число слагаемых.

7.37. Алгоритм Евклида. Даны целые неотрицательные a, b. Найти наибольший общий делитель (НОД) a и b. Для нахождения НОД a и b

можно воспользоваться алгоритмом Евклида:

$$\text{НОД}(a,b) = \begin{cases} a, & \text{если } a = b; \\ \text{НОД}(a-b,b), & \text{если } a > b; \\ \text{НОД}(a,b-a), & \text{если } a < b. \end{cases}$$

7.38. Алгоритм Евклида-2. Решить задачу 7.37, используя другой вариант алгоритма Евклида:

$$\text{НОД}(a,b) = \begin{cases} a, & \text{если } b = 0; \\ \text{НОД}(b,r), & \text{в противном случае;} \\ & r \text{ — остаток от деления } a \text{ на } b. \end{cases}$$

7.39. Разрезать. Какое минимальное число прямолинейных разрезов нужно сделать, чтобы разрезать прямоугольную пластину размером $m \times n$ на равные квадраты максимальной площади?

7.40. Разложить число. Дано натуральное число N. Разложить N в произведение простых сомножителей.

7.41. Даны два натуральных числа, длина десятичной записи которых не превышает 10 цифр. Из цифр этих чисел составить наибольшее возможное число, сохраняя первоначальное взаимное расположение цифр.

7.42. Максимальная последовательность символов. Написать программу, которая запрашивает имя файла и печатает длину самой большой последовательности из одного и того же символа.

Пример

В тексте задачи самая длинная последовательность "мм" и длина её 2.

В задачах 7.43-7.51 применяются ограничения: натуральные числа, если особо не оговорено, не превышают 2 миллиардов; нельзя использовать массивы, строковые переменные.

7.43. Даны натуральные числа N и M ($0 \leqslant M \leqslant 9$). Найти число вхождений цифры M в число N.

7.44. Дано натуральное число N. Найти число, получаемое из N выбрасыванием всех нечетных цифр.

7.45. Дано натуральное число N ($N \leqslant 40000$). Найти число, получаемое из N дублированием всех четных цифр.

7.46. Дано натуральное число N. Поменять порядок цифр числа N на обратный.

7.47. Дано натуральное число N. Определить, является ли это число палиндромом (десятичную запись можно дополнять незначащими нулями).

7.48. Дано натуральное число N. Указать самую длинную неубывающую подпоследовательность

а) идущих подряд цифр числа N;

б) цифр числа N, не обязательно идущих подряд.

7.49. Дано натуральное число N. Найти основание системы счисления $M \leqslant 10$, в которой запись числа N содержит наибольшее число единиц.

7.50. Дано натуральное число N. Указать N-ую цифру последовательности 149162536..., в которой выписаны подряд квадраты всех натуральных чисел.

7.51. Дано натуральное число N. Указать N-ую цифру последовательности 112358132134..., в которой выписаны подряд все числа Фибоначчи (см. задачу 1.13).

7.52. Дано вещественное неотрицательное N. Напечатать это число в двоичной, восьмеричной и шестнадцатеричной системах счисления.

7.53. Лишние пробелы. Минимизировать количество пробелов в заданной строке. В этой задаче требуется написать программу которая убирает лишние пробелы. Лишние — это начальные и конечные пробелы строки, а также те, которые идут подряд. Нужно оставить из группы пробелов только один.

Пример

```
Строка: ␣␣␣␣ааа␣␣ёёё␣␣␣ввв␣
ааа␣ёёё␣ввв
```

7.54. Запоминание текста. Читать строчки до тех пор, пока не введут пустую строку. После этого напечатать все прочитанные строки, кроме пустой, в том же порядке. Текст не будет содержать более 100 строк.

Пример

```
Это
такой
вот
текст...
```

```
Это
такой
вот
текст...
```

7.55. Обращение чисел. Прочитать N целых чисел ($N < 100$) и напечатать их в обратном порядке. Можно запрашивать N, читать очередное число и запрашивать окончание или поступать, как больше нравится.

Пример

```
3, 2, -1, 8, 4
Результат: 4, 8, -1, 2, 3
```

7.56. Сумма. Прочитать N целых чисел и вывести их через '+', а в конце написать '=' и вывести их сумму.

Пример

```
 3  2  4  -1  8  4  1
3+2+4+-1+8+4+1=21
```

7.57. ✠ Плавные числа. *Плавным числом* назовём десятичное натуральное число у которого любые две соседние цифры отличаются не более чем на единицу.

Техническое требование

Написать программу запрашивающую число (не более 100 знаков) и печатающую слово «Плавное», если число *плавное* и «Нет» в противном случае. Замечание: однозначное число — плавное.

Пример

```
Число: 12100012343
Ответ: Плавное
```

7.58. Вычислить $\underbrace{2 \cdot 2 \cdot \ldots \cdot 2}_{N \leqslant 30}$

Пример

```
N=3
8
```

7.59. Посчитать количество значащих цифр в заданном числе. Число от 0 до 2 000 000 000

Пример

```
Число: 1234567890
Цифр в нём: 10
```

7.60. Самый нужный символ. Почитать строку и сообщить сколько

раз встречается *нужный символ*. *Нужный символ* – символ который встречается в заданной строке чаще других.

Пример

программа
```
Нужный символ встречается (раз): 2
```

7.61. Сколько «ы»? Создать программу для подсчета количества букв «ы» в заданной строке. Ответ должен быть развернутый.

Пример

Мама мыла раму!
```
В строке "Мама мыла раму!", букв "ы" - 1.
```

7.62. Примерное среднее арифметическое. Вводятся целые числа до тех, пор пока не будет введён 0. Вывести то из введённых чисел, которое наиболее близко к среднему арифметическому. Среднее арифметическое — это число равное сумме чисел, делённой на их количество. Последний 0 не считается.

Пример

```
Вход: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
Выход: 5
```

7.63. Натуральное число называется *совершенным*, если оно равно сумме всех своих делителей, кроме самого себя (например, число $28 = 1+2+4+7+14$ является совершенным). Дано натуральное N, ($N \leqslant 10^6$). Найти все совершенные числа, не превосходящие N.

7.64. Дано натуральное число n. Напечатать таблицу перевода сантиметров в дюймы и наоборот, состоящую из n строк. Точность – два десятичных знака. Пример таблицы для $n = 5$:

cm	inch
1.00	0.39
2.00	0.79
2.54	1.00
3.00	1.18
4.00	1.57

7.65. Снежинка Коха. Определим плоскую геометрическую фигуру «снежинка» N-го порядка следующим образом: правильный треугольник со стороной a является «снежинкой» первого порядка. На средней трети каждой стороны «снежинки» как на основании построим «наружу» правильный треугольник со стороной втрое меньше исходной стороны.

Внешние стороны полученной фигуры образуют «снежинку» второго порядка. Процесс продолжается до N-го построения.

Даны вещественное $a > 0$ и целое $N > 0$. Найти площадь и периметр «снежинки» N-го порядка, если длина стороны «снежинки» первого порядка равна a.

7.66. Дано натуральное n. Найти все перестановки первых n натуральных чисел.

Пример

```
Введите n=3
Все перестановки из 3 чисел:
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

7.67. Даны натуральные n и m $(n \geq m)$. Найти все различные сочетания первых n натуральных чисел, взятых по m штук.

Пример

```
Введите n=3
Введите m=2
Все сочетания из 3 чисел по 2:
1 2
1 3
2 3
```

7.68. Выравнивание таблицы. Ввести прямоугольную таблицу чисел и вывести её, выровненную по столбцам.

Пример

```
Строк: 2
Столбцов: 3
Таблица: 1 -2 3 -4 5 -6
Результат:
  1  -2   3
 -4   5  -6
```

7.69. Телефонный номер называется «шахматным», если его цифры набираются на телефонном кнопочном номеронабирателе ходом шахматного коня. Написать программу, подсчитывающую, сколько можно набрать различных семизначных «шахматных» номеров, начинающихся с

заданной цифры.

7.70. Диагональная змейка чисел. Записать в прямоугольную таблицу из N столбцов и M строк в порядке возрастания все числа от 1 до M*N диагональной змейкой.

В левый верхний угол записываем 1. По второй диагонали располагаем числа 2 и 3 сверху вниз. На третьей диагонали числа от 4 до 6 снизу вверх. Так продолжаем до заполнения прямоугольника.

Техническое требование

Числа N и M вводятся.

При выводе столбцы должны получиться ровными.

Пример

```
N= 5
M= 4
  1  2  6  7
  3  5  8 14
  4  9 13 15
 10 12 16 19
 11 17 18 20
```

7.71. Прямоугольная спираль. Изобразить.

Техническое требование

Числа N и M вводятся.

При выводе столбцы должны получиться ровными.

Пример

```
N= 5
M= 7

 ------+
 +----+|
 |+---||
 |+---+|
 +-----+
```

7.72. Таблица умножения. Сделать программу для печати таблицы умножения натуральных десятичных чисел размером 10×10.

Таблица 4×4 выглядит так:

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

7.73. Двоичные числа Фибоначчи. Последовательность

0111011101100011011010 1 ...
сформирована из цифр последовательных чисел Фибоначчи (см. задачу 1.13), записанных в двоичной системе счисления. Дано натуральное $N (N \leqslant 6)$. Найти количество нулей и единиц этой последовательности вплоть до первой встретившейся группы из N нулей или единиц.

7.74. Простые соседи. Сформулируем тезис: "существует такое натуральное число N, что разность между любыми двумя соседними простыми числами, не превосходящими N, не больше 255, а для некоторых соседних простых чисел, больших N, это условие нарушается". Требуется либо найти N, либо попробовать доказать, что такого N не существует.

Кстати, если этот тезис верен, то для представления любого конечного подмножества последовательных простых чисел можно использовать массив байтов, как это сделано в п.7.2.3 [5].

7.75. Найти все простые и все составные числа Фибоначчи, не превосходящие данного N $(N \leqslant 2\,000\,000\,000)$.

7.76. Средневековая задача о старушке и яйцах. Старушка шла с корзиной яиц на рынок, когда её сбил с ног всадник, и все яйца разбились. Всадник предложил возместить убытки и спросил старушку, сколько у неё было яиц в корзине. Старушка ответила, что точно не помнит, но когда она брала яйца по два, оставалось одно яйцо. Также, когда она брала по 3, 4, 5, 6 яиц в остатке было одно яйцо. Но когда она брала по 7 яиц, остатка не было. Сколько могло быть яиц в корзине старушки? Задачу обобщить, приняв возможным различные значения делителей и остатков.

7.77. Даны веса n предметов $a_1, a_2, \ldots, a_n$, $(n \leqslant 16)$. Разделить предметы на две группы так, чтобы веса групп отличались бы минимально.

7.78. Треугольник. Дан треугольник из натуральных чисел: в первой строке – одно число, во второй – два и т.д. (числа не превосходят 100). Двигаемся от вершины треугольника вниз, каждый шаг может быть от числа к любому из его двух соседей внизу. Останавливаемся на основании треугольника. Построить такой порядок прохода по треугольнику, чтобы сумма пройденных чисел была бы минимальна.

Пример

```
Треугольник:
   5
  3 4
 2 4 5
```

```
6 8 2 4
Наименьшая сумма = 14.
```

7.79. Задан текстовый файл. Длина каждой строки текста не более 100 символов. Нужно преобразовать этот текст следующим образом:

- строки меньше 20 символов удвоить;
- пустые строки убрать;
- строки больше 90 символов отрезать до 90 (убрать несколько последних);
- у всех строк убрать все ведущие (в начале строки) пробелы.
- все заглавные 'Ы' заменить на строчные;
- строки после строки заканчивающейся на слово "СТОП" убрать;
- строки после строки заканчивающейся на точку убрать, из самой строки удалить эту точку;
- строки после строки "THE END" убрать, саму строку убрать;

7.80. В прямоугольном массиве записаны натуральные числа. Найти первую такую строку, в которой:

- либо все числа простые;
- либо все числа составные;
- либо сумма каких-то двух составных чисел простое число.

7.81. Запятые после слов. Записать программу, которая печатает запятую и пробел после каждого слова заданной строки, а в конце ставит точку. Слово – последовательность символов, в которой нет пробела.

Пример

```
Строка: 2+3=5␣-␣это␣␣␣такая␣строка.
2+3=5, -, это, такая, строка..
```

7.82. Уравнение $x! \cdot N = y!$.

Решить уравнение: $x! \cdot N = y!$. N – целое число, $K!$ — определяется только для целых $K \geqslant 0$. $K! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot K$, $0! = 1$.

Техническое требование

Получить для заданного N ($-2\,000\,000\,000 \leqslant N \leqslant 2\,000\,000\,000$) все различные пары (x, y).

Пример

```
N= 12
Ответ: (3, 4)  (11, 12)
```

Глава 8

Подпрограммы

8.1. Даны четыре положительных вещественных числа a, b, c, d. Для каждой тройки этих чисел определить, существует ли треугольник с такими сторонами и среди всех треугольников найти тот, у которого площадь максимальна.

8.2. Даны длины сторон некоторого треугольника a, b, c. Найти величины углов этого треугольника в градусах и длины его медиан.

8.3. Даны натуральные a, b и c. Найти наибольший общий делитель и наименьшее общее кратное этих трех чисел.

8.4. Решить задачу 8.3 для четырех натуральных чисел a, b, c, d.

8.5. Два натуральных числа называются *дружественными*, если каждое из них равно сумме всех делителей другого числа, кроме него самого, например, числа 1184 и 1210 – дружественные. Найти все пары дружественных чисел из первых пяти тысяч натуральных.

8.6. Последовательность чисел. Вводится последовательность целых чисел, заканчивающаяся нулём. Ноль не является членом последовательности. Преобразовывать последовательность следующим образом:

1) заменить отрицательные числа в последовательности на их модули и вывести получившуюся последовательность;
2) в полученной последовательности заменить чётные числа на максимальное число последовательности, вывести результат;
3) в полученной последовательности вычесть из каждого члена последовательности 2, вывести результат;
4) в полученной последовательности заменить отрицательные на 0, вывести результат.

Пример

```
Последовательность: 1  2  3  0
```

1) `1  2  3`
2) `1  3  3`
3) `-1  1  1`
4) `0  1  1`

8.7. Двадцатизначное число. Ввести 20-значное натуральное число. Вывести результат добавления к нему 1, вывести результат умножения его на 1, 10 и 2.

Пример

```
Число: 12345678901234567890
12345678901234567890 + 1 = 12345678901234567891
12345678901234567890 * 1 = 12345678901234567890
12345678901234567890 *10 = 123456789012345678900
12345678901234567890 * 2 = 24691357802469135780
```

8.8. Значение $f(a)$. Пусть $f(a)$ — сумма цифр целого числа a. Для заданного x вычислить: $f(f(x+2)+f(2\cdot x))\cdot f(x\cdot x)$.

Пример

```
x= 12
Выход: 18
```

8.9. Даны вещественные $a_1, a_2, b_1, b_2, c_1, c_2$. Если все корни одного из уравнений $a_1x^2+b_1x+c_1=0$, $a_2x^2+b_2x+c_2=0$ больше корней другого уравнения, то выдать на экран 1, во всех остальных случаях выдать 0.

8.10. Даны вещественные a, b, c. Среди всех перестановок этих чисел (a,b,c), (a,c,b), (b,a,c), ... найти такую, чтобы квадратное уравнение, коэффициенты которого совпадают с соответствующими числами в перестановке, имело бы два вещественных корня и расстояние на числовой оси между корнями было бы наибольшим. Если такой перестановки не существует, сообщить об этом.

8.11. Даны два массива натуральных чисел $a_1, a_2, \ldots, a_n$ и $b_1, b_2, \ldots, b_n$. Найти несократимую обыкновенную дробь, равную

а) $\frac{a_1}{b_1}+\frac{a_2}{b_2}+\cdots+\frac{a_n}{b_n}$;

б) $\frac{a_1}{b_1}-\frac{a_2}{b_2}+\cdots\pm\frac{a_n}{b_n}$;

в) $\frac{a_1}{b_1+b_2}+\frac{a_2}{b_2+b_3}+\cdots+\frac{a_n}{b_n+b_1}$.

8.12. Для используемой вами системы программирования перечислите все возможные способы доступа к глобальной переменной (переменной внешнего контекста) из подпрограммы, в которой определено такое же локальное имя.

8.13. Напишите функцию Аккермана (см. задачу 1.12) таким образом, чтобы при каждом вызове эта функция, помимо выполнения вычислений, выводила бы на экран текущее количество экземпляров этой функции, размещенных в программном стеке.

8.14. По данным значениям аргументов a и b вычислить значение функции $f(a,b)$. При организации вычисления необходимо выявить сходные по своей схеме последовательности операций, различающиеся лишь операндами, и представить их в виде отдельных функций. Аргументы должны принадлежать области определения функции:

а) $f(a,b) = (a^2+ab+b^2)^2 + (a^2+ab+b^2)b + b^2$;

б) $f(a,b) = \dfrac{\dfrac{a+b}{a-b} + a}{\dfrac{a+b}{a-b} - a}$;

в) $f(a,b) = \sqrt{\sqrt{a+1+\dfrac{1}{b}} + \dfrac{1}{b+1}}$;

г) $f(a,b) = \dfrac{\dfrac{ab}{a+b}\cdot b}{\dfrac{ab}{a+b} + b}$;

д) $f(a,b) = \dfrac{\dfrac{a^2-b^2}{a^2+b^2} - a - b}{\dfrac{a^2-b^2}{a^2+b^2} + a + b}$;

е) $f(a,b) = (a^2+b^2-ab)^2 + (ab)^2 - (a^2+b^2-ab)(ab)$.

8.15. Приведите пример, когда использование параметра-подпрограммы даёт значительные выгоды.

8.16. Параметры-функции. По данному значению натурального аргумента n вычислить значение функции f. При организации вычисления необходимо выявить сходные по своей схеме последовательности операций, различающиеся лишь операндами и/или используемыми функциями, и представить их в виде отдельных функций. Рекомендуется в качестве параметров передавать функцию (процедурный тип). Аргументы должны принадлежать области определения функции:

а) $f(n) = \dfrac{1^2+2^2+\cdots+n^2}{\cos(1)+\cos(2)+\cdots+\cos(n)}(\ln(1)+\ln(2)+\cdots+\ln(n))$;

б) $f(n) = \max\{\sum_{i=1}^{n} \sin^i i, \sum_{i=1}^{n} \cos^i i, \sum_{i=1}^{n} \operatorname{ctg}^i i\}$;

в) $f(n,x) = \min(\{\cos x^i\}_{i=1}^{n}) + \min(\{\sin x^i\}_{i=1}^{n}) + \min(\{\operatorname{tg} x^i\}_{i=1}^{n})$;

г) $f(x) = \dfrac{\sin\sqrt{x} + \sqrt{\sin x}}{\cos\sqrt{x} + \sqrt{\cos x}} + \dfrac{\operatorname{tg} x^2 + \operatorname{tg}^2 x}{\operatorname{ctg} x^2 + \operatorname{ctg}^2 x}$.

8.17. Уравнение в шестнадцатеричных. Решить уравнение

$$ax + b = cx - d.$$

a, b, c, d и x — целые шестнадцатеричные[1] числа. a, b, c, d — вводятся, x — выводится.

Пример

```
a=A b=B c=C d=D
x = C
```

8.18. Значение $f(x,y)$. Вычислить значение функции $f(x,y)$ при заданных x, y, если:
для любых целых x и y

$$\begin{array}{lll} f(0,y) & = y & , \\ f(x,0) & = x & , \\ f(x,x) & = f(x-1,x-1)+1, & \text{при } x > 0, \\ f(y,y) & = f(y+1,y+1)-1, & \text{при } y < 0, \\ f(x,y) & = f(x,x)+f(y,y), & \text{при } x \neq y. \end{array}$$

Пример

$f(-2,3) = 1$

8.19. Словообильная строка. В заданном файле найти строку содержащую максимальное количество слов. Если таких строк несколько, выдать первую из них. Словом будем считать непрерывную последовательность из русских букв.

8.20. Без параметров... Напишите рекурсивную версию функции, вычисляющей факториал целого положительного числа

а) без использования параметров;

б) без использования параметров и локальных данных.

Каковы трудности и недостатки такого подхода?

[1] Шестнадцатеричные числа записываются с использованием обычных десятичных цифр $0, 1, \ldots, 9$ и ещё букв A, B, C, D, E, F, обозначающих $10, 11, 12, 13, 14, 15$, соответственно.

8.21. Напишите функцию, возвращающую среднее арифметическое одного, двух или трех своих аргументов (в языке программирования должна быть возможность определения функций с различным числом параметров, например, как в C++).

Пример

ave(1)=1.000000000000
ave(1,0)=0.500000000000
ave(1,0,0)=0.333333333333.

8.22. Суперупорядочивание. Для расположения элементов набора данных в определённом порядке достаточно для любых нескольких элементов иметь возможность проверить, расположены ли они так, как нужно (*подпрограмма сравнения*). Напишите программу упорядочивания данных любой природы, если будет дана подпрограмма сравнения.

Техническое требование

Подпрограмма сравнения представляет собой функцию с неопределённым числом аргументов, которая возвращает логическое значение «ИСТИНА», если её аргументы расположены правильном порядке и «ЛОЖЬ» — в противном случае.

8.23. Санки. Предложите способ реализации механизма санков (thunk-параметров) для традиционных языков (например, C++, Pascal или их потомков), то есть подпрограмм, обеспечивающих доступ к фактическим параметрам контекста вызова данной подпрограммы (см. 8.5.3 [5]).

8.24. Параметры по необходимости. Даны действительные a, b и c. Напишите подпрограмму, решающую уравнение

$$\sqrt{\frac{ax + a^2}{bx + b^2}} = c.$$

В схеме решения (алгоритма) отметьте те точки, в которых впервые возникает необходимость в каждом из входных параметров a, b и c. Найдите направления (части) решения, в которых значения некоторых из параметров вооще не понадобились. Как при решении этой задачи можно использовать стратегию «ленивых вычислений» и «параметры по необходимости»(см. 8.5.3 [5])?

8.25. Ленивые вычисления. Даны четыре действительных числа a, b, c, d. Требуется найти сумму площадей четырех треугольников, построенных, как в задаче 8.1. Необходимо учитывать, что площадь треугольника (а, значит, и сумма площадей) может быть вычислена, если

только это треугольник существует. До каких пор в алгоритме можно откладывать вычисления ответа (или части ответа)? Как при решении этой задачи можно использовать стратегию «ленивых вычислений» и «параметры по необходимости»(см. 8.5.3 [5])?

8.26. Требуется написать подпрограмму, позволяющую сортировать линейный массив с определенным числом и различными типами элементов. Например, для Turbo Pascal возможны вызовы:

```
var ai: array[1..100] of integer;
    ac: array[1..100] of char;
    ar: array[1..100] of real;
 ...
 sort(ai,'i');
 sort(ac,'c');
 sort(ar,'r');
```

Второй параметр, передаваемый подпрограмме определяет тип элементов массива. Воспользуйтесь для передачи параметра-массива механизмом нетипизированных указателей (см. 8.5.3 [5]). Решите эту задачу для Turbo Pascal(или Delphi) и C/C++, сравните полученные решения. Которая из систем программирования сильнее придерживается концепции статического контроля типов?

8.27. Допустим, в некотором гипотетическом языке программирования имеются всевозможные механизмы передачи параметров подпрограммам (см. 8.5.3 [5]). Напишите различные осмысленные варианты заголовка подпрограммы сортировки линейного массива, используя различные механизмы передачи (в том числе, через глобальный контекст) и обоснуйте достоинства и недостатки каждого из определений заголовка. Для каждого заголовка сформулируйте спецификацию подпрограммы (описание интерфейса подпрограммы и производимого ею эффекта).

8.28. Суперсортировка. Для расположения набора данных в определённом порядке достаточно для каждого элемента знать правильное место он занимает или нет (*подпрограмма размещения*). Напишите процедуру упорядочивания данных любой природы, если будет дана подпрограмма размещения.

Техническое требование

Подпрограмма размещения представляет собой функцию с одним ар-

гументом для элемента и с одним аргументом для набора данных, которая сообщает, на правильном месте стоит элемент в наборе или нет.

8.29. Реализовать арифметические операции с *длинными числами*, то есть с целыми числами, десятичная запись которых может содержать до 255 цифр:

а) сложение двух длинных чисел;
б) умножение двух длинных чисел;
в) возведение длинного числа в натуральную степень;
г) вычисление факториала натурального числа;
д) деление длинного числа на длинное число с остатком.

8.30. Реализовать основные операции с вещественными многочленами вида $P(x) = p_n x^n + p_{n-1} x^{n-1} + \ldots + p_1 x + p_0$: сложение, разность, произведение многочленов, производная многочлена, значение многочлена в точке.

8.31. Решить задачу 8.30 для многочленов, коэффициенты, аргументы и значения которых являются комплексными числами.

8.32. Решить задачу 8.30 для многочленов, коэффициенты, аргументы и значения которых являются рациональными числами с выделенной целой частью и дробной частью в виде правильной несократимой обыкновенной дроби.

8.33. Решить задачу 8.30 для многочленов, коэффициенты, аргументы и значения которых являются длинными целыми числами (см. задачу 8.29).

8.34. Напишите модуль (библиотеку), реализующую тип данных – многочлен, и операции с ним – сложение, вычитание, перемножение, деление с остатком, вычисление значения в точке, нахождение производной многочлена. Напишите несколько различных демонстрационных программ, использующих этот модуль.

8.35. Напишите модуль (библиотеку), реализующую тип данных – рациональное число (дробь) с выделенной целой частью, и операции с ним – сложение, вычитание, перемножение, деление и другие. Если позволяют средства языка программирования, определите эти операции с помощью обычных знаков ($+, -, *$ и т.д.). Напишите несколько различных демонстрационных программ, использующих этот модуль.

8.36. Разные языки. Как известно, в результате компиляции исход-

ного текста (кода) программы получается объектный код (.obj) – эквивиалент исходной программы на машинном языке. Далее объектный код проходит стадию связывания (компоновки) с другими программными единицами (например, библиотечными модулями) для получения иполняемого кода программы (.exe). Следовательно, концептуально не столь важно, на каком языке был сформирован исходный код данного объектного модуля и можно в исходном коде программы подключать библиотеку (модуль), исходный текст которого был написан на другом языке. Например, модуль Turbo Pascal (unit), откомпилированный в объектный код (.tpu), может быть подключен к исходному коду на C/C++ (с точностью до деталей). Обратное тоже верно.

Напишите библиотеку (модуль) stack, моделирующий стек на Turbo Pascal. Решите на языке C/C++ задачу вычисления значения выражения, записанного в обратной польской записи с использованием модуля stack. Попробуйте выполнить обратное: модуль написать на C/C++, и подключающую его программу – на Turbo Pascal. Особенности использования модуля, созданного в иной языковой среде, можно выяснить в справочной системе.

8.37. ✣ **Числовое кольцо.** В кольце записаны N цифр, составляющих по часовой стрелке три числа: два слагаемых и сумму.

Техническое требование

Написать программу, которая запрашивает строку цифр и, считая её кольцом, печатает какое-нибудь решение в виде A+B=C. Все цифры должны входить в числа в порядке следования в кольце. Цифр в кольце не более 100.

Пример

```
Ввод: 01902021
Вывод: 190+20=210
```

8.38. Обобщение последовательности Фибоначчи. Построим обобщение последовательности чисел Фибоначчи следующим образом: последовательность

$$F^N(1), F^N(2), \ldots, F^N(k), \ldots$$

назовём последовательностью чисел N-Фибоначчи ($N \geqslant 2$– целое число), где

$$F^N(k) = \begin{cases} 1, & k \leqslant N \\ F^N(k-N) + F^N(k-N+1) + \cdots + F(k-1) & k > N \end{cases}$$

Таким образом, последовательность 2-Фибоначчи —
1 1 2 3 5 8 13 21 34 56 89...
это обычные числа Фибоначии, 3-Фибоначчи —
1 1 1 3 5 9 17 31 57 ...
Дано натуральное число A. Найти наименьшее целое $N \geqslant 2$, чтобы A было бы одним из чисел последовательности N-Фибоначчи.

Пример

```
Введите A= 57
Ответ: N=3
```

Вопрос: а всегда ли существует N для произвольного натурального A?

8.39. Числа типа Фибоначчи. Определим последовательность чисел типа Фибоначчи следующим образом:
$u_1 = a$,
$u_2 = b$,
$u_k = u_{k-2} + u_{k-1}$, если $k > 2$,
где a и b — некоторые целые числа. Дано целое число c. Требуется найти такие a и b, чтобы c было бы членом последовательности типа Фибоначчи и при этом сумма $|a| + |b|$ была бы минимальна.

8.40. Формула Бине. Известна нерекуррентая формула для вычисления чисел Фибоначчи:

$$F(k) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k}{\sqrt{5}}, k = 1, 2, \ldots.$$

Исследуйте и сравните зависимость времени вычисления k-го числа Фибоначчи от номера k по рекуррентной формуле и формуле Бине. Найдите более простую приближенную формулу для чисел Фибоначчи (для этого рассмотрите асимптотическое поведение функции $F(k)$).

8.41. ✣ **Спонж.** *Спонж (от англ. sponge — губка) порядка N* строится следующим образом. Для $N = 0$ это куб со стороной 1 см и весом 1 грамм. Спонжем порядка $N > 0$ будет фигура, полученная из спонжей порядка $N - 1$, расположенных *по соседству* с центральным, но самого центрального и соприкасающихся с ним гранью спонжей нет, то есть спонж имеет сквозные отверстия. *По соседству* — выше, ниже, правее, левее дальше и/или ближе ровно на один размер куба, содержащего спонж. Никакие повороты не допускаются.

На рисунке 8.1 приведён пример спонжа порядка 2.

Задача: вычислить вес спонжа заданного порядка.

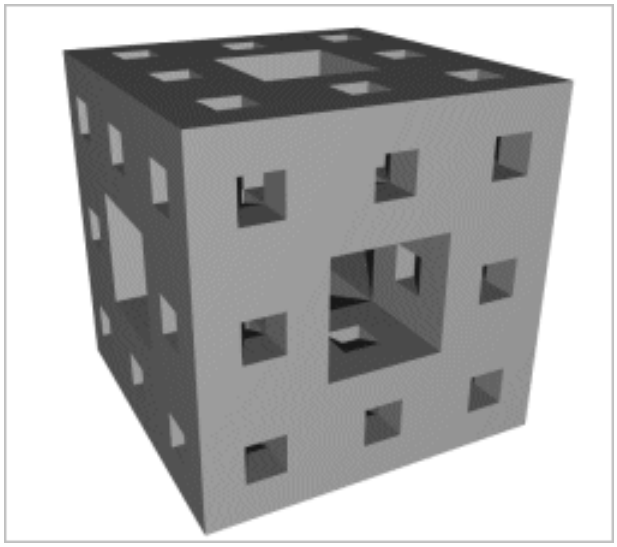


Рис. 8.1: Спонж второго порядка

Техническое требование

Написать программу, запрашивающую порядок спонжа и печатающую его вес.

Пример

```
Порядок= 2
Вес = 400
```

Глава 9

Структуры данных

9.1. Самый высокий. Запросить имя и рост каждого человека и напечатать имя самого высокого. Если таких несколько, напечатать любого.

Пример

Маша
120
Коля
050
Саша
150
Валя
0130
Результат: Саша

9.2. Цвет волос. Вводятся имена и цвета волос нескольких человек. Напечатать имена всех тех цвет волос у которых не уникален, т.е. в этом списке есть ещё люди с таким же цветом.

Пример

Дима чёрные
Коля русые
Валя красные
Саша красные
Оля русые
Результат:
Валя
Саша
Коля
Оля

9.3. Минимум. Прочитать N чисел и вывести самое маленькое из них.

Пример

3 2 4 -1 8 4 1
Результат: -1

9.4. Номер числа. Прочитать N чисел и спросить искомое число. Указать номер первого с начала числа, равного искомому или написать «нет таких».

Пример

10 -1 91 -1 15
Ищем: *-1*
Результат: 2

9.5. Упорядочивание. Прочитать N чисел и вывести их все в порядке увеличения, начиная с меньшего.

Пример

3 2 4 -1 8 4 1
-1 1 2 3 4 4 8

9.6. Вложенности. Для заданного N изобразить рисунок. Понять, что именно нужно изобразить можно из примера.

Техническое требование

Число N — нечётное, вводится.

Пример

N= 7

```
+--+--+
| / \ |
|/+-+\|
+ |+| +
|\+-+/|
| \ / |
+--+--+
```

9.7. Сокращение числа. Задано натуральное число, длина которого не больше 100 знаков. Верно ли, что убирая несколько раз по две рядом стоящие цифры, сумма которых равна 10, можно сократить всё число?

Пример

1234567891234567891
Результат: ДА

9.8. Дана десятичная запись вещественного числа a без периода. Найти запись этого числа в системе счисления с основанием R ($2 \leqslant R \leqslant 36$) (возможно появление периода).

9.9. Дано натуральное число $n \leqslant 32000$, записанное арабскими цифрами. Найти запись этого числа римскими цифрами (римские цифры означают: $I-1, V-5, X-10, L-50, C-100, D-500, M-1000$).

9.10. Какую пользу даёт использование в программе перечислений?

9.11. Логический тип. Приведите примеры языков программирования в которых:

- выделены логические константы «истина» и «ложь»;
- не выделены специальные логические константы;
- не выполнен закон $\neg\neg A = A$.

9.12. Уравнение $\mathbf{x + 2 \cdot x + 3 \cdot x + ... + x \cdot x = n}$. Решить уравнение: $x+2\cdot x+3\cdot x+...+x\cdot x = n$. Вводится $n < 2\,147\,483\,000$, найти x. Какими могут быть n и x, догадайтесь сами.

Пример

```
n= 18
x=3
```

9.13. ☐ **Уравнение $\mathbf{X^n = 2}$.** Решить уравнение $X^n = 2$ для заданного натурального n в рациональных числах с точностью $1/1000$. Под точностью, в данном случае, надо понимать следующее: $|2 - X^n| < 1/1000$

Техническое требование

Написать программу запрашивающую N и печатающую X — решение уравнения в виде несократимой дроби «числитель/знаменатель».

Пример

```
N=2
X=707/500
```

9.14. Вывести заданную строку в два столбца. В первом столбце символы с нечетными номерами, во втором — с чётными. Порядок символов остается прежним.

Пример

```
Строка:  строка
Результат:
ст
ро
ка
```

```
Строка:  12321
Результат:
12
32
1
```

9.15. Вывести заданную строку в два столбца. В первом столбце символы записанные слева от середины, во втором — справа. Порядок символов остается прежним.

Пример

```
Строка: строка
Результат:
со
тк
ра
```

```
Строка: 12321
Результат:
12
21
3
```

9.16. Вывести заданную строку в два столбца. В первом столбце символы записанные слева от середины, во втором — справа. Но в каждой получившейся строке должны быть символы, у которых расстояния до середины одинаковые.

Пример

```
Строка: строка
Результат:
са
тк
ро
```

```
Строка: 12321
Результат:
11
22
33
```

9.17. Экран пейджера. Вывести заданную строку в полосу шириной N символов. Порядок символов остается прежним.

Пример

```
Строка: Длинная строка
Ширина: 5
Результат:
Длинн
ая ст
рока
```

9.18. На вход с клавиатуры подается последовательность целых положительных чисел $a_1, a_2, \ldots, a_p$, заканчивающаяся нулем (нуль не входит в число членов последовательности; число p заранее не определено). Затем вводятся натуральные числа m и n, такие что $m \cdot n = p$. Разместить числа последовательности в левой верхней области матрицы в m строках по n элементов в каждой строке и вывести эту область матрицы на экран.

Ограничение: никакими другими структурами, кроме самой матрицы, пользоваться нельзя!

9.19. Слияние массивов. Даны значения элементов двух массивов x и y размера n, упорядоченые по неубыванию. Объединить значения этих массивов в новый массив z вдвое большего размера так, чтобы они также располагались по неубыванию.

9.20. Даны коэффициенты многочлена $P(x) = p_n x^n + p_{n-1}x^{n-1} + \ldots + p_1 x + p_0$. Найти

а) для данного x значение $P(x)$;

б) для данных x, a коэффициенты и значение многочлена $Q(x) = (x - a) \cdot P(x)$;

в) для данного x коэффициенты и значение производной многочлена $P(x)$: $R(x) = (P(x))'$;

г) для данных коэффициентов многочлена $S(x) = s_m x^m + s_{m-1}x^{m-1} + \ldots + s_1 x + s_0$ коэффициенты произведения многочленов $P(x) \cdot S(x)$;

д) для данных коэффициентов многочлена $T(x) = t_m x^m + t_{m-1}x^{m-1} + \ldots + t_1 x + t_0$, $1 \leqslant m \leqslant 5$ коэффициенты многочлена $P(T(x))$.

9.21. Как множества. Даны значения двух целочисленных массивов x и y размера n. Рассматривая массивы как конечные множества целых чисел, построить массив z размера не более $2n$, где

а) $z = x \cap y$ (пересечение множеств);

б) $z = x \cup y$ (объединение множеств);

в) $z = x \setminus y$ (разность множеств);

г) $z = x \triangle y$ (симметрическая разность множеств[1]).

9.22. Сортировка выбором. Среди элементов массива $a_1, a_2, \ldots, a_n$ выбирается наименьший и меняется местами с элементом a_1. Затем этот алгоритм применяется к элементам $a_2, a_3, \ldots, a_n$ и т.д. до тех пор, пока не останется последний элемент массива.

Написать программу, реализующую алгоритм сортировки выбором линейного вещественного массива.

9.23. Сортировка вставками. Алгоритм представляет собой последовательное применение $n-1$ раз следующего шага (для k от 1 до $n-1$): допустим первые k элементов $a_1, a_2, \ldots a_k$ уже отсортированы; для элемента a_{k+1} находится подходящее место среди первых k элементов и он вставляется на это место, при этом элементы, находящиеся правее места вставки, вплоть до a_k, сдвигаются на одну позицию вправо.

Написать программу, реализующую алгоритм сортировки вставками линейного вещественного массива.

9.24. Сортировка обменом (пузырьковая сортировка). Алгоритм представляет собой последовательное применение $n - 1$ раз следующего шага (для k от n до 2): последовательно сравниваются пары соседних элементов a_i и a_{i+1} (i от 1 до $k - 1$), если элементы в паре неупорядочены (то есть $a_i > a_{i+1}$), то они обмениваются значениями.

[1] Симметрическая разность определяется так: $A \triangle B = (A \cup B) \setminus (A \cap B)$.

Написать программу, реализующую алгоритм пузырьковой сортировки линейного вещественного массива.

9.25. Сортировка подсчетом. Для этого алгоритма требуется завести еще один массив — массив счетчиков. Для каждого элемента массива требуется подсчитать количество других элементов, не превосходящих его по значению. В случае равенства значений двух элементов, увеличивается счетчик того элемента, индекс которого больше. Таким образом вычисленное значение счетчика каждого элемента (увеличенное на единицу) будет являться индексом его нового положения в отсортированном массиве.

Написать программу, реализующую алгоритм сортировки подсчетом линейного вещественного массива.

9.26. Отсортировать элементы массива, имеющие нечетные индексы, по неубыванию, а четные индексы — по невозрастанию.
Ограничение: нельзя использовать дополнительные массивы.
Метод сортировки:

а) выбором;
б) вставками;
в) пузырьковая.

9.27. Отсортировать отрицательные элементы массива по невозрастанию, не затрагивая при этом остальные элементы, а неотрицательные — по неубыванию, не затрагивая остальные. Например, массив
3 −2 4 −3 −1 0
после сортировки должен выглядеть так:
0 −1 3 −2 −3 4
Дополнительные массивы использовать нельзя.
Метод сортировки:

а) выбором;
б) вставками;
в) пузырьковая.

9.28. Дана вещественная матрица A размера $m \times n$. Обозначим $A'(i, j)$ — верхний левый угол матрицы A до i строки и j столбца (подматрица). Каждому элементу исходной матрицы a_{ij} присвоить значение минимального среди элементов $A'(i, j)$.
Ограничение: разрешается в программе использовать единственную матрицу.

9.29. Седловые точки. Элемент матрицы называется *седловой точ-*

кой, если он является наименьшим в строке и наибольшим в столбце или, наоборот, наибольшим в строке и наименьшим в столбце. Для данной вещественной матрицы A размера $m \times n$ указать индексы всех седловых точек.

9.30. Соседи по матрице. *Соседом* элемента a_{ij} матрицы называется другой элемент a_{lk} этой же матрицы, если каждый из его индексов l и k отличается от соответственно i и j не более чем на 1. Дана вещественная матрица размера $m \times n$. Построить матрицу B такого же размера, чтобы каждый элемент b_{ij} этой матрицы был равен наименьшему среди соседей элемента a_{ij}.

9.31. Достроить забор. Матрица размера $m \times n$ заполнена нулями и единицами. Забором будем называть последовательность единичных элементов матрицы, таких, что два последовательных элемента забора являются соседями по горизонтали, вертикали или диагонали. Таким образом, заборы вместе с краями таблицы ограничивают друг от друга области, заполненные нулями (зоны).

Даны координаты двух нулевых элементов матрицы (заключенные) $(x_1; y_1), (x_2; y_2)$. Если они принадлежат разным зонам, то выдать число 0; если заключенные находятся в одной зоне, то выдать число, равное наименьшему количеству нулевых элементов матрицы, при замене которых на единицы заключенные будут находиться в разных зонах.

9.32. Подматрицы. Дана вещественная прямоугольная матрица порядка $M \times N$. Подматрицей назовем совокупность смежных элементов матрицы a_{ij}, индексы которых принадлежат отрезкам: $i \in [m_1, m_2]$, $j \in [n_1, n_2]$, где $1 \leqslant m_1 \leqslant m_2 \leqslant M$ и $1 \leqslant n_1 \leqslant n_2 \leqslant N$. Найти подматрицу данной матрицы с наибольшей суммой элементов.

9.33. Решето Эратосфена. Построить множество всех простых чисел, не превосходящих N. Для этого необходимо вначале построить множество всех натуральных чисел от 2 до N. Затем выбирают наименьшее из непросмотренных чисел (то есть 2) и исключают из множества все числа, кратные 2, кроме самого этого числа. Затем повторяют тот же шаг для $3, 5, \ldots$, пока в множестве имеются непросмотренные числа. В результате в множестве останутся только простые числа.

9.34. Упражнение Р. У. Хемминга. Все натуральные числа, которые можно представить в виде $2^i 3^j 5^k$, где i, j и k — целые неотрицательные, выстроим в возрастающую последовательность: $1, 2, 3, 4, 5, 6, 8, 9, 10, 12 \ldots$. Дано N — одно из чисел последовательности. Найти следующее

после N число последовательности.

9.35. Решите задачу о старушке и яйцах (см. задачу 7.76) с использованием метода решета, реализованного с помощью множественного типа. *Предупреждение:* стандартного или встроенного множественного типа может не хватить!

9.36. Другие счастливые числа. Имеем последовательность натуральных чисел $1, 2, 3, 4, 5, 6, 7 \ldots$. Число 1 оставляем в последовательности. Число 2 считаем «просеивающим» и исключаем каждое второе число последовательности, получается последовательность $1, 3, 5, 7, \ldots$. Следующее «просеивающее» число — 3, поэтому исключаем каждое третье число, получаем последовательность $1, 3, 7, 9, 13, \ldots$. Затем исключаем каждое седьмое число, каждое девятое и так далее. Числа, которые никогда не будут исключены из последовательности, называются счастливыми. Получить все счастливые числа, не превосходящие данного N.

9.37. Обмен множеств. В программе на языке Turbo Pascal заданы два множества. Первое хранится в переменной A, второе в — B. Написать код на языке Turbo Pascal такой, после выполнения которого, в A будет находиться значение B, а в B — значение A. Интерес представляет решение без использования дополнительных мест хранения, кроме A и B.

9.38. Написать программу, которая для заданных множеств целых чисел A, B, C проверяет, выполнены ли следующие равенства[2]:

а) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$;
б) $(A \setminus B) \setminus C = A \setminus (B \setminus C)$;
в) $(A \setminus B) \cup (A \setminus C) = A \setminus (B \setminus C)$;
г) $(A \setminus B) \cup (B \setminus C) = (A \setminus C) \cup (C \setminus B)$;
д) $(A \triangle B) \triangle (C \triangle D) = (A \triangle D) \triangle (C \triangle B)$;
е) $(A \triangle (B \triangle C)) = (A \cup B \cup C) \setminus (A \cap B) \setminus (A \cap C) \setminus (B \cap C)$;
ж) $(A \setminus B) \cup (B \setminus A) = (A \cup B) \setminus (B \cap A)$;
з) $A \cup (B \cap C) = A \cup B \cap C$;
и) $A \triangle (B \triangle C) = (A \triangle B) \triangle C$;
к) $A \triangle (B \cup C) = (A \triangle B) \cup (A \triangle C)$;
л) $A \triangle (B \cap C) = (A \triangle B) \cap (A \triangle C)$.

9.39. Своё множество. Реализовать тип данных «множество» с набором операций (конструктор множества, принадлежность элемента, объединение, пересечение, дополнение, симметрическая разность) для каждого из следующих случаев (см. 9.3.5 [5]): количество элементов множества

[2] Определение симметрической разности множеств см. в примечании к задаче 9.21

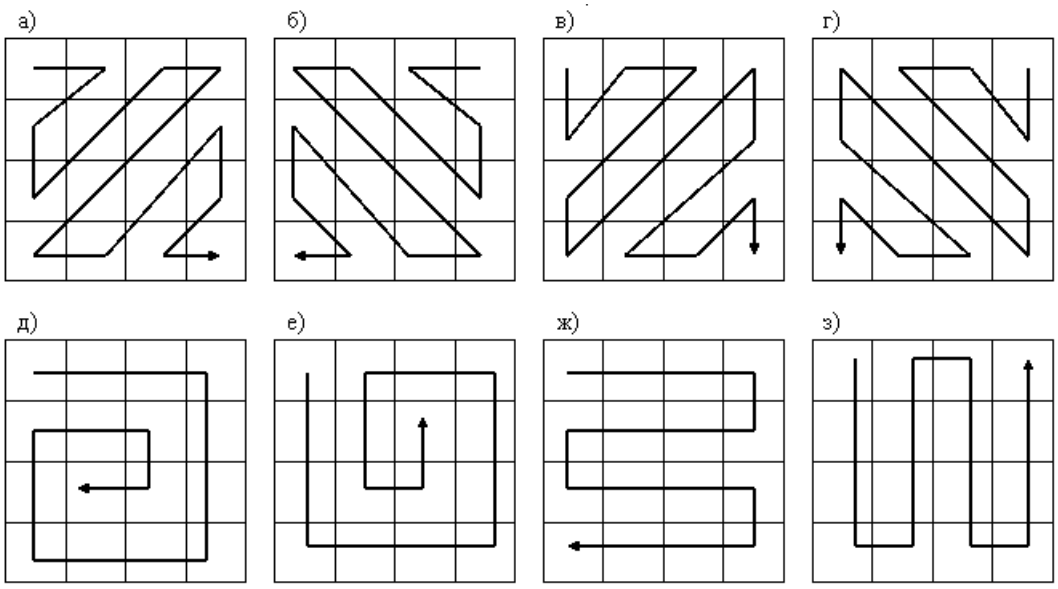


Рис. 9.1: Порядок расположения элементов матрицы.

а) невелико, не превосходит числа разрядов (порядка 2^5);
б) небольшое (до 2^8);
в) конечное, но довольно большое (2^8-2^{12});
г) очень большое, потенциально бесконечное (более 2^{12}).

9.40. Постройте пакет для работы с множеством бинарных деревьев. Реализуйте все разумные операции с множествами, в том числе операцию-итератор forall, позволяющий получить последовательность элементов множества.

9.41. Заполнение матрицы по змейке. Дано натуральное $N (N \leqslant 10)$. Заполнить квадратную матрицу размера $N \times N$ целыми числами $0, 1, 2, \ldots, N^2 - 1$ в соответствии со схемой расположения (см. рис. 9.1).

9.42. Сортировка элементов матрицы. Даны натуральное $N (N \leqslant 10)$, значения элементов целочисленной квадратной матрицы размера $N \times N$. Отсортировать матрицу таким образом, чтобы при обходе её элементов в порядке, представленном на рис. 9.1 они бы располагались по неубыванию. Метод сортировки:
а) выбором (см. задачу 9.22);
б) вставками (см. задачу 9.23);
в) пузырьковая (см. задачу 9.24);
г) подсчетом (см. задачу 9.25);

9.43. Сортировка матрицы с ограничением. Решить задачу 9.42 в условиях следующего ограничения: не допускается использовать дополнительных структур данных (например, линейных массивов), то есть сортировку нужно делать «на месте» — непосредственно в исходной матрице. Размер матрицы N вводится с клавиатуры.
Замечание: если для вычисления координат каждого элемента матрицы напрямую использовать циклические алгоритмы, как в задаче 9.41, то это существенно увеличивает временную сложность исходного алгоритма (по крайней мере — на два порядка). Рекомендуется найти такой алгоритм, в котором время выполнения сортировки было бы пропорционально квадрату количества сортируемых элементов (то есть так, как в случае с линейным массивом). Еще замечание: для сортировки методом подсчета ограничение ослабляется– разрешается использовать дополнительную матрицу счетчиков.

9.44. Решить задачу 9.43 для проямоугольной матрицы размера $M \times N$, где размеры вводятся, например, с клавиатуры.

9.45. Многочлены Чебышева. Дано целое k ($2 \leqslant k \leqslant 20$). Найти коэффициенты k-го многочлена Чебышева. Последовательность $T_i(x)$ *многочленов Чебышева* задается формулами: $T_0(x) = 1; T_1(x) = x; T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x), i = 2, 3, 4, \ldots$.

9.46. Обратная телефонная книга. Обратная телефонная книга устроена следующим образом: по номеру телефона (номер состоит из шести цифр, первые две из которых означают номер АТС) можно определить фамилию и адрес абонента. Информация должна быть расмещена по файлам (совсем не обязательно – текстовым), для каждой АТС – свой файл. Предложите структуру файла и способ доступа к нему, оптимальные по

а) расходу памяти;
б) времени доступа;
в) расходу памяти и времени доступа.

Как зависят выбираемые структуры файлов и алгоритмы доступа к ним от количества исходных данных? Реализуйте указанную систему поиска.

9.47. Сдвиг массива. Дан линейный массив из n записей (структура записи не важна, считать только, что запись достаточно большого размера). Требуется произвести циклический сдвиг записей массива на k позиций влево (то есть первые элементы последовательно перемещаются в конец массива).

Ввиду того, что копирование большой записи в новую позицию —

операция весьма затратная, нужно совершить указанный сдвиг с наименьшим количеством присваиваний значения записи.

9.48. Дана последовательность значений элементов целочисленного массива $a_1, a_2, \ldots, a_n$ и перестановка натуральных чисел $\{1, 2, \ldots, n\}$, которую обозначим $p_1, p_2, \ldots, p_n$, где $p_i \in \{1, 2, \ldots, n\}; p_i \neq p_j$ для $i \neq j$. Расположите исходные значения в массиве в порядке $a_{p_1}, a_{p_2}, \ldots, a_{p_n}$, не используя при этом дополнительных массивов.

9.49. Файловая система. Напишите программу, которая отображает структуру каталогов текущей файловой системы.

Примечание. Способ отображения структуры каталогов остается на Ваше усмотрение.

9.50. **Упаковка.** В файле хранятся только символы 'X', 'Y'. Размер файла всегда 16K. Написать две программы, одна из которых упаковывает содержимое исходного файла в файл меньшего размера, а другая обратно.

Техническое требование

Написать две программы, описанные выше. Обе программы читают по два имени файла из командной строки. Первая программа перекодирует первый файл во второй. Второй файл должен быть размером не больше 2K. Вторая программа декодирует первый файл (меньший) во второй (больший), содержимое которого (больший) должно в точности совпадать с первым для первого файла.

Время работы программы 1 минута.

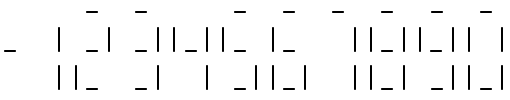
Пример

Файл first.xy: `XYXYXY... XYXY`
Вызов первой программы: `xypack first.xy second.cod`
После вызова первой программы:
файл second.cod: `8192XY`
Вызов второй программы: `xyunpack second.cod file.res`
После вызова второй программы:
файл file.res: `XYXYXY... XYXY`

9.51. Индикатор-8 Написать программу, которая для заданного целого числа (от -2000000000 до 2000000000) выводит его в виде «индикатора-8». Ниже приведен пример, из которого всё должно быть понятно.

Пример

Для *-1234567890*
на индикаторе



Глава 10

Автоматное программирование

10.1. Бесконечная последовательность
1123581321345689...
построена из цифр последоваельных чисел Фибоначчи. Дано натуральное N. Найти N-ную цифру указанной последовательности.

10.2. Бесконечная последовательность
1491625364964811001221...
построена из цифр квадратов последовательных натуральных чисел. Дано натуральное N. Найти N-ную цифру указанной последовательности.

10.3. Что распознаёт данный автомат:
A → 1B
B → CA|0
C → 0C|0
?

10.4. **ы^2.** Напишите конечный автомат, который распознаёт строчку из букв 'ы'. Длина таких строчек должна быть какой-либо степенью 2.

10.5. Что такое лексема?

10.6. Что такое парсер?

10.7. В чем отличие синтаксиса от семантики?

10.8. Создайте диаграмму переходов и таблицу графа состояний (функции перехода), напишите программу для конечного автомата, решающего следующую задачу. Целым числом называется любая непустая последовательность десятичных цифр от 0 до 9, ограниченная символами – не цифрами или границами входного потока (строки). Во входной по-

следовательности символов встречаются целые числа. Требуется найти и вывести максимальное значение. Напишите программу, реализующую конечный автомат (см. 10.2.4 [5] или §10.2 [6]).

Пример

```
Вход: ␣␣␣a123␣␣b␣,20c␣01␣2
Выход: 123
```

10.9. Создайте диаграмму переходов и таблицу графа состояний (функции перехода), напишите программу для конечного автомата, решающего следующую задачу. Дана последовательность десятичных цифр. Найти самую длинную неубывающую подпоследовательность. Напишите программу, реализующую конечный автомат (см. 10.2.4 [5] или §10.2 [6]).

Пример

```
Вход: 1325047887900
Выход: 04788
```

10.10. Создайте диаграмму переходов и таблицу графа состояний (функции перехода), напишите программу для конечного автомата, распознающего следующие конструкции какого-либо языка программирования:

а) вещественная константа;
б) идентификатор;
в) строковая константа.

10.11. Во входном тексте может встречаться бинарная операция возведения в степень, обозначаемая двумя звездочками, например, $a**b$. Заменить обозначение операции на знак ^ , например, $a\hat{}b$.

10.12. IP-адрес. На входе — IP-адрес в формате $x_1.x_2.x_3.x_4$, где x_i — целое неотрицательное число, принадлежащее отрезку $[0;255]$. Проверить запись IP-адреса на корректность.

10.13. Адрес e-mail. На входе — адрес электронной почты в формате $y_1.y_2.\cdots.y_n@x_1.x_2.\cdots.x_m$, где y_i, x_j — последовательности букв, цифр, некоторых знаков ('-', '_'). Проверить запись адреса электронной почты на корректность.

Пример

```
Введите адрес: nikolay._gogol_@narod-1.mail.ru
Да, запись адреса корректна.
```

Примечание: Точные правила написания адреса электронной почты предлагается найти самостоятельно.

10.14. Дата. На входе — запись даты в формате *dd.mm.yyyy*. Проверить дату на корректность и, если запись корректна, вывести следующую дату в том же формате.

Пример

```
Введите дату: 28.02.1900
Дата корректна.
Следующая дата 01.03.1900
```

10.15. Время. На входе указано время в формате чч : мм[: сс][*am*|*pm*]. Проверить запись на корректность и, если время записано корректно, вывести следующий момент времени в том же формате (то есть увеличить время на 1 секунду, если указаны секунды, или на 1 минуту, если секунды не указаны).

Пример

```
Вход: 11:59:59am
Выход: 12:00:00pm
```

10.16. Имя файла. Создайте диаграмму переходов и таблицу графа состояний (функции перехода), напишите программу для конечного автомата, решающего следующую задачу. На вход поступает последовательность символов, в которой приведено полное имя файла (с путём) в формате dir1/dir2/. . . /dirn/name.dst Вывести только непосредственно имя файла.

Пример.

```
Вход: Public/Ivanov/Files/prog_1+1.cpp
Выход: prog_1+1.cpp
```

10.17. На вход посимвольно поступает строка, в которую могут входить цифры от 0 до 9 и пробелы, длина строки не превосходит 300 символов. Признак окончания ввода — любой отличный от цифры и пробела символ. Последовательно идущие цифры (без пробелов) образуют числа, которые разделяются пробелами. Требуется найти сумму чисел строки. *Ограничение:* из строки разрешается хранить только единственный символ (последний введенный); считать, что количество идущих подряд цифр не превосходит 6.

Пример

```
Вход:␣12␣␣345␣␣␣6789q
Сумма=7146
```

10.18. Ё. Сколько заглавных букв «Ё», таких, что рядом нет ещё такой же буквы, в заданном файле?

Пример

```
В данной задаче букв 'Ё' - 2.
```

10.19. Пробелы. Верно␣␣ли,␣␣что␣␣в␣␣файле␣␣пробелы␣␣␣␣␣␣ встречаются␣␣чётными␣␣␣␣␣␣␣␣группами?

Пример

```
Вход: «Текст задачи»
Выход: Да
```

10.20. Напишите программу-препроцессор, проеобразующий таблицу переходов конечного автомата в текст программы-интерпретатора автомата на каком-нибудь языке программирования.

10.21. Предложение. Есть ли в файле заданное предложение? Предложения сравниваются с точностью до разделяющих пробелов, переходов на новую строку и табуляций. Знаки препинания — все символы отличные от букв (русских или латинских).
Напоминание: знаки препинания могут выполнять роль разделителей. Переносов в словах не будет.

Пример

Предложение:

```
Знаки препинания-все символы отличные от
     букв    (  русских или латинских   )     .

Есть в тексте задания.
```

10.22. Автомат с газировкой. Автомат по продаже газированной воды может воспринимать следующие внешние воздействия: опускание монеты (1 копейка), нажатие на кнопку выдачи напитка, нажатие на кнопку возврата денег. Кроме этого, автомат может исполнять следующие действия: принимать очередную монету (1 копейка), выдавать стакан напитка (если набрана сумма монет не менее 3 копеек и нажата соответствующая кнопка), возвращать всю сумму денег (если эта сумма имеется и нажата соответствующая кнопка). Нарисовать графы состояний автомата (в терминах автомата Мура и автомата Мили) и таблицу функции переходов. Построить программу, моделирующую работу автомата (желательно — каждым из четырех способов, см. 10.2.4 [5] или §10.2 [6]).

10.23. Дан линейный целочисленный массив $a_1, a_2, \ldots, a_n$. Найти число чередований знака (то есть количество пар a_i, a_{i+1} , что из a_i и a_{i+1} одно положительно, а другое — отрицательно) и число смен четности (то есть количество пар a_i, a_{i+1} , что a_i и a_{i+1} имеют разную четность). Требуется решить задачу за один проход по массиву.

Пример

```
Массив: 0 -1 -2 2 -3 3 4
Число чередований знака =3,
Число смен четности =4.
```

10.24. Прогрессии в массиве. В линейном вещественном массиве найти самую длинную цепочку подряд идущих элементов, являющихся членами арифметической или геометрической прогрессии.

10.25. Римское число. Дана запись числа в римской системе счисления. Найти его запись в десятичной системе или сообщить, что в записи допущена ошибка.

10.26. Азбука Морзе. Текстовое сообщение закодировано в азбуке Морзе, буквы азбуки отделяются пробелами. Вывести текст в обычном виде.
Замечание: постройте такую структуру хранения кодировки символов азбуки Морзе, чтобы алгоритм поиска нужной буквы был бы наименее затратным.

10.27. Баланс скобок. Считается, что в тексте соблюден баланс круглых скобок, если каждой открывающей круглой скобке далее по тексту соответствует закрывающая скобка. Определить, имеется ли баланс круглых скобок в данной строке.

10.28. Баланс разных скобок. В тексте могут быть скобки трех видов: круглые ( ), квадратные [ ], фигурные { }. Считается, что в тексте соблюден баланс скобок, если открывающей скобке каждого вида далее по тексту соответствует закрывающая скобка того же вида, причем часть текста между этими скобками также сбалансирована. Дана строка. Определить, соблюден ли баланс трех видов скобок в этой строке.

10.29. В строке особую роль играют символы '#'. Каждый такой символ отменяет предыдущий значащий символ (то есть любой, кроме '#'). Если символов '#' несколько, то они отменяют столько же предыдущих символов строки. Дана строка. Вывести результат строки после применения к ней всех символов '#' или сообщить, что корректно этого сделать нельзя.

Пример

```
Строка: ПП#РРЮ##ИВЕТЬ#
Результат: ПРИВЕТ
```

10.30. Дано натуральное число n. Найти

а) $\sum_{i=1}^{n}(-1)^{f(i)}$;

б) $\sum_{i=1}^{n}(-1)^{\text{Ц}(f(i))}$;

в) $\prod_{i=1}^{n}(-1)^{f(i)}$.

Здесь $f(i)$— числа последовательности Фибоначчи (см. задачу 1.13), значение функции $\text{Ц}(x)$ равно количеству цифр десятичной записи целого числа x.

10.31. Дан текст, слова которого являются последовательностями букв, отделёнными друг от друга пробелами или знаками препинания. Удалить из текста подряд идущие одинаковые слова, оставив только одно вхождение слова.

10.32. Даны натуральные числа N $(N \leqslant 2\,000\,000\,000)$ и m $(2 \leqslant m \leqslant 36)$. Является ли палиндромом запись числа N в системе счисления с основанием m?

10.33. Вычисление. Вычислить значение арифметического выражения, записанного с использованием шестнадцатеричных целых чисел, стандартных операций $(+, -, /, *)$ и круглых скобок. Для обозначения шестнадцатеричных цифр могут быть использованы строчные или прописные буквы.

Пример

```
-a+F/(1*F)-d0
Ответ: -d9
```

10.34. **Крестики-нолики 4×4.** В крестики-нолики 4×4 играют два игрока. Игра ведется на квадратном поле, разделенном на 16 клеток — по 4 в каждом ряду и столбце. Ходы выполняются по очереди, начинают крестики. За один ход игрок может поставить два своих значка, первый — «крестики», второй — «нолики».

Выигрывает тот, кто первый займёт своими значками одну из вертикалей, горизонталей или диагоналей. Написать программу игры в крестики-нолики 4×4 с пользователем в режиме диалога.

Техническое требование

Должна быть предусмотрена возможность выбора значка (т.е. очерёдности хода). Необходимо предусмотреть реакцию на нарушение правил или некорректные действия. Диалог должен вестись в текстовом режиме, где «крестики» обозначаются значком 'X', а «нолики» — '0'.

Способ указания места для знака оставлен на усмотрение программиста и должен быть понятен для пользователя (например, использование управления курсором).

10.35. **Шашки по Чапаеву.** На классической шахматной доске расставляются белые и чёрные шашки. На каждой вертикали — по одной шашке каждого цвета. Белая шашка всегда стоит ниже черной. Ходы выполняются по очереди. Во время хода можно передвигать несколько своих шашек по вертикали на несколько клеток вниз или вверх. Ход допустим, если выполнены следующие условия:

- каждая шашка может двигаться лишь в одном направлении в один ход, но разные шашки могут двигаться в разных направлениях;
- шашка не может выйти за край доски;
- нельзя вставать на шашку противника или переходить за неё;
- за один ход ходят не более двух шашек;
- одно перемещение обязательно.

Проигрывает тот, чьим шашкам некуда ходить.

Написать программу, которая играет в такую игру и выигрывает, когда это возможно.

Техническое требование

Программа должна читать файл с именем *move.txt* в текущей директории и записывать свой ответ в этот же файл.

Формат файла *move.txt*

Шестнадцать чисел, каждое в своей строке. Пары чисел в порядке следования указывают расположение шашек на вертикалях слева направо. Первое число пары показывает положение белой шашки, второе — чёрной. Очевидно, что первое число меньше второго.

Затем в новой строке следует символ W или B. Во входном файле B означает, что программа должна сделать ход за черных, W — за белых. В выходном файле символы W и B должны обозначать того чей ход следующий. В случае, если хода нет или программа сдалась, в файл *move.txt* нужно записать число 0. Нарушение правил или технических требований считается проигрышем программы.

На партию отводится не больше 100 ходов. На обдумывание хода не более 10 секунд.

Пример

Исходное состояние файла *move.txt*:

```
1
8
1
8
```

```
1
8
1
8
1
8
1
8
1
8
1
8
W
```

Такому файлу соответствует следующая позиция:

B	B	B	B	B	B	B	B
W	W	W	W	W	W	W	W

В этом случае в файл может быть записан такой ответ:

```
1
8
1
8
1
8
7
8
1
8
1
8
1
8
1
8
B
```

10.36. Дана строка символов. Если каждый символ строки заменить его кодом в двоичной системе счисления, то получим последовательность нулей и единиц (таблица кодировки символов — любая). Определить, является ли указанная последовательность палиндромом?
Дополнение: считать, что операция получения очередной двоичной цифры очень накладная, поэтому ответ желательно получить за наименьшее число таких операций.

10.37. Шаблон. Дан шаблон для имени файла (или для произвольной строки), который может содержать, кроме обычных символов, символы '*' (означает любую последовательность символов, в том числе – пустую) и '?' (любой один символ либо его отсутствие). Кроме этого, вводится строка. Проверить, подходит ли она под шаблон.
Пример. `Шаблон: ?a**n?.c?p`
`Строка: cartoon.cpp`
`Ответ: Да, строка подходит под шаблон.`

10.38. Вхождение по шаблону. Дан шаблон и строка символов (как в задаче 10.37). Найти в строке наидлиннейшую подстроку, подходящую под данный шаблон, или сообщить, что таковой не существует.
Пример. `Шаблон: a*?b?c`
`Строка: bbaabacbccc`
`Ответ: Подходит подстрока = aabacbc`

10.39. Без комментариев. Дана программа на языке Turbo Pascal. В тексте могут быть комментарии, ограниченные фигурными скобками, либо скобками вида '(*' и '*)'. Вывести текст программы без комментариев.

10.40. Снова без комментариев. В тексте особую роль играют символы '+' и '−', после которых в круглых скобках должны быть две непустые последовательности символов, разделяемые запятой. Эти две последовательности являются обозначениями соответственно начала и конца комментария. Символ '+' включает режим комментирования (то есть далее по тексту все, что между этими символами считается комментарием), '−' его отключает (если такой режим был включен). В тексте могут быть включены одновременно несколько режимов комментирования с различными обозначениями комментариев. Считается, что открывающим комментарий скобкам всегда далее по тексту соответствуют закрывающие.

Дан текст. Вывести этот текст без комментариев и управляющих ком-

ментариями конструкций '+' и '−'.

Пример

Вход: *Мама +({,}) мы{ло}ла р+(/*,*/)а/***/-({,}){му/*{*}*/*}*

Выход: Мама мыла ра{му*}

10.41. Период строки. Периодом строки $s = a_1a_2 \dots a_n$ назовем такое целое положительное число p, что строка s может быть представлена как сцепление (конкатенация) k одинаковых строк длины p $(k \geqslant 1)$, то есть $s = (w)^k = \underbrace{ww \dots w}_{k}$, где $w = b_1b_2 \dots b_p$.

Дана строка s, длина которой не превышает 255 символов. Найти все её периоды.

Пример.

Вход: *abababab*

Выход: Периоды строки:

8 (w=abababab, k=1)

4 (w=abab, k=2)

2 (w=ab, k=4)

10.42. Строки Фибоначчи. Определим последовательность строк Фибоначчи $s_1, s_2, \dots, s_k, \dots$ следующим образом: $s_1 = "b", s_2 = "a", s_k = s_{k-1}s_{k-2} (k > 2)$. Тогда начало последовательности строк Фибоначчи выглядит так:

b a ab aba abaab abaababa abaababaabaab

а) Чему равна длина s_k?

б) Дано k. Найти s_k.

в) Дана строка s. Проверить, не является ли она строкой Фибоначчи?

10.43. Обобщенный палиндром. Введем рекуррентное определение:

Строка символов s называется обобщенным палиндромом (ОП), если выполняется ровно одно из трех условий:

а) s — пустая строка (то есть не имеет ни одного символа);

б) s состоит из одного символа;

в) s можно представить так: $s = uxu$, где u — непустая строка, а x является ОП (то есть префикс и постфикс совпадают, а между ними — ОП).

Например, строки "abcdedabc", "abaababaabaab" являются ОП.

Дана строка. Определить, является ли она обобщенным палиндромом?

10.44. Напишите распознаватель в виде
- таблицы;
- программы

для следующих языков:
а) $L_1 = \{uxu | u \in (a|b)^*\}$;
б) $L_2 = \{a^n b^m c^n d^m | 0 \leqslant n, 0 \leqslant m\}$;
в) $L_3 = \{a^n b^m c^{n+m} | 0 \leqslant n, 0 \leqslant m\}$.

10.45. Напишите распознаватель в виде
- таблицы;
- программы

для языка, предложения которого являются последовательностями вида $a^k b^l c^m d^n \ldots$, то есть состоят из подпоследовательностей строчных букв латинского алфавита (возможно, пустых), взятых в алфавитном порядке.

10.46. Преобразовать грамматику, устранив левую рекурсию:
а) <идент>::=<буква>|<идент><буква>|<идент><цифра>
б) <list>::=<list>,<atom> | <atom>
<atom>::=a|b

Напишите распознаватель для языка, порождаемого данной грамматикой.

10.47. Преобразуйте грамматику, устранив неопределенность:
а) <условный>::= if <выражение> then <оператор> |
if <выражение> then <оператор> else <оператор>
б) <expr>::=<term>+<expr>|<term>−<expr>
в) грамматика из задачи 2.32.

Напишите распознаватели для языков, порождаемых данными грамматиками (неопределенные здесь нетерминальные символы доопределите самостоятельно или используйте источники, например [5] или §10 [6]).

10.48. Текст-таблица. Задан текст. Известно, что в каждой строке записаны фамилия, имя, отчество и прочие сведения. Имя, фамилия и отчество — последовательность символов без пробела. Требуется написать программу, которая выровняет этот текст. Программа должна добавить или убрать пробелы так, чтобы все фамилии, имена, отчества и прочие сведения начинались в одной и той же позиции на строке.

Пример

До выравнивания:

```
Smith Иванович Иванов сведения отсутствуют
Петров-селёдкин Билл Билл Билл Билл Билл
```

1 12 123 1 2 3 4 5 6
Пушкин Александр Сергеевич
После выравнивания:

```
Smith            Иванович   Иванов     сведения отсутствуют
Петров-селёдкин  Билл       Билл       Билл Билл Билл
1                12         123        1 2 3 4 5 6
Пушкин           Александр  Сергеевич
```

10.49. Задан текст. Известно, что в каждой строке записаны, фамилия, имя отчество, прочие сведения и личный код. Имя, фамилия и отчество — последовательность символов без пробела. Личный код — натуральное число. Требуется написать программу, которая выровняет этот текст. Программа должна добавить или убрать пробелы так, чтобы все фамилии, имена, отчества, прочие сведения и код начинались в одной и той же позиции на строке.

Пример

До выравнивания:
Smith Иванович Иванов сведения 1234
Ф-Ы Билл Билл Билл Билл 100000
1 12 123 1 2 3 4 5 6 7
Пушкин Александр Сергеевич 112836376458763
После выравнивания:

```
Smith   Иванович   Иванов     сведения     1234
Ф-Ы     Билл       Билл       Билл Билл    100000
1       12         123        1 2 3 4 5 6  7
Пушкин  Александр  Сергеевич                112836376458763
```

10.50. Задан текст. Известно, что в каждой строке записаны фамилия, псевдонимы и прочие сведения. Фамилия и псевдоним — последовательность символов без пробела. Псевдонимов может быть несколько. Требуется написать программу, которая выровняет этот текст. Программа должна добавить или убрать пробелы так, чтобы все фамилии, псевдонимы и прочие сведения начинались в одной и той же позиции на строке.

Пример

До выравнивания:
Иванов Петров Сидоров сведения отсутствуют
Петров-селёдкин селёдка Пётр килька всё ясно
1 1 2 3 4 5 6 это просто числа
Пушкин Саша Сергеевич это поэт
После выравнивания:

```
Иванов            Петров Сидоров       сведения отсутствуют
Петров-селёдкин   селёдка Пётр килька  всё ясно
1                 1 2 3 4 5 6          это просто числа
Пушкин            Саша Сергеевич       это поэт
```

10.51. ✣ **Нужные числа.** Назовём число *нужным с индексом N*, если в десятичной записи этого числа есть подряд все цифры числа N и в том же порядке. Другими словами, подстрока N в строке, если считать записи чисел строками.

Техническое требование

Найти количество нужных чисел с заданным индексом N среди натуральных чисел до 1 000 000 000. Число N — целое и неотрицательное до 1 000 000 000. Незначащие нули не учитывать.

Программа должна запрашивать N и выдавать количество чисел.

Пример

```
Количество нужных чисел с индексом 03112002 равно 300
```

10.52. Эквивалентность опциональных выражений. Опциональные выражения — выражения построенные из маленьких латинских букв, знака '?' и круглых скобок.

Знак '?' показывает, что выражения может и не быть, ставится после выражения, на которое действует. Круглые скобки показывают, область действия знака '?'.

Множеством опционалъного выражения назовём все строки, которые можно получить применив операцию '?'. Например, множеством выражения "a?b?c" будет $\{abc, ac, bc, c\}$.

Опциональные выражения будем считать эквивалентными, если совпадают их множества.

Написать программу, которая читает два опциональных выражения и определяет их эквивалентность. Длина строк, задающих выражения не более 100 символов.

Пример

```
a?b?c
(a?b?)?c
Эквивалентны
```

10.53. Вложенность опциональных выражений. Заданы опциональные выражения (см. определение в задаче 10.52) A и B. Написать программу для их сравнения. Ответ выдать в виде A>B, если множество выражения B является собственным подмножеством[1] множества выра-

[1]Множество X является *собственным подмножеством* множества Y, если X яв-

жения A; A<B, если множество выражения A является собственным подмножеством множества выражения B; A#B если ни то и не другое. Длина строк, задающих выражения не более 100 символов.

Пример

```
A= (abc)?
B= (a?b?c?)?
A<B
```

10.54. Пересечение опциональных выражений. Заданы опциональные выражения (см. определение в задаче 10.52) A и B. Написать программу, которая будет строить опциональное выражение, описывающее пересечение множеств выражений A и B. Длина строк, задающих выражения не более 100 символов.

Пример

```
A= (abc?)?
B= (a?bc)?
(abc)?
```

10.55. Эквивалентность регулярных выражений. Регулярные выражения —выражения построенные из маленьких латинских букв, знака '+' и круглых скобок.

Знак '+' показывает, что регулярное выражение нужно повторить несколько раз (может быть ни разу), ставиться после выражения на которое действует. Круглые скобки показывают, область действия знака '+'.

Множеством регулярного выражения назовём все строки, которые можно получить, применив операцию '+' любое количество раз.

Регулярные выражения будем считать эквивалентными, если совпадают их множества.

Написать программу, которая читает два регулярных выражения и определяет их эквивалентность. Длина строк, задающих выражения не более 100 символов.

Пример

```
(ab+c)+
ab+c(ab+c)+
Эквивалентны
```

10.56. Вложенность регулярных выражений. Заданы регулярные выражения (см. определение в задаче 10.55) A и B. Написать программу для их сравнения. Ответ выдать в виде A>B, если множество

ляется подмножеством Y и они не равны.

выражения B является собственным подмножеством множества выражения A; A<B, если если множество выражения A является собственным подмножеством множества выражения B; A#B, если ни то и не другое. Длина строк, задающих выражения не более 100 символов.

Пример

```
A= (abc)+
B= (a+b+c+)+
A<B
```

10.57. Пересечение регулярных выражений. Заданы регулярные выражения (см. определение в задаче 10.55) A и B. Написать программу, которая будет строить регулярное выражение, описывающее пересечение множеств выражений A и B. Длина строк, задающих выражения не более 100 символов.

Пример

```
A= (abc+)+
B= (a+bc)+
(abc)+
```

10.58. Мощность регулярных выражений. Пусть строки, порождаемые регулярными выражениями, ограничены длиной в 1000 символов. Мощностью регулярного выражения назовём количество строк в множестве этого выражения. Будем обозначать мощность выражения A как |A|.

Пусть заданы регулярные выражения (см. определение в задаче 10.55) A и B. Сравнить их мощности. Длина строк, задающих выражения не более 100 символов.

Пример

```
A= (abc+)+
B= (a+bc)+
|A|=|B|
```

10.59. Расстояние Хемминга. Расстоянием Хемминга между строками одинаковой длины s_1 и s_2 называется количество несовпадающих симоволов в s_1 и s_2, стоящих на одинаковых позициях. Даны две строки. Найти расстояние Хемминга между ними.

10.60. Расстояние. Расстоянием между строками s_1 и s_2 называется минимальное количество вставок и удалений символов из строки s_1 для преобразования её в строку s_2. Даны две строки. Найти расстояние между ними.

10.61. Расстояние редактирования. Расстоянием редактирования

между строками s_1 и s_2 называется минимальное количество вставок, удалений и замен символов из строки s_1 для преобразования её в строку s_2. Даны две строки. Найти расстояние редактирования между ними.

10.62. Подсказка при вводе. Программа вводит последовательность слов, каждое — с новой строки. В процессе ввода каждого слова программа предлагает возможные варианты продолжения, если введенная начальная часть слова совпадает с началом одного или нескольких ранее введенных слов. Например, после ввода слов 'компилятор' и 'коммунизм', при вводе первых трех букв слова 'компьютер', программа должна предоставить список возможных продолжений ('пилятор', 'мунизм'). Пользователь должен иметь возможность выбора одного из предлагаемых вариантов или продолжить ввод слова. После ввода пустой строки программа должна выдать список всех введенных слов (без повторов) и завершить работу. Вопрос для обдумывания: какая структура хранения слов будет наиболее оптимальной с точки зрения времени поиска и исключения повторных просмотров массива слов?

10.63. Метаграмма «Из мухи слона». Существует интересная словесная игра (метаграмма). Требуется из слова s_1 получить слово s_2 за конечное число преобразований. Игра начинается со слова s_1, его заменяют таким словом, чтобы они отличались не более чем на одну букву. Затем заменяют новое слово и так далее до тех пор, пока не получат слово s_2. Цепочка полученных слов (если она существует) является решением метаграммы. Дано множество слов (например, русских), и даны два слова s_1 и s_2 одинаковой длины. Построить цепочку преобразований из слова s_1 в слово s_2.

Пример

(считается, что словарь слов задан)

Вход: *миг эра*

```
Выход: миг - мир - мор - бор - боа - бра - эра
```

10.64. Метаграмма М. Гервера. Немного изменим правила игры в метаграммы (см. задачу 10.63). За каждый ход в слове не только заменяется одна буква на другую, но также разрешается менять порядок букв в слове. Задание то же: построить цепочку преобразований из исходного слова в конечное.

10.65. Горячие клавиши. Известно, что в командах меню применяются горячие клавиши, то есть некоторые выделенные буквы в названиях команд, нажатие на которые выполняет команду (например, **N**ew, **O**pen, **S**ave, Save **A**s,...). Даны несколько слов-пунктов меню (англий-

ских или русских), состоящих только из строчных букв. Требуется в каждом слове выделить одну букву (сделать её прописной), которую можно использовать как горячую клавишу, или сообщить, что этого сделать в данном наборе слов нельзя.

10.66. Третий по величине. Дана последовательность различных n натуральных чисел $a_1, a_2, \ldots, a_n$. Найти значение третьего по величине элемента этой последовательности.

10.67. Нарезать строку. Дана строка символов. Найти минимальное число разрезов, чтобы получившиеся части представляли собой неубывающие или невозрастающие последовательности символов (символы сравнивать в соответствии с их кодами, например, ASCII).

10.68. Рассмотрите программу нахождения корней квадратного уравнения. Определите, какие состояния можно выделить при анализе текста программы.

10.69. Без состояний. Приведите пример алгоритма, который нельзя записать ни в каком ином стиле кроме, как с помощью состояний.

10.70. Б **Узел.** На плоскости разложена завязанная в узел кольцевая веревка. Требуется ее развязать, т. е. разложить в виде кольца без пересечений.

Исходные данные

Веревка считается разложенной на прямоугольнике m на n (m клеток по вертикали, n клеток по горизонтали, $m, n \leqslant 11$). В каждой клетке задан вид расположения веревки следующими кодами из таблицы 10.1.

На вход подаются числа m, n и матрица кодов (последовательных целых чисел, разделенных пробелами и переводами строк). Входные данные находятся в файле *input.txt*.

Выходные данные

На экран выдаются последовательные промежуточные состояния процесса развязывания узла в виде (псевдо)графических изображений или исходное положение узла и слово «невозможно», если узел не развязывается. В результате развязывания должен получиться квадрат размером 2 на 2 как на рисунке 10.2.

Веревка считается неограниченно сжимаемой и растягиваемой. Прямоугольник, на котором она разложена можно также расширять в любую сторону.

Выдача последовательных изображений управляется нажатием клавишей "пробел" (для перехода к следующему изображению) и "backspace"

№	Пояснение	Изображение

0	пусто	`.....`
		`:   :`
		`:...:`

1	веревка расположена вертикально	`..|..`
		`: | :`
		`:.|.:`

2	веревка расположена горизонтально	`.....`
		`-----`
		`:...:`

3	левый нижний угол	`..|..`
		`: +--`
		`:...:`

4	левый верхний угол	`.....`
		`: +--`
		`:.|.:`

5	правый верхний угол	`.....`
		`--+ :`
		`:.|.:`

6	правый нижний угол	`..|..`
		`--+ :`
		`:...:`

7	пересечение, горизонтальная часть — сверху	`..|..`
		`-----`
		`:.|.:`

8	пересечение, горизонтальная часть — снизу	`..|..`
		`--|--`
		`:.|.:`

Рис. 10.1: Элементы отображения узла

```
.........
: +---+ :
:.|.:.|.:
: +---+ :
:...:...:
```

Рис. 10.2: Развязанный узел

("назад", "←" для возврата к предыдущему изображению). За один шаг можно перекладывать веревку только на соседние клетки (по вертикали, по горизонтали или по диагонали, в любом числе клеток). Допустимые комбинации расположений веревки в одной клетке — те, коды которых перечислены выше. Прямоугольник не разрешается увеличивать до размера большего, чем 25 на 80.

Пример

№ 1

Входные данные:

```
4 4
4 2 5 0
1 4 8 5
3 8 7 6
0 3 6 0
```

Результат:

```
. . . . . .
: +-------+ :   :
:.|.:...:.|.:...:
: | : +---|---+ :
:.|.:.|.:.|.:.|.:
: +---|-------+ :
:...:.|.:.|.:...:
:   : +---+ :   :
:...:...:...:...:
```

"невозможно".

Пример

№ 2

Входные данные:

```
3 3
4 5 0
3 7 5
0 3 6
```

Результат:

```
. . . . . .  . . . . . .  . . . . . .
: +---+ :   :  : +-------+ :  : +---+ :   :
:.|.:.|.:...:  :.|.:...:.|.:  :.|.:.|.:...:
: +---|---+ :  : | :   : | :  : +---+ :   :
:...:.|.:.|.:  :.|.:...:.|.:  :...:...:...:
:   : +---+ :  : +-------+ :  :   :   :   :
:...:...:...:  :...:...:...:  :...:...:...:
```

Требования к решению задачи

Вместе с программой представляются два файла данных (с расширениями .d1 и .d2), первый из которых (*.d1) представляет развязываемый узел, другой — не развязываемый. Программа должна сама уметь развязывать развязываемый узел и распознавать не развязываемый. Файлы данных будут использоваться как тестовые данные для других участников.

Оценка

В первую очередь оценивается правильность решения задачи: распознавание не развязываемых узлов и правильность развязывания. В интерфейсе главное — соответствие спецификации задачи: показ всех последовательных положений веревки со сдвигами на один шаг. Графические достоинства оцениваются в последнюю очередь.

10.71. Перемешанный алфавит. Рассмотрим простейший способ шифровки текста – с помощью перемешанного алфавита. Каждую букву шифруемого текста заменяют на другую букву (ее шифр). Между буквами и их шифрами имеется взаимно-однозначное соответствие, которое представлено в виде таблицы и называется перемешанным алфавитом. Даны русский текст, перемешанный алфавит. Зашифровать и расшифровать текст с помощью перемешанного алфавита.

10.72. Ключевое слово. Метод шифровки текста с помощью ключевого слова состоит в следующем. Пронумеруем буквы русского алфавита (А — 1, Б — 2, В — 3, ..., Я — 33), назовем эти номера «кодами» букв. Допустим, ключевое слово состоит из k букв ($k \geqslant 1$). Разобъем весь текст на последовательные группы по k букв в каждой группе; в последней группе букв может быть меньше, чем k. Пробелы, знаки препинания игнорируются. Первая группа k букв исходного текста шифруется следующим образом: к коду первой буквы группы прибавить код первой буквы ключевого слова, если полученное значение больше 33, то вычесть 33. Полученное значение является кодом буквы, которой нужно зашифровать первую букву группы. Тот же прием нужно применить ко второй букве группы и второй группе ключевого слова и так далее до конца группы. Указанный алгоритм применить к следующим группам по k букв в каждой. Дан русский текст, ключевое слово. Зашифровать и расшифровать текст с помощью описанного алгоритма.

10.73. Подстрока. Даны две строки символов $s1$ и $s2$. Найти наиболее длинное вхождение начальной части строки $s2$ в строку $s1$.

Пример

```
Вход: s1=aababcabce, s2=abcde
```

```
Выход: abc
```

Создайте диаграмму переходов и таблицу графа состояний (функции перехода) для конечного автомата, решающего указанную задачу.

10.74. Алгоритм Рабина. Решить задачу поиска подстроки в строке. При прямом побуквенном сравнении задача решается долго. Предлагается другой алгоритм: допустим есть подстрока $w_1w_2\dots w_n$, которую нужно обнаружить в строке $s_1s_2\dots s_m (n \leqslant m)$. Вычислим некоторую однозначную функцию на строках $w_1w_2\dots w_n$ и $s_1s_2\dots s_n$. Если значения функции не совпали, то эти строки не совпадают и нужно перейти к следующему шагу. На следующем шаге сравним значения функции на строках $w_1w_2\dots w_n$ и $s_2s_3\dots s_{n+1}$ и так далее до исчерпания строки $s_1s_2\dots s_m$; если на каком-то шаге значения функции совпали, то только тогда имеет смысл проводить побуквенное сравнение. Возникает вопрос, а чем лучше вычисление функции на строке побуквенного её сравнения с другой строкой? Ответ прост — каждая следующая за подстрокой $s_is_{i+1}\dots s_{i+n-1}$ подстрока может быть получена из предыдущей только удалением первого символа s_i и добавлением нового s_{i+n} . Поэтому, если подобрать 'хорошую' функцию, которую не нужно заново полностью перевычислять, а можно воспользоваться вычисленным на предыдущем шаге значением функции, то действительно алгоритм дает некоторое преимущество по сравнению с прямым перебором. Примером такой функции может служить сумма кодов входящих в строку символов, тогда вычисление нового её значения будет заключаться в вычитании кода первого символа и прибавлением кода нового символа. Реализовать предложенный алгоритм. Попробуйте найти другие функции и оцените выигрыш по времени.

10.75. Даны две строки симоволов $s1$ и $s2$. Найти наиболее длинную общую подстроку $s1$ и $s2$, то есть такую строку $s3$, чтобы она входила бы как подстрока в каждую из строк $s1$ и $s2$ и имела бы наибольшую длину из всех таких строк.

10.76. Сканер. Даны два входных файла. В первом файле (словарь) находятся различные цепочки букв и других символов (кроме пробелов), разделяемые пробелами. Назовём эти цепочки словами и будем считать, что слова в словаре пронумерованы натуральными числами. Второй файл содержит произвольный текст, в котором слова разделяются пробелами. Записать в выходной файл последовательность номеров слов текста в соответствии со словарём; если слово в словаре отсутствует, считать его номер равным нулю.

Пример. `Словарь: if then else a b m > < :=`
`Тест: if a > b then m := a else m := b ;`
`Выход: 1 4 7 5 2 6 9 4 3 6 9 5 0`

10.77. Простое форматирование текста. Задан текст. Абзацы в нём разделены пустой строкой. Слова отделаются друг от друга пробелами или переводами строк. Нужно получить этот же текст, но со следующими ограничениями:

- каждая строка не длиннее 50 символов;
- первая строка абзаца начинается ровно с трёх пробелов;
- все строки абзаца, кроме последней, одинаковой длины.
- в каждой строке должно быть максимально возможное количество слов;

Разрешается изменять количество пробелов между словами.

10.78. Красивое форматирование текста. Задан текст. Абзацы в нём разделены пустой строкой. Слова отделаются друг от друга пробелами или переводами строк. Нужно получить этот же текст, но со следующими ограничениями:

- каждая строка не длиннее 50 символов;
- первая строка абзаца начинается ровно с трёх пробелов;
- все строки абзаца, кроме последней, одинаковой длины;
- не должно быть больше 3 строк, в которых были бы пробелы расположенные один под другим (кроме абзацных оступов).
- в каждой строке должно быть максимально возможное количество слов;

Разрешается изменять количество пробелов между словами.

10.79. ✣ **Поиск подстроки.** Один из простых способов сжатия информации заключается в поиске одинаковых частей, идущих подряд. Вместо нескольких повторений можно записать их количество и всего одну часть.

Задача.

Найти в заданной сжатой строке заданную подстроку.

Техническое требование

Задана строка из латинских больших и маленьких букв. Числа являются разделителями частей строки и указанием количества повторений. Числа не более 6 знаков.

Длины строк не более 100 символов. В качестве ответа нужно выдать номер символа, с которого начинается искомая подстрока в несжатой строке. Если такой подстроки нет, то нужно выдать 0.

Пример

Строка: *7abcd20ab2003cd1cdEFGH*
Подстрока: *abcdc*
Ответ: 67

10.80. Постройте конечный автомат для фактов языка Prolog.

10.81. Приведите примеры современных систем программирования, в которых используется автоматное программирование?

Глава 11

Методы, основанные на рекурсии

11.1. 10 вложенных треугольников. Задан треугольник. Программа должна: нарисовать этот треугольник, выбрать на каждой стороне случайную точку, по полученным точкам нарисовать треугольник и так 10 раз.

11.2. Написать рекурсивную подпрограмму, вводящую с клавиатуры последовательность вещественных положительных чисел, заканчивающуюся нулем, которая возвращает их сумму.
Ограничение: массивы, циклы не использовать.

11.3. Написать рекурсивную подпрограмму, вводящую из файла последовательность вещественных чисел, которая выдает на экран вначале отрицательные, а затем – неотрицательные члены последовательности (в любом порядке).
Ограничение: массивы, циклы не использовать.

11.4. $\mathbf{F_n(2)}$. Вычислить значение функции

$$F_n(x) = \begin{cases} 100, & n = 0; \\ \sqrt{F_{n-1}(x)}, & n > 0. \end{cases}$$

при $x = 2$. n — натуральное, вводится.

11.5. Дан линейный вещественный массив. Реализовать рекурсивную функцию, находящую индекс первого из всех минимальных элементов массива.
Ограничение: циклы не использовать.

11.6. Салфетка Серпинского. На рисунке 11.1 показана салфетка Серпинского. Проанализируйте рисунок, напишите программу для построения «салфетки» произвольного порядка.

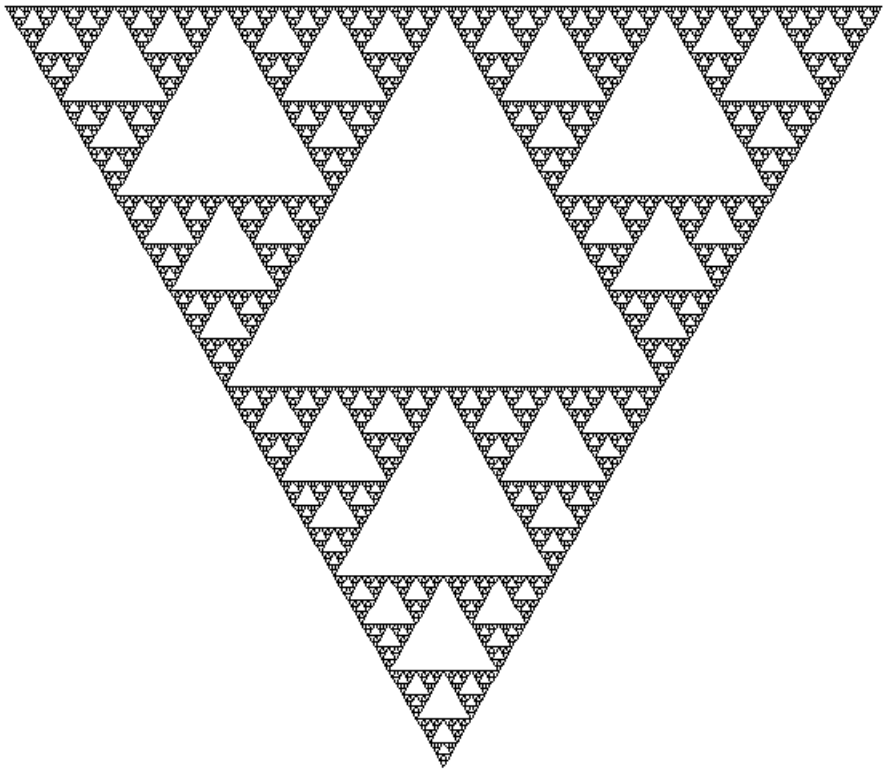


Рис. 11.1: Салфетка Серпинского

11.7. Фрактал «дракон» Хартера-Хейтуэя. На рисунке 11.2 показаны несколько шагов, начиная с нулевого, построения «дракона» Хартера-Хейтуэя. Проанализируйте рисунок, напишите программу для построения N-го шага «дракона».

11.8. Псевдодерево. Есть такой способ рисовать дерево на компьютере: рисуем ствол и в случайные моменты справа или слева пририсовываем дерево поменьше. Различные виды деревьев можно получить, если определить, что значит: «дерево поменьше» и сколько ветвей нужно.

11.9. Фрактал «Лист папоротника». На рисунке 11.3 показан «лист папоротника». Проанализируйте рисунок, напишите программу для построения «листа» N-ой «пышности».

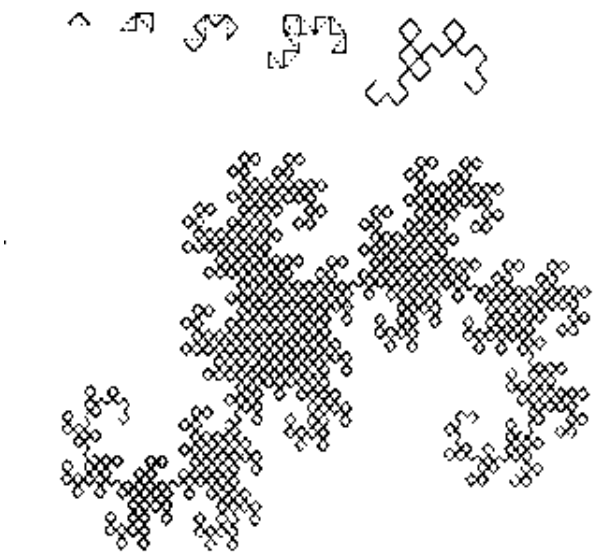


Рис. 11.2: Построение «дракона» Хартера-Хейтуэя

11.10. Озеро Мандельброта. Найдите информацию о том, как строится известный фрактал Мандельброта, и запрограммируйте это построение.

11.11. Жюлиа. Найдите информацию о том, как строится известный фрактал Жюлиа, и запрограммируйте это построение.

11.12. Фракталы. Есть ещё большое множество красивых и интересных фракталов. Запрограммируйте построение одного из них.

11.13. Алгоритм быстрой сортировки. Дан линейный вещественный массив $a_1, a_2, \ldots, a_n$. Алгоритм быстрой сортировки заключается в следующем: из массива выбирают один из элементов a_k (обычно – средний элемент), затем, двигаясь от начала массива слева направо, находят первый элемент, больший a_k; двигаясь от конца массива справа налево, находят первый элемент, не больший a_k; найденные элементы меняют местами. Движение слева и движения справа повторяют, чтобы найти следующую пару элементов для обмена; в последнюю очередь занимает свое окончательное место элемент a_k. В результате массив будет состоять из двух групп элементов: элементы слева от a_k не большие его, а справа – элементы, превосходящие a_k. Затем рекурсивно применяют алгоритм к левой и правой частям массива. Реализовать рекурсивный алгоритм быстрой сортировки массива.

Рис. 11.3: Лист папоротника

11.14. Реализовать рекурсивную версии алгоритма сортировки
а) выбором (см. задачу 9.22);
б) вставками (см. задачу 9.23);
в) обменом (см. задачу 9.24).

11.15. Вычислить значения математических функций из задачи 7.33 по их разложениям в ряд.
Ограничение: циклы не использовать.

11.16. Ханойские башни. На одном из трех алмазных шпилей надето 64 круглых золотых диска. Диски имеют разные радиусы и расположены на шпиле в порядке убывания радиусов от основания к вершине. Требуется перенести диски с первого шпиля на второй, используя при необходимости и третий шпиль. При этом неукоснительно должны соблюдаться следующие правила:

- за один раз можно перемещать только один диск;
- больший диск нельзя располагать на меньшем диске;
- снятый диск необходимо надеть на какой-либо шпиль перед тем, как будет снят другой диск.

Трудолюбивые буддийские монахи день и ночь переносят диски со шпиля на шпиль. Легенда утверждает, что когда монахи закончат свою работу, наступит конец света. Можно легко подсчитать, что для решения задачи с 64 дисками потребуется $2^{64} - 1$ перемещений (около 10^{20}).

Поэтому что касается конца света, то он произойдет по истечении пяти миллиардов веков, если считать, что один диск перемещается за одну секунду. Впрочем, и задачу, и легенду для неё придумал в 1883 году математик Э. Люка. Это дает нам право отложить заботы о конце света в сторону и перейти к решению следующей задачи.

Задача. Составить рекурсивную программу-функцию, которая бы решала поставленную выше задачу о Ханойских башнях при количестве дисков, равном n ($n = 1, 2, ...$).

Техническое требование

Пронумеруем шпили справа налево 1, 2, 3. В начале все диски расположены на шпиле номер 1. В результате работы программы на экран выводится последовательность пар чисел (переносов дисков). Первое число в паре — номер шпиля, с которого сняли диск, второе число — номер шпиля, куда диск надели.

Пример

```
n= 3
1 2
1 3
2 3
1 2
3 1
3 2
1 2
```

11.17. Функция Маккарти. Написать программу, которая будет вычислять для заданного n функцию

$$M(n) = \begin{cases} n - 10, & n > 100; \\ M(M(n + 11)), & n \leqslant 100. \end{cases}$$

Пример

```
n= 99
M(99)=91
```

11.18. Функция Кадью. Написать программу, которая будет вычислять для заданных целых неотрицательных x, y, z функцию

$$C(x, y, z) = \begin{cases} z, & x = y; \\ C(x, y + 1, (y + 1) \cdot z), & x \neq y. \end{cases}$$

Решите задачу только для случаев, когда значение функции не превышает $1\,000\,000\,000$.

Пример

```
x= 4
y= 0
z= 1
C(4, 0, 1)=24
```

11.19. Площадь в пикселах. На плоскости дан набор отрезков. Координаты концов отрезков — натуральные числа, не больше 10000. Может так случиться, что отрезки огораживают некоторую область, может быть и не одну. Кроме отрезков дана точка, координаты её тоже натуральные числа не больше 10000. Вычислить количество точек с натуральными координатами внутри области, в которой находится точка.

Если точка попадает на отрезок, в области точек нет. Если область не замкнута, ответ: «Бесконечность».

11.20. Количество областей. На плоскости дан набор отрезков. Координаты концов отрезков — натуральные числа не больше 10000. Может так случиться, что отрезки отгораживают некоторую область, может быть и не одну. Вычислить количество замкнутых областей.

11.21. Лошадью ходи! Ходом коня обойти всю шахматную доску, начиная с клетки $a1$, не проходя по каждой клетке более одного раза.

11.22. Лошадью ходи-2! Даны координаты двух клеток шахматной доски. Найти наиболее короткий из путей от первой до второй клетки, двигаясь ходом коня.

11.23. Расстановки ферзей. Найти все такие расстановки восьми ферзей на шахматной доске, чтобы ферзи не били друг друга.

11.24. Города. Дан список русских названий городов. Требуется построить максимально длинную цепочку названий городов таким образом, чтобы первая буква названия каждого следующего города совпадала с последней буквой названия предыдущего города. Если название города заканчивается на букву "ь", то она игнорируется. Город в цепочку может входить только один раз. Вместо названий городов можно взять любой список слов.

11.25. Дана вещественная квадратная матрица a порядка n. Найти определитель матрицы A, используя формулу разложения по k-ой строке:

$$\det A = \sum_{i=1}^{n} (-1)^{k+i} a_{ki} \cdot \det A_{ki}$$

где A_{ki} – матрица (минор), полученная из матрицы A выбрасыванием k-ой строки и i-го столбца.

Указание: реализовать рекурсивную функцию от двух параметров-множеств s_1 и s_2, вычисляющую по данной формуле определитель матрицы, полученной из исходной выбрасыванием строк, номера которых принадлежат s_1, и столбцов, номера которых принадлежат s_2.

11.26. Максимальное $R(x, y)$. Вычислить максимальное значение $R(x, y)$ для заданных целых x, y ($x < 10000, y < 50$).

$$R(x,y) = \begin{cases} x, & y < 1; \\ R(x+3, y-1), & x+y \text{ не делится на 5 и } y \geqslant 1; \\ R(x+4, y-1), & x+y \text{ не делится на 2 и } y \geqslant 1; \\ R(x, y-1), & x+y \text{ делится на 10 и } y \geqslant 1. \end{cases}$$

Внимание, «функция» R — неоднозначная (может давать разные значения на одинаковых аргументах).

Пример

```
X=3
Y=4
Максимальное значение R(3,4)=16
```

11.27. Игра «пятнадцать» На квадратном поле 4×4 случайным образом расставлены 15 квадратных фишек с номерами от 1 до 15. Передвигая их поочередно в свободную позицию, игрок пытается расставить фишки в порядке возрастания номеров, начиная с левого верхнего угла. Вынимать фишки с поля нельзя. Написать программу, которая по заданной исходной позиции фишек находит последовательность их перемещения до окончательной расстановки, либо сообщает, что никакими перемещениями решить игру нельзя. Желательно найти самую короткую последовательность.

Указание: можно вначале решить задачу «восемь», в которой размер квадрата 3×3.

11.28. Обход конём шахматной доски. Дана прямоугольная шахматная доска. Начав с любого поля шахматный конь должен пройти по всем клеткам. В каждую клетку можно вставать только один раз.

Напишите программу, которая решает эту задачу.

11.29. Расстановка коней. Дана прямоугольная шахматная доска. Расставить максимальное количество коней так, чтобы они не били друг друга.

Напишите программу, которая решает эту задачу.

11.30. Расстановка слонов. Дана прямоугольная шахматная доска. Расставить максимальное количество слонов так, чтобы они не били друг друга.

Напишите программу, которая решает эту задачу.

11.31. Расстановка произвольных фигур. Дана прямоугольная шахматная доска и некоторое множество шахматных фигур. Расставить фигуры так, чтобы они не били друг друга.

Напишите программу, которая решает эту задачу.

11.32. Мат в два хода. Дана прямоугольная шахматная доска на которой расставлены шахматные фигуры. Определить, можно ли поставить мат чёрным не более чем за два хода. Традиционно считаем, что начинают ходить белые.

Напишите программу, которая решает эту задачу.

11.33. Ладейный контроль. Дана прямоугольная шахматная доска. Расставить минимальное количество шахматных ладей так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.34. Слоновий контроль. Дана прямоугольная шахматная доска. Расставить минимальное количество шахматных слонов так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.35. Королевский контроль. Дана прямоугольная шахматная доска. Расставить минимальное количество шахматных королей так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.36. Ферзевый контроль. Дана прямоугольная шахматная доска. Расставить минимальное количество шахматных ферзей так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.37. Конский контроль. Дана прямоугольная шахматная доска.

Расставить минимальное количество шахматных коней так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.38. Не классическая доска. Решите задачи 11.28–11.37, где доска будет тором. Доска будет считаться тором, если дойдя до края на одной стороне доски можно перейти на противоположную сторону.

11.39. Задача о ранце. Имеется n предметов, известны их веса $v_1, v_2, \ldots, v_n$ и стоимости $p_1, p_2, \ldots, p_n$. Требуется выбрать из них такие, чтобы их суммарный вес не превосходил бы V, а суммарная стоимость была бы максимальной.

11.40. На две группы. Даны веса n предметов $v_1, v_2, \ldots, v_n$. Нужно все предметы разделить на две группы, чтобы суммарные веса групп различались бы минимально.

11.41. Игра Ним. В игре участвуют три игрока. Даны три кучки камешков. Ход игрока состоит в том, что он забирает из какой-то одной (произвольной) кучки положительное число камешков. Игроки ходят по очереди. Выигрывает тот, кто заберет последний камешек. Написать программу, моделирующую игру Ним, в которой участвуют два человека, а за третьего играет программа.

11.42. Король в лабиринте. Дана прямоугольная шахматная доска. Некоторые клетки доски заняты нейтральными фигурами, назовём такие фигуры *стена*. Стена не ходит и не атакует, но через неё не могут ни ходить, ни атаковать другие фигуры, даже конь.

Расставить на заданной доске со стенами минимальное количество королей так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.43. Ферзь в лабиринте. Расставить на заданной доске со стенами (определение стены в задаче 11.42) минимальное количество ферзей так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.44. Конь в лабиринте. Расставить на заданной доске со стенами (определение стены в задаче 11.42) минимальное количество коней так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.45. Слон в лабиринте. Расставить на заданной доске со стенами (определение стены в задаче 11.42) минимальное количество слонов так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.46. Ладья в лабиринте. Расставить на заданной доске со стенами (определение стены в задаче 11.42) минимальное количество ладей так, чтобы нельзя было поставить ещё одну фигуру, и она бы не оказалась под боем.

Напишите программу, которая решает эту задачу.

11.47. Простые цепи. Дан ориентированный граф. Найти все простые цепи в графе из вершины a в вершину b. *Простой цепью* называется последовательность отличных друг от друга дуг, начало каждой последующей дуги совпадает с концом предыдущей и вершины дуг не повторяются.

11.48. Циклы графа. Дан ориентированный граф. Найти самый большой цикл в графе, то есть цикл, содержащий наибольшее количество дуг. *Циклом* в графе называется замкнутая простая цепь, то есть цепь, начало первой дуги в которой совпадает с концом последней.

11.49. Деревья. Ориентированный граф, не имеющий циклов, называется *деревом*. Проверить, является ли данный орграф деревом.

11.50. Склеивание треугольников. *Треугольником* в неориентированном графе называется тройка попарно соединенных дугами вершин. Склеиванием треугольника называется операция замены вершин треугольника новой вершиной с сохранением всех связей с остальными вершинами графа. Дан неориентированный граф. Склеить все треугольники графа.

11.51. Гамильтонов граф. Граф называется *гамильтоновым*, если существует цикл, проходящий через все вершины графа. Дан орграф. Проверить, является ли он гамильтоновым.

11.52. Эйлеров граф. Граф называется *эйлеровым*, если существует замкнутая цепь, содержащая все дуги графа. Дан орграф. Проверить, является ли он эйлеровым.

11.53. Проверка на связность. Граф называется *связным*, если любые две его вершины связаны простой цепью. Дан орграф. Определить, связен ли он.

11.54. Компоненты связности. *Компонентой связности* графа назовем такое подмножество K его вершин, что любые две вершины K связаны простой цепью, но не существует цепи от любой вершины K до любой вершины, не принадлежащей K. Таким образом, граф может быть разбит на непересекающиеся компоненты связности. Дан орграф. Найти все его компоненты связности.

11.55. Латинские квадраты. *Латинским квадратом* порядка n называют квадратную матрицу m размером $n \times n$, элементы которой принадлежат множеству $M = \{1, 2, \ldots, n\}$, причем каждое число из M встречается ровно один раз в каждой строке и в каждом столбце m.

Задача. Написать программу, которая при заданном натуральном n формирует и выводит все латинские квадраты размером $n \times n$ с первой строкой и первым столбцом вида: $1, 2, \ldots, n$.

Пример

```
n= 4
 1  2  3  4
 2  1  4  3
 3  4  1  2
 4  3  2  1

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

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

 1  2  3  4
 2  4  1  3
 3  1  4  2
 4  3  2  1
```

11.56. Количество латинских квадратов. Написать программу, которая при заданном натуральном n подсчитывает количество латин-

ских квадратов (определение см. в задаче 11.55) размером $n \times n$ с первой строкой вида: $1, 2, \ldots, n$.

Пример

```
n= 4
24
```

11.57. Магические квадраты. Дана целочисленная квадратная матрица A порядка n. Проверить, можно ли в этой матрице переставить числа так, чтобы она стала магическим квадратом, то есть такой, в которой суммы элементов в каждой строке и в каждом столбце совпадают.

11.58. Гипотеза Эйлера. Среди многочисленных гипотез, выдвинутых Л. Эйлером и, как правило, впоследствии оказавшихся верными, имеется и такое утверждение, сформулированное им в середине XVIII века:

Для любого натурального показателя $k > 3$ и натуральных $n_1, n_2, \ldots, n_p$ уравнение

$$n_1^k + n_2^k + \ldots + n_p^k = n_{p+1}^k \ (2 \leqslant p < k)$$

не имеет решений.

Если гипотеза Эйлера справедлива, то из неё, как частный случай при $p = 2$, следует справедливость Великой Теоремы Ферма.

Задача. Написать программу, которая найдёт опровержение гипотезы Эйлера.

11.59. Возвратная рекурсия. Что такое возвратная рекурсия?

11.60. Глубина рекурсии. Что такое глубина рекурсии?

11.61. Взаимная рекурсия. Что такое взаимная рекурсия?

11.62. Рекурсивное зацикливание. Что такое рекурсивное зацикливание? Какие есть способы избежать рекурсивного зацикливания?

11.63. Уравнение a+2b+3c+4d=N. Рассмотрим диофантово (только целые решения) уравнение: $a + 2b + 3c + 4d = N$, где a, b, c и d — некоторые положительные целые, которые нужно найти.

Написать программу, которая по заданному N будет находить числа a, b, c и d.

11.64. Лабиринт. Задан лабиринт. Можно ли в нём пройти из нижнего левого угла в верхний правый?

Техническое требование

Лабиринт задаётся восьмизначным шестнадцатеричным числом, где двоичное разложение цифры числа указывает расположение стен и проходов. 0 — проход, 1 — стена. Двоичные числа дополняются слева нулями до четырёх знаков. Полученные числа задают лабиринт, если их цифры расположить сверху вниз, а сами числа слева направо. В результате получится прямоугольная таблица размерами 4×8. Ходить можно только по проходам по вертикали и горизонтали. Считается, что вокруг стоят стены. Если в нижнем левом углу или верхнем правом углу находится стена, то пройти нельзя.

Пример

```
Вход: 0A0D0A60
Выход: Да
```

11.65. Поиск пути. Имеется N населенных пунктов. Некоторые пары пунктов соединены дорогами. Требуется определить, можно ли из пункта с номером n_1 попасть в пункт с номером n_2 и, если можно, то указать какой-нибудь путь. Система дорог задается в виде квадратной симметричной относительно главной диагонали матрицы, состоящей из нулей и единиц: если элемент a_{ij} равен 1, значит пункты с номерами i и j напрямую соединены дорогой, а если равен 0 — то не соединены (матрица смежности). Путь нужно указать в виде последовательности номеров пунктов $i_1, i_2, \ldots, i_k$, где $i_1 = n_1, i_k = n_2$.

11.66. Короткий путь. Имеется N населенных пунктов. Некоторые пары пунктов соединены дорогами определенной длины. Требуется определить, можно ли из пункта с номером n_1 попасть в пункт с номером n_2 и, если можно, то каков наименьший путь. Система дорог задается в виде квадратной симметричной относительно главной диагонали матрицы, состоящей из вещественных неотрицательных чисел: если элемент $a_{ij} = L$ больше 0, значит пункты с номерами i и j напрямую соединены дорогой длины L, а если равен 0 — то не соединены. Путь нужно указать в виде последовательности номеров пунктов $i_1, i_2, \ldots, i_k$, где $i_1 = n_1, i_k = n_2$.

11.67. Мы пойдём другим путем. Система дорог организована как в задаче 11.66. Требуется найти второй по длине путь из n_1 в n_2.

11.68. Дорожная разметка. Система городских улиц состоит из перекрестков и соединящих их дорог. Количество перекрестков — n, они пронумерованы от 1 до n, каждая дорога обозначается номерами двух перекрестков i и j, которые она соединяет, a_{ij} — длина дороги. Разметочная машина наносит на дорожное полотно разметку, двигаясь по дорогам

от перекрестка к перекрестку, при этом она может идти в двух режимах: с покраской и без покраски. Требуется определить такой маршрут движения машины, чтобы разметка была нанесена на все дороги и при этом пройденный ею путь был бы минимальным. Путь машины начинается от перекрестка с номером k. Входные данные в файле задаются в следующей последовательности:

```
<Количество перекрестков n>
<Начальный перекресток k>
<Количество дорог m>
<Номер перекрестка-начало><Номер перекрестка-конец><Длина дороги>
<Номер перекрестка-начало><Номер перекрестка-конец><Длина дороги>
. . .
```

Выходные данные определяют маршрут движения машины и режим работы на каждой дороге; выводятся построчно по три числа в каждой строке: номер перекрестка-начало дороги, номер перекрестка-конец дороги, режим движения (0 — разметка не наносится, 1 — разметка наносится).

11.69. **Исправление лабиринта.** При строительстве лабиринтов в парках развлечений случайным возведением стен оказалось, что некоторые лабиринты могут не иметь пути от входа к выходу. Но этот способ отлажен и дёшев. В связи с этим было принято решение строить лабиринты в два шага: первый — тот же, что и раньше; второй — убираем некоторые стены, чтобы обеспечить наличие пути.

Ваша задача — разработать компьютерную программу для поддержки второго шага. В лабиринте созданном на первом шаге указать, какие стены следует убрать. Очевидно, что нужно убирать минимальное количество стен.

Техническое требование

Схема лабиринта задаётся в файле maze.txt. Все лабиринты имеют одинаковые размеры 10×10. Вход всегда в левом верхнем углу сверху, выход всегда в правом нижнем углу снизу. По контуру всегда построена стена с одним входом и одним выходом. Пробелы обозначают проход, решётки (#) — стену. Итого в файле записано 144 символа.

Время работы программы ограничено.

Пример лабиринта

```
# ##########
############
# # # # # ##
## # # # # #
# # # # # ##
## # # # # #
# # # # # ##
## # # # # #
# # # # # ##
## # # # # #
############
########## #
```

Задача: написать программу, которая по заданному в файле *maze.txt* лабиринту выдает на экран одно из возможных решений.

Спецификация выхода

На экране в текстовом виде должен быть отображен исходный лабиринт на чёрном фоне. Стены раскрашиваются так: убираемые — красным цветом, остальные — белым. Тут же нужно отобразить один из путей от входа к выходу при данном удалении стен. Для этого клетки пути должны иметь синий фон. Через строчку после рисунка лабиринта нужно вывести количество убираемых стен.

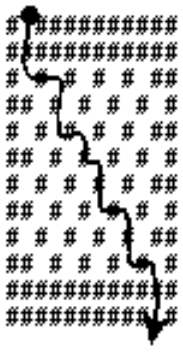


Рис. 11.4: Снести стены на этом пути

Пример

Для лабиринта из примера показан на рисунке 11.4.

Снести стен: 10

11.70. Ледяной лабиринт. Если мы пытаемся передвигаться в очень скользком лабиринте, то единственный способ передвижения от одной стенки до другой — проскользить через весь коридор. При таком способе свернуть в некоторые из проходов не удаётся.

Задача. Для заданного лабиринта, начального и конечного положений определить, как добраться из начального положения в конечное.

11.71. Длинный лабиринт. Лабиринт задается двоичной матрицей из N строк и 64 столбцов. Цифра 1 задает стенку, цифра 0 — проход. Требуется пройти из первой строки лабиринта до строки с наибольшим номером, двигаясь в любых направлениях по горизонтали и вертикали по нулевым элементам матрицы. Позиции начала пути в первой строке и конца пути в последней строке безразличны.

Исходные данные задаются в виде двоичного (не текстового!) файла *input.bin*, каждая строка матрицы занимает 8 байтов: старший бит первого байта — первый элемент строки, ..., младший бит восьмого байта — последний элемент строки.

Ограничение: значение N не превышает два миллиарда.

Ответ в виде номера строки, до которой удается дойти, в десятичном виде выдается на экран (строки нумеруются с 1). Время работы программы не должно превышать времени однократного копирования файла более, чем на 10 секунд.

Пример

Если *input.bin* представляет собой следующую двоичную матрицу:

```
0111111111111111111111111111111011111111111111111111111111111111
0100010100010001010001000100010001000100010001000100010001000100
0001000101000101000100010001000100010001000100010001000100010001
1111111101111111111111111111111111111111111111111111111111111100
0000000000000000000000000000000000000000000000000000000000000111
```

то программа должна выдать число 4.

11.72. Дана символьная строка. Реализовать логическую рекурсивную функцию, проверяющую, является ли строка палиндромом.

Ограничение: циклы не использовать.

11.73. Дана символьная строка. Реализовать рекурсивную функцию, печатающую символы строки в обратном порядке.

Ограничение: циклы не использовать.

11.74. Даны две символьные строки s_1, s_2. Реализовать рекурсивную функцию, вычисляющую количество вхождений строки s_2 в строку s_1.

Ограничение: циклы не использовать.

11.75. Дано целое неотрицательное число n. Реализовать рекурсивную функцию, вычисляющую количество и сумму цифр в десятичной записи n.
Ограничение: циклы не использовать.

11.76. ✣ **Сравнение с образцом.** Образец имеет вид:

```
образец ::= <PT>
  <PT> ::= <A><PT> | <A>
  <A>  ::= '['<PT>']' | <ST>
  <ST> ::= <символы>
```

Часть образца, заключенная в квадратные скобки показывает, что эту часть можно исключать. Таким образом, образец описывает множество строк. Образец всегда правильный. Например:

образец: [a[b]]c

описывает строки: c, ac, abc

Слово подходит под образец, если оно совпадает с одним из слов, описываемых образцом. Задача написать программу, которая по заданным образцу и слову печатает «НЕТ», если слово подходит под образец, и «ДА», если не подходит.

Пример

```
образец: [a[b]]c
слово: ab
ответ: ДА
```

11.77. Строковый калькулятор. Вычислить значение арифметического выражения записанного с помощью операций '+', '−' и круглых скобок. Числа целые не длинее 8 знаков.

11.78. Логический калькулятор. Вычислить значение логического выражения записанного с помощью операций 'and', 'or', 'not' и круглых скобок. T — истина, F — ложь.

11.79. Символьный калькулятор. Вычислить значение строкового выражения записанного с помощью операций '+', '∗' и круглых скобок. Строки не длиннее 10 символов и состоят только из маленьких латинских букв.

Операция + обозначает конкатенацию строк, $A+B+C = (A+B)+C$. Например, $adbc+123 = abcd123$. Результатом операции $A*B$ будет строка,

где после каждого символа A записана B; если A или B — пустые строки, то результат будет также пустой строкой, $A*B*C = (A*B)*C$. Например, $ab * cd = acdbcd$. Операции выполняются в порядке их следования слева направо.

11.80. Порядок присваиваний. В файле *prog.xyz* записана программа. В записи программы использованы только операторы присваивания (:=). Присваиваться будут только арифметические выражения.

Определить как нужно изменить порядок операций присваивания, чтобы значение переменной Z было минимальным.

Считаем, что вначале значение всех переменных равно 0.

Пример

Исходная программа (*prog.xyz*):

```
A:=1;
B:=2;
Z:=A+B;
A:=B-A;
```

Результат:

```
A:=1;
A:=B-A;
Z:=A+B;
B:=2;
```

11.81. Значение программы. В файле *prog.xyz* записана программа. Для записи программы использованы только операторы **if** языка программирования Pascal (включая **begin** и **end**) и присваивания вида `X:='<строка>'`.

Техническое требование

В условиях используется только переменная `X`, операции =, <> и строки. <строка> — последовательность любых символов кроме "´" не длиннее 100.

Задача

Для заданного начального значения `X` вычислить её значение после завершения программы.

Пример

При X='A' в результате работы программы:

```
if X<>'A' then X:='B'
else X:='C';
```

Результат: X='C';

11.82. Почти по Маркову. Даны n правил преобразования строк в виде пар $(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_n, \beta_n)$, где α_i и β_i — некоторые строки, α_i— непустая строка. Правило (α_i, β_i) называется применимым к строке x, если α_i является подстрокой строки x. Применением применимого правила (α_i, β_i) к строке x называется замена первого вхождения подстроки α_i в строку x на подстроку β_i. Например, при применении к строке 'abcabcd' правила ('bc','klm') получается строка 'aklmabcd'.

Даны две строки s и f, правила преобразования. Требуется указать наиболее короткую последовательность применения правил к строке s для того, чтобы получить строку f или сообщить, что этого сделать невозможно. Правила указываются своими номерами.

Пример. (Здесь кавычки не являются частью строк.)

```
Строка s='МУХА'
Строка f='СЛОН'
Правила:
('МУХ','СЛО'),('А','НН'),('Н',''),('МУХ','СЛ'), ('ХА','ХОН')
Ответ: номера примененных правил - 5 4
```

11.83. В каждой ячейке прямоугольной матрицы написаны русские буквы. Последовательность букв может образовать слово, если ячейки с буквами располагаются друг за другом; при движении от ячейки можно переходить к другой ячейке ниже, выше, слева или справа от неё. Ячейка с буквой может быть прочитана только один раз и принадлежать только одному слову.

Дана матрица букв и список слов. Каждое слово можно прочитать в матрице только один раз. Какое максимальное число букв матрицы может быть прочитано в виде данных слов?

Пример.

```
Матрица:
МЫЛМ
АМАУ
МАРЫ
Слова: МАМА МЫЛА РАМУ
Ответ:8.
```

11.84. В чём преимущество обратной польской записи перед обычной инфиксной?

11.85. Без рекурсии. Приведите пример рекурсивной программы, которую нельзя записать без рекурсии.

11.86. Приведите примеры современных систем программирования, в

которых используются рекурсивные методы?

11.87. В последовательность $x_1?x_2?x_3?...?x_n$ вместо знаков '?' поставить '+' либо '−' так, чтобы полученное выражение равнялось 0. x_i — натуральные десятичные числа, их длина не более трёх цифр. Количество чисел не более 20.

Числа x_i вводятся до тех пор, пока не введут 0.

Вывести получившееся выражение или сообщение, что решений нет.

Пример

1
2
3
4
0
1-2-3+4

11.88. Подгонка под ответ. Между некоторыми цифрами числа $x_1x_2x_3...x_n$ вставить знаки '+' либо '−' так, чтобы полученное выражение равнялось N. x_i — десятичные цифры, длина числа X не более 20, $0 \leqslant N \leqslant 1\,000\,000\,000$.

Числа X и N вводятся.

Вывести получившееся выражение или сообщение, что решений нет.

Пример

X=*123456789*
N=*171*
$12 + 34 + 56 + 78 - 9 = 171$

11.89. В арифметическом выражении расставить скобки так, чтобы оно равнялось N. Выражение содержит только десятичные положительные числа не больше 10000, операции '+' и '−'. Длина выражения не более 100 символов. $0 \leqslant N \leqslant 1\,000\,000\,000$. Выражение и число N вводятся.

Вывести получившееся выражение или сообщение, что решений нет.

Пример

$1 - 2 - 3 + 4 + 5$
11
$1 - (2 - (3 + 4)) + 5 = 11$

Глава 12

Объектно-ориентированный подход

12.1. Сущности, свойства, связи. Рассмотрим некоторую предметную область. Требуется её смоделировать для дальнейшего этапа построения базы данных или использования модели в программной системе. Опишите существенные для целей моделирования абстрактные объекты (сущности) и их свойства. Приведите примеры несущественных объектов или свойств, от которых необходимо абстрагироваться. Опишите устойчивые связи между различными сущностями, типы этих связей. Опишите, какого рода события могут происходить с объектами и какие действия над ними можно производить. В качестве примера можно рассмотреть следующие предметные области:

а) библиотека;
б) факультет;
в) расчетно-кассовый центр банка;
г) графический интерфейс пользователя программы;
д) стратегическая игра;
е) симулятор автогонок.

12.2. Приведите примеры отношений между сущностями следующих типов:

а) целое-часть (когда сущность одного типа может являться частью, элементом сущности другого типа);
б) род-вид (когда сущность одного типа может являться частным случаем сущности более общего типа);
в) ассоциация (когда сущности связаны друг с другом каким-либо взаимоотношением).

12.3. Определите мощность ассоциации («один-к-одному», «один-ко-

многим», «многие-ко многим»), выражающей связь между сущностями:
а) каждый студент учится в одной учебной группе;
б) каждый студент изучает учебные предметы;
в) каждый преподаватель может работать на нескольких факультетах;
г) преподаватель может являться деканом только одного факультета;
д) у каждого студента есть научный руководитель.

12.4. Что такое наследование?

12.5. Что такое полиморфизм?

12.6. Что такое инкапсуляция?

12.7. Каковы недостатки множественного наследования?

12.8. В чём польза переопределяемых (виртуальных) методов?

12.9. Для чего нужны конструкторы и деструкторы?

12.10. Много конструкторов. В некоторых объектных языках и системах программирования (например, Object Pascal) существует возможность явного определения нескольких конструкторов (или деструкторов) класса с разными именами, в то время как в других системах (C/C++) конструктор может иметь только одно имя, правда, их также можно определить несколько, применяя механизм перегрузки (переопределения, статический полиморфизм). Равноценны ли эти два способа определения нескольких конструкторов? Перечислите и обоснуйте недостатки и преимущества каждого из них.

12.11. Дружественность. Некоторые объектно-ориентированные языки предоставляют возможность определения так называемых дружественных (friend) классов и подпрограмм. Дружественным данному классу называется класс (или подпрограмма), которому (которой) предоставлен полный доступ до членов данного класса, в том числе скрытых. Существуют ли веские основания наличия такой возможности в языке и, если существуют, приведите их. Можно ли вообще без них обойтись? Как наличие (или отсутствие) механизма дружественности согласуется с базовыми концепциями таких систем, как Delphi, C/C++, Java и других.

12.12. Сформулируйте ответы на следующие вопросы:
а) какие члены класса предпочтительно создавать доступными?
б) какие члены класса предпочтительно создавать скрытыми?
в) каковы основания члены класса делать защищенными?

г) какие методы класса определяют виртуальными?
д) в чем заключается назначение статических членов класса?
е) для чего какой-либо класс определяется как абстрактный?
ж) в каких случаях обосновано использование шаблонов класса?

12.13. Структура или класс? В объектно-ориентированных языках, подобных C/C++, структуры, как и классы, также могут инкапсулировать поля и методы; более того, они могут иметь конструкторы и деструкторы и обладать другими свойствами классов. Определите, какими конкретно свойствами и возможностями классов обладают структуры в вашей системе программирования. Сформулируйте правило, в каких случаях реализация сущности обоснована в виде структуры, а в каких — в виде класса.

12.14. Комплексные числа. Опишите класс, реализующий тип данных «комплексные числа». Реализуйте операции с объектами этого класса: ввод, вывод, сложение, вычитание, умножение, деление, вычисление модуля, представление в тригонометрической форме. Рекомендуется использовать перегрузку (переопределение) существующих в языке или определенных в других классах операций, таких, как '+', '−', '∗', извлечение квадратного корня и других. Напишите программу, демонстрирующую работу с созданным классом.

12.15. Рациональные числа. Опишите класс, реализующий тип данных «рациональные числа» , то есть обыкновенные дроби с выделенными числителем и знаменателем. Реализуйте операции с объектами этого класса: ввод, вывод, сложение, вычитание, умножение, деление, сравнение. Рекомендуется использовать перегрузку (переопределение) существующих в языке или определенных в других классах операций, таких, как '+', '−', '∗' и других. Напишите программу, демонстрирующую работу с созданным классом.

12.16. Смешанные числа. Опишите класс, реализующий тип данных «смешанные рациональные числа» , то есть обыкновенные дроби с выделенными числителем, знаменателем и целой частью, причем числитель и знаменатель образуют правильную несократимую обыкновенную дробь. Указанный класс сделайте потомком класса рациональных чисел (см. задачу 12.15) Переопределите необходимые операции с объектами этого класса. Напишите программу, демонстрирующую работу с созданным классом.

12.17. Длинные числа. Опишите класс, реализующий тип данных

«длинные вещественные числа» , то есть числа с большим количеством десятичных цифр (например, до 256 знаков). Реализуйте операции с объектами этого класса: ввод, вывод, сложение, вычитание, умножение, деление, извлечение корня, сравнение. Рекомендуется использовать перегрузку (переопределение) существующих в языке или определенных в других классах операций, таких, как '+', '−', '∗' и других. Напишите программу, демонстрирующую работу с созданным классом.

12.18. Стек. Опишите класс, реализующий стек, элементами которого могут являться целые числа. Реализуйте операции с объектами этого класса: включение, исключение элемента, проверка на пустоту, заполненность. Напишите программу, демонстрирующую работу с созданным классом.

12.19. Дек. Опишите класс, реализующий дек (двустороннюю очередь, включение и исключение в которой допускаются на обоих концах), элементами которого могут являться целые числа. Реализуйте операции с объектами этого класса: включение, исключение элемента, проверка на пустоту, заполненность. Напишите программу, демонстрирующую работу с созданным классом.

12.20. Очередь. Опишите класс, реализующий очередь, элементами которого могут являться целые числа. Реализуйте операции с объектами этого класса: включение, исключение элемента, проверка на пустоту, заполненность. Напишите программу, демонстрирующую работу с созданным классом.
Замечание: так как очередь является частным случаем дека (см. задачу 12.19), то этот класс можно сдлеать производным от класса, реализующего дек.

12.21. Бинарное дерево. Опишите класс, реализующий бинарное дерево, элементами которого могут являться целые числа. Реализуйте операции с объектами этого класса: включение, исключение элемента из дерева, поиск элемента в дереве, ввод, вывод, последовательный доступ к элементам и другие. Напишите программу, демонстрирующую работу с созданным классом.

12.22. Бинарное дерево поиска. Опишите класс, реализующий бинарное дерево поиска, элементами которого могут являться целые числа. Бинарное дерево поиска— это бинарное дерево, корень которого больше всех вершин его левого поддерева и меньше всех вершин правого поддерева (если эти поддеревья есть), а поддеревья являются также бинар-

ными деревьями поиска. Реализуйте операции с объектами этого класса: включение, исключение элемента из дерева, поиск элемента в дереве, ввод, вывод, последовательный доступ к элементам и другие. Напишите программу, демонстрирующую работу с созданным классом.
Замечание: так как бинарное дерево поиска является частным случаем бинарного дерева (см. задачу 12.21), то этот класс можно сделать производным от класса, реализующего бинарное дерево.

12.23. Многочлены. Опишите класс, реализующий многочлены, которые задаются последовательностью коэффициентов и показателей степени. Реализуйте операции с объектами этого класса: сложение, вычитание, умножение, деление с остатком, ввод и вывод, нахождние значения многочлена для заданного аргумента, взятие производной, композиция многочленов (то есть подстановка в качестве аргумента многочлена другого многочлена). Напишите программу, демонстрирующую работу с созданным классом.

12.24. Линейный динамический массив. Опишите класс, объекты которого являются линейными динамическими вещественными массивами с изменяющимися границами. Реализуйте операции с объектами этого класса: возможность задания границ, доступ к отдельным элементам массива, сортировка, поиск элемента. Напишите программу, демонстрирующую работу с созданным классом.

12.25. Векторы. Опишите класс, объекты которого являются векторами (одномерными вещественными массивами). Реализуйте операции с объектами этого класса: сложение, вычитание, скалярное произведение и умножение на скаляр, ввод, вывод, доступ к отдельным координатам вектора. Напишите программу, демонстрирующую работу с созданным классом.
Замечание: этот класс можно сделать производным от класса, реализующего линейный динамический массив (см. задачу 12.24).

12.26. Матрицы. Опишите класс, реализующий вещественные матрицы. Определите операции с объектами этого класса: задание размера матрицы, сложение, вычитание матриц одинаковых размеров, нахождение произведения матриц соответствующих размеров, умножение на скаляр, ввода и вывода. Напишите программу, демонстрирующую работу с созданным классом.

12.27. Квадратные матрицы. Опишите класс, реализующий квадратные вещественные матрицы, сделав его порожденным от класса ве-

щественных матриц (см. задачу 12.26). Помимо имеющихся операций, определите еще нахождение обратной и транспонированной матрицы, определителя, нормы. Напишите программу, демонстрирующую работу с созданным классом.

12.28. Множества. Опишите класс, объектами которго являются множества целых чисел. Определите операции с множеством: конструирование множества из элементов, нахождение объединения, пересечения, разности, симметрической разности множеств, проверку принадлежности элемента множеству. Напишите программу, демонстрирующую работу с созданным классом.

12.29. Конечный автомат. Опишите класс, реализующий конечные автоматы. Определите операции для конечного автомата: задание множества состояний, функции перехода, входного алфавита, начального и заключительных состояний; задание входной строки, моделирование (интерпретация) работы автомата с выводом последовательности переходов. Напишите программу, демонстрирующую работу с созданным классом.

12.30. Файлы. Опишите класс, реализующий целочисленный файл внешней памяти. Определите операции для файла: открытие, закрытие, добавление, очистка, чтение очередной компоненты файла; соединение двух файлов (дозапись после первого файла содержимого второго файла), переименование, удаление и, возможно, другие. Напишите программу, демонстрирующую работу с созданным классом.

12.31. Требуется определить функцию max, возвращающую наибольшее значение двух или трех своих числовых параметров (либо целочисленных, либо вещественных). Приведите все возможные варианты решения этой задачи с использованием различных возможностей вашей системы/языка программирования с привлечением средств ООП или других.

12.32. Геометрические фигуры. Определите систему классов, представляющих геометрические фигуры на координатной плоскости. Для этого определите абстактный класс «геометрическая фигура», а от него породите производные конкретные классы («треугольник», «прямоугольник», «окружность»). В абстрактном классе опишите общие для всей системы фигур свойства и действия, а в конкретных — определите их для конкретных фигур. В классах дожны быть предусмотрены следующие операции: определение и изменение координат и размеров фигуры, вращение, изменение цвета, заполнение фигуры, вывод на экран и, возможно, другие. Напишите программу, демонстрирующую работу с

созданной системой классов. Например, можно выводить движущиеся в различных направлениях фигуры, изменяя параметры их движения при взаимных столкновениях.

12.33. Избыточность. Под ортогональностью в языке программирования будем понимать отсутствие практической возможности выражения одних программных конструкций (возможностей, средств языка) через другие. Большое количество выразимых друг через друга конструкций ведет к избыточности средств языка программирования. Исследуйте весь набор объектно-ориентированных средств вашей программной системы (самого языка, а не библиотек!). Сформулируйте и обоснуйте минимальный набор взаимно ортогональных друг другу средств, которыми можно ограничиться при программировании в системе без потери мощности, гибкости и читабельности. Как этот набор согласуется с парадигмой языка?

12.34. Укажите и обснуйте, какие из следующих механизмов объектно-ориентированного программирования:

а) инкапсуляция;
б) наследование;
в) множественное наследование;
г) абстракция;
д) сокрытие членов;
е) статические члены;
ж) полиморфизм;
з) статическое и динамическое связывание;
и) шаблоны классов;
к) абстрактные классы, виртуальные методы

каким-либо образом присутствовали (пусть даже в зачаточном состоянии) в технологиях более ранних этапов развития программирования, а именно:

а) программирование в машинных кодах;
б) структурное программирование;
в) процедурное программирование;
г) модульное программирование.

12.35. Приведите пример программистской задачи, когда для её решения применение множественного наследования не просто обосновано, но является оптимальным из всех объектных решений.

12.36. Система линейных уравнений. Опишите класс, реализующий алгоритм решения системы из n линейных уравнений с n неизвест-

ными. Определите методы этого класса: задание матрицы системы, вектора правых частей, вывод решаемой системы, решение системы (например, методом Гаусса), вывод решения. Класс, решающий поставленную задачу, сделайте производным от двух классов — класса квадратных вещественных матриц (см. задачу 12.27) и класса вещественных векторов (см. задачу 12.25), использовав для решения систем операции из базовых классов. Напишите программу, демонстрирующую работу с созданным классом.

12.37. Автомат с магазинной памятью. Опишите класс, реализующий автомат с магазинной памятью. Указанный класс сделайте производным от двух классов — класса конечных автоматов (см. задачу 12.29) и класса, реализущего стек (см. задачу 12.18), использовав при моделировании действий автомата операции из базовых классов. Помимо унаследованных операций, реализуйте необходимые дополнительные. Напишите программу, демонстрирующую работу с созданным классом.

12.38. Алгорифм А. А. Маркова. Опишите класс, реализующий алгорифм А. А. Маркова. Реализуйте операции для объектов класса (каждый объект является алгорифмом): задание последовательности правил преобразования, интерпретация (исполнение) алгорифма для входной строки; композиция алгорифмов (то есть возможность определения алгорифма из двух других, когда выходная строка первого алгорифма является входом для второго). Напишите программу, демонстрирующую работу с созданным классом.

12.39. Произвольный стек. Опишите шаблон класса, реализующий стек, элементами которого могут являться произвольные значения. Реализуйте операции с объектами этого класса, как в задаче 12.18. Напишите программу, демонстрирующую работу с созданным шаблоном класса, в которой задаются различные типы элементов стека, например целые числа, символы, комплексные числа (задача 12.14), длинные числа (задача 12.17), смешанные числа (задача 12.16).

12.40. Произвольная очередь. Опишите шаблон класса, реализующий очередь, элементами которого могут являться произвольные значения. Реализуйте операции с объектами этого класса, как в задаче 12.20. Напишите программу, демонстрирующую работу с созданным шаблоном класса, в которой задаются различные типы элементов очереди, например целые числа, символы, комплексные числа (задача 12.14), длинные числа (задача 12.17), смешанные числа (задача 12.16).

12.41. Произвольное бинарное дерево. Опишите шаблон класса, реализующий бинарное дерево, элементами которого могут являться произвольные значения. Реализуйте операции с объектами этого класса, как в задаче 12.21. Напишите программу, демонстрирующую работу с созданным шаблоном класса, в которой задаются различные типы элементов бинарного дерева, например целые числа, символы, комплексные числа (задача 12.14), длинные числа (задача 12.17), смешанные числа (задача 12.16). Аналогично, определите шаблон класса для произвольного бинарного дерева поиска (см. задачу 12.22).

12.42. Произвольная матрица. Опишите шаблон класса, реализующий матрицу, элементами которой могут являться произвольные значения. Реализуйте допустимые операции с объектами этого класса, как в задаче 12.26. Напишите программу, демонстрирующую работу с созданным шаблоном класса, в которой задаются различные типы элементов матрицы, например целые числа, символы, комплексные числа (задача 12.14), длинные числа (задача 12.17), смешанные числа (задача 12.16).

12.43. Произвольный линейный динамический массив. Опишите шаблон класса, реализующий линейный динамический массив, элементами которой могут являться произвольные сравнимые значения. Реализуйте допустимые операции с объектами этого класса, как в задаче 12.24. Напишите программу, демонстрирующую работу с созданным шаблоном класса, в которой задаются различные типы элементов массива, например целые числа, символы, длинные числа (задача 12.17), смешанные числа (задача 12.16).

12.44. Произвольное множество. Опишите шаблон класса, реализующий множество, элементами которого могут являться произвольные значения. Реализуйте операции с объектами этого класса, как в задаче 12.28. Напишите программу, демонстрирующую работу с созданным шаблоном класса, в которой задаются различные типы элементов множества, например целые числа, символы, комплексные числа (задача 12.14), длинные числа (задача 12.17), смешанные числа (задача 12.16).

12.45. Произвольный файл. Опишите шаблон класса, реализующий типизированный файл внешней памяти, элементами которого могут являться значения какого-то произвольного типа. Реализуйте операции с объектами этого класса, как в задаче 12.30. Напишите программу, демонстрирующую работу с созданным шаблоном класса, в которой задаются различные типы элементов множества, например целые числа, символы, комплексные числа (задача 12.14), длинные числа (задача 12.17), сме-

шанные числа (задача 12.16), множества (задача 12.28), бинарные деревья (задача 12.21) или другие классы данного раздела.

12.46. Решение квадратных уравнений. Опишите шаблон класса, реализующий алгоритм полного решения уравнения $ax^2 + bx + c = 0$, где a, b и c - значения одного из числовых типов:

- вещественные числа;
- комплексные числа (задача 12.14);
- длинные числа (задача 12.17).

Напишите программу, демонстрирующую работу с созданным шаблоном класса, в которой задаются различные числовые типы.

12.47. Решение системы линейных уравнений для разных типов. Опишите шаблон класса, реализующий алгоритм решения системы из n линейных уравнений с n неизвестными (как в задаче 12.36), для произвольных числовых типов значений коэффициентов и неизвестных. Напишите программу, демонстрирующую работу с созданным шаблоном класса, в которой задаются различные числовые типы, например комплексные числа (задача 12.14), длинные числа (задача 12.17), смешанные числа (задача 12.16).

12.48. Приведите примеры современных систем программирования, в которых используются объектно-ориентированный подход?

12.49. Сформулируйте достоинства и недостатки методологии объектно-ориентированного программирования в сравнении с другими технологиями по следующим аспектам:

а) сложность создаваемой системы;
б) временные затраты на разработку системы;
в) автоматизация отдельных видов деятельности при создании системы;
г) количество разработчиков системы;
д) уровень квалификации разработчиков системы;
е) повторное использование кода;
ж) уровень абстракции используемых понятий;
з) вероятность допущения ошибки;
и) время обнаружения ошибки;
к) объём исходного кода и временные характеристики системы.

12.50. Приведите пример задания на разработку программы (программной системы), в процессе которой ООП применять не желательно.

Глава 13

Сентенциальные методы

13.1. Конкретизация. Что такое конкретизация?

В задачах 13.2-13.5 для программирования рекомендуется пользоваться языком РЕФАЛ.

13.2. Одинаковые подстроки Написать программу, которая читает сроку и определяет, есть ли в ней одинаковые непересекающиеся подстроки. Если есть, печатает одну самую длинную из них, если нет, печатает: «Нет одинаковых частей».

Пример

Строка: *abcdefxyzcdezyx*
Ответ: cde

13.3. Перестановкой двух частей. Можно ли разбить заданную строку на две части так, чтобы, переставив их местами, получить результирующую? Если можно, то напечатать эти части по одной в строке.

Пример

Исходная: *строка*
Результирующая: *кастро*
стро
ка

13.4. Порядочная строка. Назовём *порядочной* строку состоящую только из заглавных латинских букв, в которой все буквы идут по алфавиту. Написать программу, которая читает сроку и определяет, какие операции сдвига нужно сделать, чтобы получить *порядочную* строку из заданной. Причём, этих операций должно быть минимальное возможное число. Операция сдвига — это перенос подстроки на несколько символов

влево. Операция указывается тройкой чисел: <номер первого символа подстроки>, <номер последнего символа подстроки>, <количество символов сдвига>.

Пример

```
Строка: DCBA
Ответ:
4 4 3
4 4 2
4 4 1
```

13.5. Делёж строки. На сколько, минимум, кусков нужно разделить исходную строку, чтобы, переставляя эти куски, можно было получить результирующую?

Пример

```
Исходная: test
Результирующая: tset
на 4
```

13.6. Отождествление. Что такое отождествление?

13.7. Refal-отождествление. Отождествите, как в стандарте языка РЕФАЛ:

а) `A B` с `'abcd'`;
б) `A C 'd'` с `'abcd'`;
в) `A B 'c' C` с `'abcd'`.
г) `A B 'd' C` с `'abcd'`.
д) `A B C D E` с `'aaaa'`.

13.8. Общее отождествление. Сколько вариантов отождествления может быть для следующих случаев:

а) `A B` с `'abcd'`;
б) `A C 'd'` с `'abcd'`;
в) `A A 'c' C` с `'acac'`.
г) `A B 'c'` с `'abcd'`.
д) `A B C D E` с `'abcd'`.

13.9. Отождествите:

а) `A B A` с `'ababab'`;
б) `A 'd' B C D` с `'abcd'`;
в) `A A A A` с `'abcd'`.
г) `'b' A B 'd'` с `'abcd'`.
д) `T E C T` с `'тест тест тест'`.

13.10. Только Refal. Приведите пример программы на языке РЕФАЛ, которую нельзя переписать на языке Prolog.

13.11. Только Prolog. Приведите пример программы на языке Prolog, которую нельзя переписать на языке РЕФАЛ.

В задачах 13.12-13.15 для программирования рекомендуется пользоваться языком Prolog.

13.12. Победители соревнования. Есть база фактов вида: X победил Y. Напишите программу, которая будет отвечать на следующие вопросы:
Кто проиграл X?
Сколько побед одержал X?
Может ли X победить Y (может, если он уже победил или X может победить того, кто может победить Y)?
Верно ли, что есть X, который может победить всех остальных?
Корректна ли база (никто не может победить сам себя)?

13.13. Родственники. Есть база фактов вида: X дочь Y и Z; X сын Y и Z. Напишите программу, которая будет отвечать на следующие вопросы:
Кто родители X?
Каков пол X?
Родственники ли X и Y?
Кто является потомками X?
Верно ли что есть одна пара, от которой все произошли?
Корректна ли база (не является ли кто-то своим предком, у X ровно один отец и одна мать и т.п.)?

13.14. Планирование поездок. Есть база фактов вида: Автобус X проезжает остановку Y во время T. Напишите программу, которая будет отвечать на следующие вопросы:
Сколько всего автобусов?
Сколько всего остановок?
Каков маршрут автобуса X?
Верно ли, что с любой остановки можно добраться до любой другой?
Верно ли, что нет таких двух остановок и такого автобуса, что этот автобус ходит через несколько других остановок быстрее чем какой-то другой автобус напрямик (без промежуточных остановок)?
Как быстрее всего добраться с остановки A во время T до остановки B?
Корректна ли база (нет ли двух остановок на которых автобус оказывается одновременно)?

13.15. Трёхместная перевозка. Жил-был на свете мужик. Случилось как-то ему купить на базаре двух волков, собаку, козу и капусту. Отправился он домой. По пути домой нужно переправиться через реку. Да вот беда — в лодке всего три места. Одно займёт он сам, а два других его покупки. Беда была бы не велика, съездить несколько раз не трудно. Однако покупки не сидят спокойно. Если их без присмотра: волк пытается съесть козу, два волка загрызут собаку, собака покусает козу, ну а коза съест капусту.

Нужно составить программу, которая подскажет мужику, как перевезти все покупки без ущерба.

13.16. В чем отличие процедурной семантики от декларативной?

13.17. В чем сила декларативной семантики?

13.18. Что такое поле зрения в языке Prolog?

13.19. Что такое поле памяти в языке Prolog?

13.20. Унификация. Что такое унификация?

13.21. Унифицируйте:
а) A(X) и A(1);
б) A(X,X) и A(1,2);
в) A(X,Y,X) и A(2,2,Z);
г) A(X,Y) и B(X,Y);
д) A(X,Y) и A(Y,X);
е) A(X,1) и A(2,Y);
ж) A(X,Y,Z) и A(X,Z).

13.22. Унифицируйте:
а) A(X), A(Y) и A (X+Y);
б) A(X,Y), A(Y,Z) и A(Y,Z);
в) A(X,Y,Z), A(Z,Y,X), A(X,Y,X) и A(Z,Y,Z);
г) A(X,Y,Z), A(Y,Y,X), A(X,Z,Z) и A(Y,Y,Y);
д) A(X,Y,Z), A(X,Y), A(X).

13.23. Чем унификация отличается от конкретизации?

13.24. ❐ **Скупка акций.** Некоторая компания располагает сведениями о покупке акций фирмами. После перепроверки источников стало известно, сколько из этих сведений неверны. Задача — выяснить, какая фирма какие акции покупает на самом деле.

Техническое требование

Исходная информация содержится в файле.

Структура файла:

```
<Количество строк с информацией>
Фирма [не] "<название>" [не] покупает акции фирмы [не] "<название>"
Фирма [не] "<название>" [не] покупает акции фирмы [не] "<название>"
...
Фирма [не] "<название>" [не] покупает акции фирмы [не] "<название>"
<Количество неверных сведений>
```

Фирм не больше 5. Строк с информацией не больше 100. Совпадающих сведений нет. Файл синтаксически верен. Квадратные скобки указывают опциональное вхождение.

Напечатать все стопроцентно верные сведения о фирмах из исходного набора либо строчку "Ничего не ясно".

Пример

Во входном файле:

3

Фирма "Рога и копыта" покупает акции фирмы "Копыта и рога"

Фирма "Рога и копыта" не покупает акции фирмы "Копыта и рога"

Фирма не "Копыта и рога" покупает акции фирмы "Копыта и рога"

1

Ответ.

```
Фирма не "Копыта и рога" покупает акции фирмы "Копыта и рога"
```

13.25. Закономерность. Даны десять двадцатизначных целых неотрицательных десятичных чисел. Определить, какова закономерность получения очередного числа и записать одиннадцатое.

О закономерностях известно лишь, что

- следующее число зависит только от предыдущего;
- следующее число состоит из тех же цифр, что и предыдущее.

Пример

```
Числа:
12345678901234567890
01234567890123456789
90123456789012345678
89012345678901234567
78901234567890123456
67890123456789012345
56789012345678901234
45678901234567890123
34567890123456789012
```

23456789012345678901
Очередное число: 12345678901234567890

13.26. Приведите пример синтаксической конструкции, которую невозможно описать в HTML.

13.27. Приведите пример синтаксической конструкции, которую невозможно описать в SGML.

13.28. Приведите пример структуры, которую невозможно описать в XML.

13.29. Приведите пример синтаксической конструкции, которую невозможно описать в TEX.

13.30. Конвертор HTML в LATEX. Напишите программу конвертирования текста на языке HTML в текст на языке LATEX.

13.31. XML-конвертор HTML в LATEX. Решите задачу 13.30 с помощью технологии XML.

13.32. Эквивалентные преобразования. Даны два арифметических выражения, построенные из переменных, целых чисел (до 9 знаков), операций '+', '−', '∗' и круглых скобок. Определить, эквивалентны[1] они или нет. Эквивалентность, в данном случае, лучше определять, путём преобразования к какому-то общему виду с последующим сравнением.

Пример

Первое выражение: *99999*(99999+A)-(B-C)-C*
Второе выражение: *-333333*(A+1000000-1)*(-3)-B*
Эквивалентны

13.33. Дифференцирование. Задана функция, зависящая от x. Вычислить её производную.

Пример

F(x) = *(1-x)/g(x)*
F'(x)= ((x-1)*g'(x)-g(x))/g^2(x)

13.34. Эквивалентные программы. В файлах записаны две программы. Для записи используются только переменные единственного

[1] Арифметические выражения эквивалентны, если принимают одинаковые значения при одинаковых значениях входящих в них переменных.

целого типа и управляющие конструкции только вида `if`. Остальные операции не исключены.

Определить, для всех ли одинаковых значений входных переменных будут выданы одинаковые результаты.

13.35. **Расшифровка.** Известно, что некоторый текст на некотором языке, не обязательно грамматически правильный, написан кириллицей в кодировке DOS (cp866) (возможно, с вкраплениями других символов ASCII) и зашифрован следующим алгоритмом.

- Выбрано некоторое простое число p, оно дополняется до четного числа k шестнадцатеричных знаков. $k/2-1$ символов текста, идущих подряд, рассматриваются как шестнадцатеричное число.
- Это число возводится в некоторую степень $m < p-1$, берется по модулю p и записывается как результат шифровки.
- В конце текста может быть много лишних байтов, чтобы нельзя было оценить число k по делителям длины текста и применить статистические методы дешифровки.

Но Вашим заказчикам известно, что в тексте встречается несколько важных для них слов, причем неоднократно, например,

```
бугор
Кутдузов
стрелка
разборка
автомат
доллар
```

Беда в том, что текст может быть написан не на русском языке (а, скажем, на чеченском или еще каком-то другом из Вам неизвестных). Увидев подлинный текст, заказчик сразу его опознает и по содержанию, и по характерным грамматическим ошибкам. Поэтому Ваша задача — расшифровать текст либо выдать несколько вариантов расшифровки, поскольку заказчик выберет содержательный текст из компьютерной чуши и сам.

Вторая беда состоит в том, что в естественном языке слова изменяются, так что окончания у слов могут быть и другими, сохраняются лишь основы. Правда, основы можно считать такими же, как и в русском языке.

Поскольку заказчик не доверяет Вам (и правильно!), программа будет работать без Вас. Так что сами отобрать правдоподобные варианты расшифровки вы не сможете, и даже подлинного текста шифровки или ключевых слов Вам в руки не дадут.

Техническое требование

Длина текста не более 5 страниц (10K).

Число ключевых слов не более 16, каждое не более 16 байтов, слово от слова отделяются переводом строки. Вместе с текстом задается допустимое количество вариантов расшифровки. Рассчитывайте на 4 значения данного параметра: 1, 10, 100, 1000. $2^{16} < p < 2^{48}$.

Время работы программы на одном тесте не более 1 мин. на Pentium 100 MHz.

Формат входного файла *message.bin*

```
Число вариантов
Ключевые слова
***
Текст шифровки
```

Формат выходного файла *decyph.bin*

```
Вариант 1
...
***
Вариант 2
....
***
...

***
Конец!
```

И тот, и другой файл двоичные. Перед $***$, разделяющими варианты, вставлять перевод строки не обязательно. Программа может пользоваться вспомогательным файлом величиной не более 1 МБ, и занимать оперативной памяти не более 4 МБ.

На соревнованиях оценивали так

Если Вы выдали меньше максимального числа вариантов, и правильный находится среди них, Ваша оценка повышается. Тот, кто выдал минимальное число вариантов для данного теста, получает премиальные очки.

13.36. Даны две строки. Можно ли вставить N символов (возможно

одинаковых) в первую строку так, чтобы получилась вторая? Если можно, то напечатать минимальное число N.

Длина входных строк не более 200 символов.

Пример

```
Первая: 12345
Вторая: 12131415
3
```

13.37. ✣ **Игра в фигуры.** Два школьника играют в такую игру: первый ставит четыре различные точки с целыми координатами. А второй соединяет эти точки отрезками. Каждая точка должна оказаться концом ровно двух отрезков. Общими точками отрезков могут быть те, что поставил первый игрок. После построения фигуры считают очки второго игрока по следующей таблице (в зависимости от получившейся фигуры):

Невыпуклый четырехугольник	0
Выпуклый четырехугольник	1
Трапеция	2
Параллелограм	3
Прямоугольник	4
Треугольник	5
Ромб	6
Квадрат	7
Есть совпадающие точки	8
Построить нельзя	9

Техническое требование

Ограничимся координатами от 0 до 9 по осям X и Y. Задавать координаты точек будем восьмизначным числом. Тогда первые две цифры будут координатами первой точки, вторые – второй точки и т.д. Нужно определить, какое максимальное количество очков может получить второй игрок.

Пример

```
Точки: 00900963
Максимальное количество очков - 5
```

13.38. Дано классическое регулярное выражение и набор строк (в файле). Определить, какие строки не подходят под это выражение.

13.39. Дан набор строк (в файле). Построить классическое регулярное выражение для их описания.

13.40. Приведите примеры современных систем программирования, в которых используются сентенциальные методы?

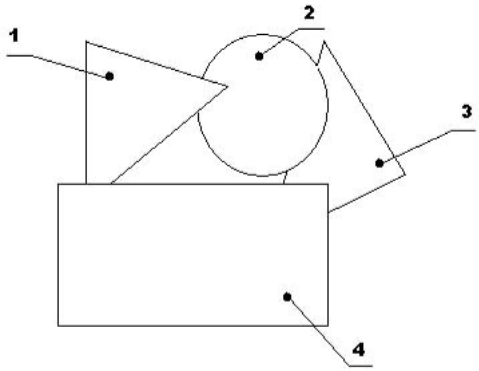


Рис. 13.1: Пример оформления выносок.

13.41. Выноски на рисунках. Задан набор различных геометрических фигур, возможно, перекрывающих друг друга. Нужно вывести этот набор в графическом виде и добавить выноску для каждой фигуры с указанием её номера. Выноски не должны пересекаться, и, желательно, не задевать другие фигуры. Каждая фигура задается номером типа фигуры (1 — отрезок, 2 — круг, 3 — треугольник, 4 — четырехугольник; можно добавить свои), и её параметрами (для каждого типа — своими), например, для треугольника задаются координаты вершин, для круга — координаты центра и радиус и т.д. Кроме этого, задается номер слоя для размещения каждой фигуры. Слои с меньшими номерами расположены ниже. Пример оформления выносок см. на рис. 13.1.

Пример

```
Введите параметры фигур, в каждой строке - по одной фигуре и
номера слоёв:
3 100 100 280 120 100 300 3
2 320 150 80 2
3 400 100 500 250 320 320 1
4 80 250 80 450 420 250 420 450 4
```

13.42. В строку $x_1?x_2?x_3?...?x_n$ вместо знаков '?' поставить '+' либо '$*$' так, чтобы полученное выражение равнялось S. Операция + обозначает конкатенацию строк, $A + B + C = (A + B) + C$. Например, $adbc + 123 = abcd123$. Результатом операции $A * B$ будет строка, где после каждого символа A записана B; если A или B — пустые строки, то результат будет также пустой строкой, $A*B*C = (A*B)*C$. Например, $ab*cd = acdbcd$. Операции выполняются в порядке их следования.

x_i — строки, их длина не более пяти символов. Количество строк (n) не более 20.

Строки x_i вводятся до тех пор пока не введут пустую строку, затем вводится строка S.

Длина S не более 100 символов.

Пример

A
ab
cd
x

Axaxbxcxdx
A+abcd*x =Axaxbxcxdx

13.43. Между некоторыми символами строки X поставить знаки '*' (операция описана в задаче 13.42) так, чтобы полученное выражение равнялось S.

Длины X и S не более 100 символов.

Строки X и S вводятся.

Пример

Тест
Тметемст
Те*с*т = Ттстетст

13.44. **XOR-шифрование.** Известен такой способ шифрования. Выбирается один символ из алфавита — ключ. Вместо каждого символа текста записывается результат операции XOR между кодом этого символа и кодом ключа.

Написать программу, которая читает файл input.xor с зашифрованным текстом и записывает в файл output.txt исходный (не зашифрованный текст).

Техническое требование

Исходный текст написан символами в кодировке ASCII. В алфавите находятся русские буквы, знаки препинания и пробелы.

Операция XOR — побитовая операция. Работает она так: если операнды одинаковые, то записывается 0, если разные — 1.

Зашифрованный текст представляет собой последовательность двухзначных шестнадцатеричных чисел. Эти числа стоят на месте соответствующих символов и обозначают его зашифрованный код.

Переводы строк не шифруются и остаются в исходном виде.

После знаков препинания обязательно стоит пробел или переход на новую строку или знак препинания, но никогда пробел или переход на

новую строку не стоит перед знаком препинания. Исключение составляют символы: открывающая кавычка (") и тире. Перед и после тире всегда стоит пробел или перевод строки. Тот же знак (-) используется в качестве дефиса: в этом случае перед ним стоит буква, а после него — буква или переход на новую строку.

Перед открывающей кавычкой должен стоять пробел, а после — буква. Но помните, что не любая кавычка — открывающая. По правилам русского языка, знаки препинания ставятся после закрывающей кавычки. Считается, что текст не содержит скобок.

Предупреждение. Длина зашифрованного файла — это предмет соревнования.

Пример

В файле input.xor:

```
0B3E3C3E7B39363B74BB3135343374B7BB7B3E3C3EB6
7B39363B74BB1115141304BB BBBBBBBBBBBBBBBBBBBB33BB
3E726ABB7B3B3CBB0B1E1C1E0B19161B04BB1115141304BA
```

В файле output.txt:

```
Резервная копия, резе- рвная КОПИЯ
          и
ещё раз РЕЗЕРВНАЯ КОПИЯ!
```

13.45. ☐ **Девятизначные быки и коровы.** В «Быки и коровы» играют вдвоём. Первый игрок загадывает число, второй игрок его отгадывает. Для уточнения числа второй записывает число, а первый говорит, сколько в нем быков, а сколько коров. Такое уточнение называется ходом. Быком называется цифра числа, которая стоит на том же месте, что и в загаданном числе и совпадает с ним по значению. Коровой называется цифра числа, которая есть в загаданном числе, но не является быком.

Второй игрок записывает число. В ответ первый записывает, сколько в этом числе быков и сколько коров.

Цель игры

Определить за наименьшее количество ходов загаданное число.

Варианты

Можно договариваться о разном количестве цифр в числе. Интересные варианты начинаются с 4.

Задача

Написать программы для первого и для второго игрока отдельно. Цифры будут десятичные от 0 до 9 и их количество в числе ровно 9. Цифры в числе могут быть одинаковыми. Назовём программу для первого игрока «загадыватель», а для второго — «отгадыватель».

Техническое требование

Программы общаются через файл *numbers.txt*. Обе программы имеют право только дописывать этот файл. Т.е. все предыдущие строки нужно сохранять. В файле numbers.txt заполняется так: В первой строке загадыватель записывает десятичное число из 20-ти знаков. В нем закодировано загаданное число. В следующей строке отгадыватель записывает девятизначное число (если нужно дополнив незначащими нулями). В третьей строке загадыватель обязан написать двухзначное число. Первая цифра — количество быков, вторая цифра — количество коров в числе, написанном в предыдущей строке.

Игра продолжается до момента пока загадывающий не напишет 90 (9 быков 0 коров). Время на ход — 1 секунда. Если игра достигла 100 ходов, отгадывающему записывается поражение.

Пример

number.txt после окончания игры

```
00112233445566778899
123456789
04
000000001
45
123404321
09
010203040
27
000011234
90
```

Оценка задачи

Между программами будет устроен турнир. Можно получить до 100 баллов.

Блеф

Эта задача предполагает блеф. Загадывателю разрешается менять загаданное число в во время игры, сообщать не верное количество быков и коров. Но отгадыватель может потребовать проверку. Для этого нужно записать не девятизначное число, а -1. И, если он окажется прав, то ему будет засчитан выигрыш за 1 ход. Если проверяющая программа не найдет обман, то отгадывателю записывается проигрыш.

Отгадыватель, в свою очередь, может пытаться раскодировать число из первой строки.

Все остальные жульничества запрещаются и за них программа может быть снята с соревнований.

13.46. Кубик-рубика. Напишите программу, которая предоставляет пользователю возможность осуществлять перемещения в известной головоломке кубик-рубика. Предусмотрите вариант автоматической сборки с демонстрацией.

13.47. Тетраэдр-рубика. Напишите программу, которая предоставляет пользователю возможность осуществлять перемещения в известной головоломке тетраэдр-рубика. Предусмотрите вариант автоматической сборки с демонстрацией.

13.48. Solitaire. Написать программу, решающую головоломку «Solitaire». Исходное положение головоломки показано на рисунке 13.2.

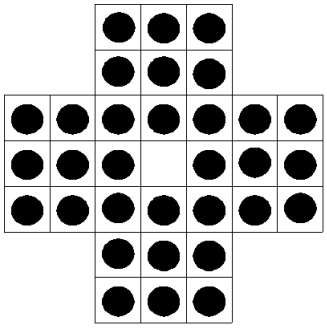


Рис. 13.2: Исходное положение в головоломке «Solitaire»

Правила.

а) ход заключается в перепрыгивании по горизонтали или по вертикали (при запрете движения по диагонали) некоторой выбранной фишки через соседнюю фишку на свободное поле и со снятием при этом с поля фишки, через которую перескочили,

б) никаких других перемещений и удалений фишек с поля быть не может.

Головоломка считается решенной, если осталась одна фишка в центре.

13.49. Восстановление таблицы результатов. Результаты однокругового турнира для парных встреч записываются в таблицу результатов следующим образом. Результаты каждого участника записываются в столбец, соответствующий номеру участника, и в строке, соответствующей номеру противника. В каждой встрече разыгрывается 2 очка. Выигравшему записывается 2 очка, проигравшему — 0. В случае ничьей — обоим по 1 очку. В последний столбец записывают суммарные результаты. Сами с собой участники не встречаются, и поэтому по диагонали клетки не используются.

По некоторым причинам таблица результатов была повреждена. И некоторые данные прочитать не удаётся.

Задача

Напишите программу, которая по испорченной таблице сможет восстановить как можно больше данных.

Техническое требование

В файле *table.txt* хранится испорченная таблица. Столбцы таблицы выровнены. В первой строке записано число участников. Их будет не более 50. Далее идут строки таблицы. Первая строка — номера участников, а в конце — слово "Всего". Дальше идут сроки с результатами, где символы означают следующее:

'0' - проигрыш,
'1' - ничья,
'2' - выигрыш
'X' - обозначение диагонали (сам с собой не встречается),
'?' - потерянная информация.

Последнее число каждой строки является суммарным результатом участника.

Программа должна читать файл *table.txt* и записывать один из вариантов восстановленной таблицы в файл *correct.txt*.

Формат файла *correct.txt* такой же, как и *table.txt*, но в нем не будет символов '?'.

Формат файла *table.txt* всегда будет соблюдаться.

Время работы программы не более 30 секунд на компьютере класса P-200.

Пример

table.txt

```
3
01 02 03 Всего
 X  ?  ?  ?
 1  X  ?  1
 ?  ?  X  2
```

correct.txt

```
3
01 02 03 Всего
 X  1  2  3
 1  X  0  1
 0  2  X  2
```

13.50. Плитки майя. Написать программу решающую головоломку

«Плитки майя». Форматы входных и выходных данных, а также ограничения придумайте сами.

Исходное положение таково. Есть различные канавки, в них имеются ямки разного цвета. В некоторых из канавок находятся плитки, не более одной плитке в ямке. Плитки разноцветные. Чёрная плитка может быть только одна.

Один из вариантов исходного положения головоломки показан на рисунке 13.3, где цвета обозначены числами.

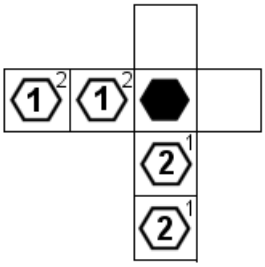


Рис. 13.3: Пример одной из головоломок «Плитки майя»

Правила перемещения плиток. Ход заключается в перемещении любой плитки по ямкам на любую свободную ямку вдоль канавок из ямок, причем за один ход можно переместить плитку вдоль любого количества ямок, но перепрыгивать через другие плитки запрещено.

Головоломка считается решенной, если плитки после всех перемещений удалось расположить в ямках-углублениях того же цвета, что и сами плитки, причем чёрная плитка должна остаться на своём месте.

13.51. Доехать вовремя. Нужно доехать из пункта A в пункт B. Дорог много. Каждая дорога имеет своё время и стоимость проезда. Нужно добраться за время не большее T, потратив на это как можно меньше денег.

Техническое требование

Структуру входных и выходных данных выбирайте ту, которая удобнее.

13.52. Японские сканворды. В японских сканвордах зашифрованы не слова, а изображения. Задача — восстановить картинку по числам, которые проставлены слева от строк и над колонками. Числа показывают, сколько групп черных клеток находится в соответствующей строке или колонке и сколько слитных черных клеток содержит каждая группа.

Например, набор чисел 4, 1, и 3 означает, что в этом ряду есть три группы: первая — из четырех, вторая — из одной, третья — из трех черных клеток. Группы разделены как минимум одной пустой клеткой. Пустые клетки могут быть и по краям рядов. Необходимо определить размещение групп клеток.

Написать программу решающую японские одноцветные сканворды.

Пример

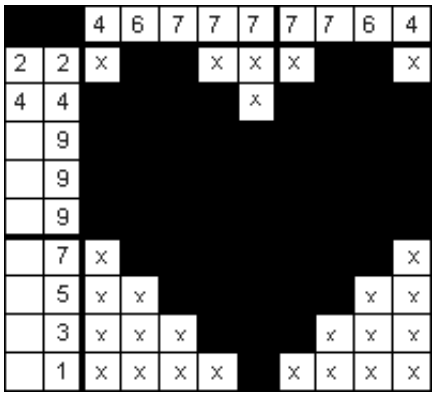


Рис. 13.4: Пример разгаданного японского сканворда

Пример разгаданного японского сканворда показан на рисунке 13.4.

13.53. Линпикс. Линпикс (филиппинский кроссворд, восточный сканворд) — это такая головоломка, в которой с помощью чисел зашифрована какая-то картинка. С помощью логических рассуждений необходимо выявить пары одинаковых чисел, количество клеток между которыми равняется этим числам. Таким образом, эти числа будут находиться на концах линии. Линии не должны пересекаться между собой. Написать программу решающую линпиксы.

Пример

Пример разгаданного линпикса показан на рисунке 13.5.

13.54. Составитель японских сканвордов. Написать программу, составляющую японские одноцветные сканворды (см. 13.52).

13.55. Составитель линпиксов. Написать программу, составляющую линпиксы (см. 13.53).

13.56. Приглашения на юбилей. Некто z хочет пригласить на свой юбилей только симпатизирующих друг другу знакомых. Какие правила r приглашения позволят это осуществить:

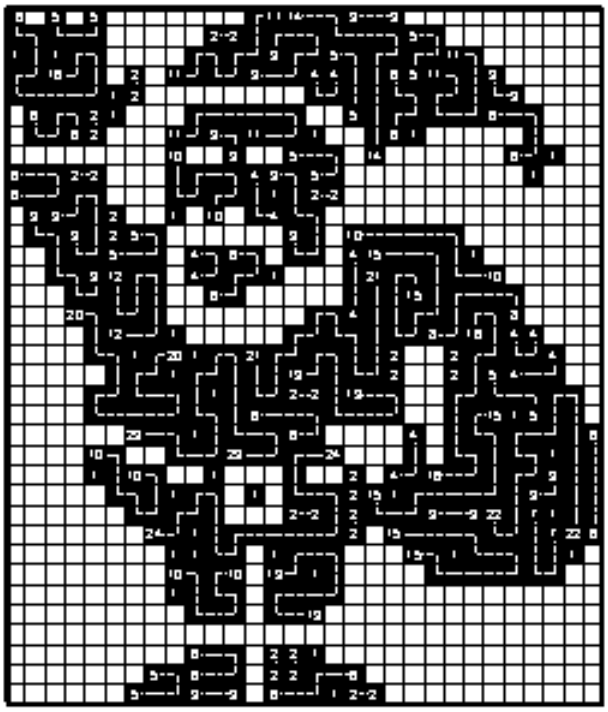


Рис. 13.5: Пример разгаданного линпикса

а) r(x,y): «x симпатизирует y»;
б) r(x,y,z): «x и y симпатизируют z»;
в) r(x,y): «x и y не имеют лиц, не симпатичных им»;
г) r(x): «x симпатичен всем»?

Придумайте одно правило, позволяющее осуществить такое приглашение. Напишите соответствующую программу.

13.57. Интуиционистская система. Язык программирования Prolog основан на классической логике, а какой бы могла быть система программирования, основанная на интуиционистской логике?

13.58. Многозначная логика. Какой бы могла быть система программирования, основанная на многозначной логике?

13.59. Нечёткая логика. Где можно использовать нечёткую логику (и, кстати, где её уже используют)?

13.60. ❐ **Задача риэлтера.** Есть несколько человек желающих про-

дать, купить или обменять квартиры. Риэлтер должен помочь им в этом и сам остаться с прибылью. Все обмены осуществляются через продажу старых квартир и покупку новых.

Техническое требование

Все данные хранятся в файле *exchange.txt*. В каждой строке файла записана информация ровно об одном обмене. В строке содержится информация в описанном ниже порядке.

```
<пробелы>
<номер обмена>
<пробелы>
<количество комнат первой продаваемой квартиры><пробелы><её цена>
','
<количество комнат второй продаваемой квартиры><пробелы><её цена>
','
...
<количество комнат последней продаваемой квартиры><пробелы><её цена>
';'
<пробелы>
<максимальная сумма доплаты>
<пробелы>
'='
<пробелы>
<количество комнат первой желаемой квартиры><пробелы><её цена>
','
<количество комнат второй желаемой квартиры><пробелы><её цена>
','
...
<количество комнат последней желаемой квартиры><пробелы><её цена>
';'
<пробелы>
<минимальная сумма выручки>
```

Цель

Найти цепочку взаимных обменов, выгодную для риэлтера. Выдать в файл *output.txt* последовательность номеров обмена, каждый номер в своей строке.

Пример

В файле *exchange.txt*

```
20  1 300, 2 400; 0        = 3  600; 0
10  2 500, 1 200, 3 700; 0 =       ; 1400
```

```
12  5 1100;  100          = 1  200, 1 200, 2 400; 0
 1  5 1000;    0          =       ; 1100
 2        ; 1305          = 4 1100; 0
```

В файле *output.txt*

```
20
10
12
1
```

Задание 1

Написать программу для выполнения цели, которая описана выше.

Задание 2

Предоставить пару файлов (*exchange.txt*, *output.txt*) — тест, для проверки задания 1.

Оценка заданий

Оценка задания 1 соревновательная. Чем более выгодную цепочку обмена находит Ваша программа, тем больше баллов она получает.

13.61. Какое понятие алгоритма лежит в основе языка программирования Prolog?

13.62. Какое понятие алгоритма лежит в основе языка программирования РЕФАЛ?

13.63. Одной из основных составляющих XML-технологий являются XSLT-преобразования. Напишите программу, которая будет их выполнять, не используя готовые пакеты.

13.64. Экспертная система «фигура». Построить систему, которая будет задавать вопросы про плоские геометрические фигуры и идентифицировать их. У системы должна быть возможность автоматического пополнения своих знаний.

Пример

```
Есть углы: Да
Сколько углов: 3
Это треугольник? Да
```

13.65. Признаки делимости. Написать программу, которая для заданного числа строит алгоритм, который является признаком делимости на это число. Всё делается в десятичной системе счисления.

Идея. Например, при делении на 7 возникает ряд остатков: 1, 3, 2, 6, 4, 5, 1... Далее остатки повторяются. Повторение возникает всегда ввиду конечности множества остатков при делении на конкретное число. Число

1999 на 7 не делится, потому что получается сумма $9\cdot1+9\cdot3+9\cdot2+1\cdot6 = 9+27+18+6 = 60$, которая на 7 не делится.

Глава 14

Функциональное программирование

14.1. Программа для расстановки ферзей. На квадратной доске размером $N \times N$ расставить N ферзей так, чтобы они не «били» друг друга.

Техническое требование

Решением задачи будет система, которая читает $3 < N < 100$ и выдает N чисел, соответствующих искомой расстановке.

Поскольку на каждой вертикали располагается ровно один ферзь, решением будет N вложенных циклов. Но, обычно, языки программирования не допускают переменную вложенность циклов.

Требуется построить программу которая по заданному N создаёт новую программу, решающую задачу только для заданного количества ферзей вложенными циклами. Затем запускает созданную программу и получает ответ.

14.2. Что можно сказать о функции $f(\overline{x})$, если $f(f(\ldots f(\overline{x})\ldots)) = c_0$?

14.3. Что можно сказать о функции $f(\overline{x})$, если $f(f(\ldots f(\overline{x})\ldots))$ — неопределено?

14.4. Что можно сказать о функции $f(\overline{x})$, если $f(f(\ldots f(\overline{x})\ldots)) \in A$, где A — конечное множество?

14.5. Что такое неподвижная точка?

14.6. ❐ **Интроспекция с версией.** Написать программу, которая при запуске с ключом '-v' печатает на экран 0, а при запуске без ключа выдаёт текст новой программы. Новая программа должна: при запуске

с ключом '-v' печатать номер поколения, а без ключа печатать новую программу с описанными выше условиями.

Например, Ваша программа была написана на каком-то языке программирования. Пусть транслятор называется trans, файл с программой называется *a.txt*, результат работы программы в файле *r.txt*.

При последовательности действий:

```
trans a.txt  ---создаёт--> a.exe
a
a -v  ---на экране--> 0
rename r.txt a.new
trans a.txt  --> a.exe
a
rename r.new a.txt
trans a.txt  --> a.exe
a
rename r.new a.txt
trans a.txt  --> a.exe
a
a -v  ---на экране--> 3
```

Техническое требование

Программа должна работать без внешних модулей и файлов. Результат работы записывается в файл **.new* в текущей директории, где * — имя Вашей программы. Например, программа *input.pas* должна выдавать файл *input.new*.

Как оценивались программы во время соревнований

Программа оценивалась следующим образом: программа была выполнена, результат прочитан из файла, оттранслирован, снова выполнен, полученная программа взята за исходную, и так N раз. Потомок с номером N был вызван с ключом '-v' и, если он выдавал число, отличное от N, то такое решение получало 0 баллов. 0 баллов также получали программы, у которых какой-либо потомок содержал синтаксическую ошибку.

Правильные решения могли принести 40 баллов. Дополнительно 20 баллов давались за близость старого и нового текстов. Длина во внимание не принималась.

14.7. **Уравнение $\mathbf{f(X) = A}$.** Решить уравнение вида $f(X) = A$, где A — строка из маленьких латинских букв, X — переменная строка, f — функция на строках, построенная с помощью операций $*$ и $+$ над строками. Операция $+$ обозначает конкатенацию строк, $A + B + C =$

$(A + B) + C$. Например, $adbc + 123 = abcd123$. Результатом операции $A*B$ будет строка, где после каждого символа A записана B; если A или B — пустые строки, то результат будет также пустой строкой, $A*B*C = (A*B)*C$. Например, $ab*cd = acdbcd$. Операции выполняются в порядке их следования.

Техническое требование

Программа должна запросить уравнение и напечатать одно из решений. Строка может содержать разделительные пробелы. Проверка на синтаксическую правильность не требуется.

Пример

```
Уравнение: X*ab + (c * X) = aabcabcac
X=ac
```

14.8. Минимум с дополнительной информацией. В заданной последовательности чисел $X_1, X_2, \ldots, X_n$ найти минимальный элемент. Известно, что для некоторых пар (i, j) есть дополнительная информация об этой последовательности вида "$i < j$", которая означает, что $X_i < X_j$. Желательно в алгоритме воспользоваться дополнительной информацией для повышения эффективности.

Техническое требование

Способы задания последовательности чисел и множества пар дополнительной информации выберите сами. Важно рассмотреть случай, когда дополнительная информация меняется значительно реже чем последовательность.

Пример

```
Последовательность:10 1 20 0
Информация: 2<1 4<3 4<2
Ответ: 0
```

14.9. Сортировка с дополнительной информацией. Отсортировать заданную последовательность чисел $X_1, X_2, \ldots, X_n$. Известно, что для некоторых пар (i, j) есть дополнительная информация об этой последовательности вида "$i < j$", которая означает, что $X_i < X_j$. Желательно в алгоритме воспользоваться дополнительной информацией для повышения эффективности.

Техническое требование

Способы задания последовательности чисел и множества пар дополнительной информации выберите сами. Важно рассмотреть случай, когда дополнительная информация меняется значительно реже чем последовательность.

14.10. Какое понятие алгоритма лежит в основе языка программирования Lisp?

14.11. **MicroLisp.** Простейший язык функционального программирования задан следующим образом.

Переменными и константами являются малые буквы латинского алфавита.

Элементарными типами являются большие буквы латинского алфавита.

Типы строятся из элементарных по следующим правилам:

Если $T_1, T_2, \ldots, T_n$ — типы, то и $(T_1 T_2 \ldots Tn)$ — тип $(n \geqslant 2)$.

Если $T_1, \ldots, Tn, T$ — типы, то и $(T_1 \ldots T_n \to T)$ $(n \geqslant 1)$ тоже тип.

Выражения языка

Переменные и константы считаются выражениями языка.

Применение функции — запись вида $Apply(EE_1 \ldots E_n)$, где число аргументов не меньше 1, каждый из аргументов является выражением. Если E имеет тип $(A_1 \ldots A_n \to B)$, каждое из E_i— тип A_i, то все это выражение имеет тип B.

Описание функции — запись вида $Lambda(x_1 \ldots x_n E)$, где все x_i $(1 \leqslant i \leqslant n)$ — различные переменные, E— произвольное выражение. Внутри E все x_i считаются связанными. Если считать, что связанным переменным приписаны типы $T_1, \ldots, T_n$, и после этого оказывается, что выражение E имеет тип T, то описание функции имеет тип $(T_1 \ldots T_n \to T)$.

Список — запись вида $(E_1 E_2 \ldots E_n)$. Если E_i имеют тип T_i, то список имеет тип $(T_1 T_2 \ldots T_n)$.

Извлечение члена списка — запись вида $Mem(i, E)$, где E имеет тип списка не менее чем с i компонентами.

Постановка задачи

В начале входного файла *input.txt* идут описания констант, а затем идет тип.

Пример входного файла:

```
const
a: (A B -> A)
h: (A B)
j: D
result A
```

Все буквы, не описанные в списке констант, считаются переменными.

Нужно построить выражение, не имеющее свободных переменных и имеющее данный тип. Типы связанных переменных могут быть назначены произвольно, но одна и та же переменная, связанная одним и тем же квантором *Lambda*, всегда имеет один и тот же тип.

В частности, для данной задачи правильным ответом является
`Apply(a Mem(1,h) Mem(2,h))`

Ответ нужно записать в файл *output.txt*.

Каждая строка входного файла содержит описание одной константы и не длиннее 250 символов. Выходной файл должен не превышать 1000 символов. Разбиение на строки несущественно, но служебные слова *Apply*, *Mem* и *Lambda* разрывать нельзя.

Внимание! Одна и та же задача может допускать много (в принципе бесконечно много) правильных ответов. Например, для приведенной выше задачи правильными ответами являются также:
`Mem(1,h)`
`Apply(Lambda(x x) Mem(1,h))`

14.12. Теорема Черча-Россера 1. Что можно сказать о таком варианте формулировки теоремы Черча-Россера: «Если в языке вызовы по имени и по значению приводят к определенному результату, то это один и тот же результат»?

14.13. Теорема Черча-Россера 2. Что можно сказать о таком варианте формулировки теоремы Черча-Россера: «Если вычисление значения выражения приводит к определенному результату, то к нему всегда приводит вызов по имени и не всегда — по значению»?

14.14. Генетика на чужой планете. На планете Красивый Фингал, открытой петербургскими космонавтами, генетический код определяется тридцатью одним кодоном. Два из них определяют структуру гена и обозначаются '(' и ')'. Три из них определяют преобразования генетического кода и обозначаются I, K и S. Все остальные, как обычно, порождают организм. Они обозначаются малыми латинскими буквами. Вначале геном некоторое время преобразуется, а затем начинает расти организм, поэтому, например, у свинофингала может родиться собакофингал, а то и дубофингал. Правила преобразований следующие.

```
((X)Y) в (XY), (1)
(XY) в ((X)Y), (2)
(IA) в A,
(KAB) в A,
```

`(SABC) в ((AC)(BC))`.

Здесь A, B, C — либо кодоны, либо последовательность кодонов, сбалансированная по скобкам и заключенная в скобки, X и Y — произвольные сбалансированные по скобкам непустые последовательности кодонов. Если к некоторому фрагменту было применено преобразование (1), то к нему уже не может быть применено преобразование (2), и наоборот.

Генетический код называется стабильным, если он не может преобразовываться, кроме как по правилам (1) и (2).

Если генетический код превысит в длину 250 символов, развитие зародыша прекращается. Точно так же оно прекращается, если код преобразуется более 100 шагов.

Задание

Даны два входных файла. *Species.txt* содержит пары строк, описывающих геномы некоторых стабильных видов на планете. Первая, третья и так далее строки — имена видов, вторая строка — геном первого вида, четвертая — второго и так далее. Количество видов не более 8, длина каждой строки до 80 символов. Конец ввода — пустая строка. Биологи, как им и положено, пишут названия видов на варварской латыни, так что в строках могут быть лишь латинские буквы.
Пример файла:

```
Swintus Fingalae
(swintuS)
Canis Fingali
((((ca)n)I)s)
Chimera Baldina
(SSSSS)
```

Embrion.txt состоит из одной строки с генетическим кодом эмбриона. Для эмбриона нужно вывести историю преобразования и определить, какому виду он принадлежит.
Пример файла:

```
(SKKs(SKIw)(IIIi)(KnK)tu)S
```

В файл *output.txt* нужно вывести результат, какому виду принадлежит эмбрион, либо квалификацию генетического кода. Квалификация может иметь следующие формы:

- новый вид, когда преобразование заканчивается, но получившийся геном принадлежит не описанному стабильному виду;
- опасный, когда преобразование приводит к слишком длинному коду или может продолжаться более 100 шагов;

- фатальный, когда преобразование приводит к печальному исходу с гарантией, по Вашему мнению.

Ваш вывод должен быть обоснован предъявленным преобразованием. Файл заканчивается пустой строкой.

Пример

Выходной файл для данного случая.

```
Swintus Fingalae
((SKKs(SKIw)(IIIi)(KnK)tu)S)
(((SKKs)(SKIw)(IIIi)(KnK)tu)S)
((((Ks)(Ks))(SKIw)(IIIi)(KnK)tu)S)
(((Ks(Ks))(SKIw)(IIIi)(KnK)tu)S)
((s(SKIw)(IIIi)(KnK)tu)S)
(s(SKIw)(IIIi)(KnK)tuS)
(s(SKIw)(IIIi)ntuS)
(s(SKIw)((II)Ii)ntuS)
(s(SKIw)(IIi)ntuS)
(s(SKIw)((II)i)ntuS)
(s(SKIw)(Ii)ntuS)
(s(SKIw)intuS)
(s((Kw)(Iw))intuS)
(s(Kw(Iw))intuS)
(swintuS)
```

Внимание! Биологи, изучавшие планету, не очень грамотны в математике и программировании. Поэтому некоторые виды могут быть описаны неправильно, соответствующие геномы нестабильны; для таких геномов нужно вывести в файл *output.txt* ошибку, написав строчки вида:

```
Геном Chimera Baldina неверен
(SSSSS)
((SSSS)S)
(((SS)(SS))S)
***
```

указав неверный геном и его возможное преобразование. Диагностика ошибки в описаниях видов заканчивается строкой с тремя звездочками и не отменяет задачи распознавания эмбриона. Так что полное решение для приведенного теста получается сочленением диагностики ошибок и преобразования генома. Если Вы не продиагностировали ошибку в описании вида, то количество очков за тест снижается, но не до нуля.

14.15. Сборка мусора. Что значит с точки зрения функционального программирования централизованная «сборка мусора»?

14.16. Исключения. Что значит с точки зрения функционального программирования обработка ошибок и исключительных ситуаций?

14.17. Жулик на тестах. Лаборатория по обработке текстов разработала язык программирования СТРОКА. Этот язык предназначен для обработки строк. Для данного языка создан интерпретатор ***string***.

Среди сотрудников лаборатории решено провести турнир по владению этим языком. Для этого решено дать какую-то задачу, а решения проверять только по тестам. После того как задача и тесты были готовы один из сотрудников сумел раздобыть тесты к задаче. Помогите сотруднику выступить в этом конкурсе хорошо.

Синтаксис языка СТРОКА

Все команды записываются в виде: A r B, где A, B — строки или переменные, r — операция языка. Команда означает, что нужно вычислить результат операции r для операндов A и B, после чего записать этот результат в переменную Z.

Переменными языка являются только заглавные латинские буквы. Переменные A и Z имеют специальные значения. A — хранит входную цепочку. Значение Z будет выведено после окончания работы интерпретатора и считается результатом.

Константами языка являются любые строки, заключённые в кавычки. Строки в тестах будут текстовые, без управляющих символов и одинарных кавычек.

Команды этого языка записываются по одной в строке.

Имеются следующие операции.

$x + y$ — сцепление строк (конкатенация);

$x - y$ — удаление максимальной хвостовой части x совпадающей с некоторой головной частью y. Например, `'abcd'-'bcdb'` → `'a'`;

$x|y$ — результатом будет строка, полученная удалением из x всех символов, которых нет в y на том же расстоянии от начала. Например, `'abcd'|'accbef'` → `'ac'`;

$x\&y$ — результатом будет строка, полученная удалением из x всех символов, которые стоят в y на том же расстоянии от начала. Например, `'abcd'&'accbef'` → `'bd'`;

$x = y$ — замена символов строки x на последовательности символов y до исчерпания x. Например, `'abcdeadfrtg'='123'` → `'12312312312'`;

$X < Y$ — поместить значение переменной Y в X.

Задача

Написать программу для построения программы на языке СТРОКА по известным тестам. Тесты будут находится в файле *test.txt*. Построенная программа записывается в файл *program.str*. Эта программа должна

проходить как можно больше тестов.

Формат записи тестов в файле *test.txt*

В файле с тестами каждая нечетная строка — это входные данные, а четная — выходные.

Техническое требование

Тестов в файле будет не больше 10. Таким образом строк до 20. Размер файла с построенной программой не должен превышать 100 Мб. Каждая строка до 200 символов. Количество строк не ограничивается. Константы будут текстовые, без управляющих символов. Вторая одинарная кавычка обозначает конец константы. Время работы программы не должно превышать 50 секунд на P-200. Памяти дисковой и оперативной будет выделено не менее 1 Мб.

Пример

Тесты
```
abc
b
aaaaa
aaa
123456
2345
```
Программа
```
'1234' = ' '
Z < A
Z & '1-----'
Z & '----6'
Z & 'a----'
Z & '-a--'
Z & '-c'
```

ОЦЕНКА ЗАДАЧИ

Программы будут оцениваться по количеству пройденных тестов созданной ими программы.

14.18. Композиция. Что является аналогом операции $f \circ g$ в известных Вам системах, использующих функциональный стиль программирования?

14.19. Запомни функцию. Дана программа, вычисляющая гладкую непрерывную функцию. Написать программу, которая будет давать целые числа, наиболее близкие к значению функции на всех целых числах от 0 до 2 000 000 000.

Техническое требование

Программа должна читать число из входного потока и выдавать целое число в выходной поток. Время работы программы не более 1 секунды. Размер файла с исходным текстом программы не должен превышать 10000 байт.

Как оценивались программы во время конкурса

Всем программам подавались на вход числа в порядке возрастания. Программы, которые ошиблись более чем на 1 или не вычислившие ничего, далее не тестировались. Процесс продолжался до тех пор, пока не осталась одна программа. Баллы распределялись пропорционально количеству верных ответов.

14.20. Комбинатор I. Что является аналогом комбинатора **I** в известных Вам системах, использующих функциональный стиль программирования?

14.21. Комбинатор K. Что является аналогом комбинатора **K** в известных Вам системах, использующих функциональный стиль программирования?

14.22. Комбинатор S. Что является аналогом комбинатора **S** в известных Вам системах, использующих функциональный стиль программирования?

14.23. Ремонт стиральной машины. У Васи сломалась стиральная машина. Анализ поломки показал, что требуется заменить командоаппарат (командоаппарат — устройство, которое обеспечивает электрические соединения различных блоков стиральной машины в заданных последовательностях с заданными временными задержками). Вася купил новый командоаппарат, отсоединил старый, но тут обнаружилось, что он не помнит, какие электрические разъемы стиральной машины к каким электрическим разъемам командоаппарата следует присоединить. К счастью, Вася знает, как должна работать стиральная машина, то есть что с чем и в какой последовательности в ней должно быть соединено. Знает Вася также и электрические схемы стиральной машины и командоаппарата, а также — циклограмму командоаппарата (циклограмма — последовательность соединений внутренних контактов командоаппарата во времени). Помогите Васе правильно присоединить командоаппарат, так как неправильное соединение может привести к коротким замыканиям и поломке других блоков машины. Ваша задача состоит в написании программы, выдающей инструкцию по присоединению командоаппарата

по электрической схеме машины, схеме и циклограмме командоаппарата, а также по схемам электрических соединений, которые командоаппарат должен обеспечивать.

Техническое требование

Исходные данные содержатся в файле *input.txt* в следующем виде:
N — число разъемов командоаппарата,
M — число точек соединения в электрической схеме машины, включая N разъемов которые следует присоединить к командоаппарату, $M = N+R$, где R — число блоков стиральной машины, которые следует соединять командоаппаратом,
K — число точек соединения в электрической схеме командоаппарата, включая N внешних разъемов, к которым следует присоединить стиральную машину, $K = N + 2Q$, где Q — число контактных пар командоаппарата, которые соединяются и разъединяются в соответствии с его циклограммой,
L — число внутренних соединений в стиральной машине,
$X_1, Y_1, ..., X_L, Y_L$ — внутренние соединения в стиральной машине (числа в диапазоне от 1 до M, каждое i-е соединение соединяет точку с номером X_i с точкой с номером Y_i в электрической схеме стиральной машины),
P — число внутренних соединений в командоаппарате, $Z_1, U_1, ..., Z_P$, U_P — внутренние соединения в командоаппарате (числа в диапазоне от 1 до K, каждое i-е соединение соединяет точку с номером Z_i с точкой с номером U_i в электрический схеме командоаппарата),
T — длина циклограммы командоаппарата (число моментов времени в ней),
$\begin{matrix} C_{1,1} & \cdots & C_{1,Q} \\ \vdots & \ddots & \vdots \\ C_{T,1} & \cdots & C_{T,Q} \end{matrix}$ — циклограмма командоаппарата ($C_{i,j}$ равно 0 или 1, '0' означает, что в момент времени номер i контактная пара номер j [точки с номерами $N + 2j - 1$ и $N + 2j$ в электрической схеме командоаппарата] разомкнута [то есть эти точки соединены], '1' означает, что она замкнута).

Далее заданы T электрических схем, которые эта циклограмма должна обеспечивать. Каждая схема задается в следующем виде:
R_i — число соединений,

$V_1, W_1, ..., V_{R_i}, W_{R_i}$ — соединения (числа в диапазоне от 1 до R, каждое j-е соединение соединяет блок стиральной машины номер V_j с блоком номер W_j [то есть точку номер $N + V_j$ с точкой номер $N + W_j$ на электрической схеме стиральной машины], направление соединения не играет роли, если две точки соединены с третей, то они считаются соединенными

между собой, и т.д., точки, не соединяемые по этой схеме должны быть не соединены и командоаппаратом в соответствующий момент времени, все соедияемые по этой схеме точки должны быть каким-то способом соединены и командоаппаратом в соответствующий момент времени).

Выходные данные записываются в файл *output.txt* в следующем виде $S_1, ..., S_N$, где S_i — номер разъема стиральной машины, который следует соединить с i-м разъемом командоаппарата (S_i — числа в диапазоне от 1 до N). Если задача не имеет решения, выводится пустой файл *output.txt*.

Технические ограничения: числа M, K и T в файле *input.txt* не превосходят 200; L, P, R_i не превосходят 30000. Все числа в *input.txt* и *output.txt* разделяются пробелами или переводами строк, после последнего числа — конец файла. Время решения ограничено 20 секундами процессорного времени (для тактовой частоты процессора 200 МГц). Вспомогательные файлы использовать запрещается. Доступная оперативная память — 16M.

Пример

Пример правильного содержимого *input.txt*:

```
2 4
4 2 1 3 2 4 2 1 3 2 4 2 0 1 0 1 1 2
```

Здесь N=2, M=4, K=4, L=2, X_1=1, Y_1=3, X_2=2, Y_2=4, P=2, Z_1=1, U_1=3, Z_2=2, Z_2=4, T=2, Q=1, $C_{1,1}$=0, $C_{2,1}$=1, R_1=0, R_1=1, V_1=1, W_1=2.

Пример соответствующего содержимого *output.txt*:

```
 1 2
```

Этот файл задает такой способ присоединения командоаппарата, который обеспечивает разъединение блоков с номерами 1 и 2 в первый момент времени по циклограмме и соединение их во второй момент времени.

14.24. **Яблоневый сад.** У одного фермера есть квадратный участок земли 10000 метров в длину и 10000 метров в ширину. Это поле мысленно разлиновано на клетки с шагом в 1 метр. Линии пронумерованы от 0 до 10000. Таким образом, получилась координатная сетка.

Фермер посадил яблони точно в узлах (точках пересечения линий) сетки. Ни одна яблоня не посажена на краю участка. Теперь он решил поставить вокруг забор. Забор нужно поставить так, чтобы выполнялись следующие условия:

1) должен состоять из четырёх прямых пролётов одинаковой длины;
2) должен быть непрерывным;
3) углы должны находится точно в узлах сетки;
4) все яблони должны быть внутри области огороженной забором.

Желательно, чтобы забор был покороче.

Техническое требование

Расположение узла с деревом задаётся двумя числами, номерами линий, при пересечении которых получается этот узел. Расположение всех яблонь задается в файле *orchard.txt* следующим образом. В первой строке записано одно число – количество яблонь. Начиная со второй строки записаны пары чисел – расположение узлов. Пар столько же, сколько и яблонь. Числа разделены пробелами. Нужно написать программу, которая читает файл *orchard.txt* и не позже чем через 10 секунд записывает в файл fence.txt пары чисел — узлы, где должны быть углы забора.

Пример

В *orchard.txt*

```
3
 1  1
 5  6
30 30
```

В *fence.txt*

```
0  0
40 40
30 10
10 30
```

Оценка задачи

Оцениваться будет длина забора, чем короче, тем больше баллов. Среди программ будет проведено соревнование.

14.25. Приведите примеры современных систем программирования, в которых используются функциональные методы.

Глава 15

Моделирование

15.1. Прыгающий мяч. Под углом α к горизонту бросается мяч с начальной скоростью v_0; после каждого удара о землю его скорость уменьшается на p процентов. Смоделировать на экране траекторию полета мяча.

15.2. Математический и физический маятники. Дана длина нити маятника, ускорение свободного падения, начальная высота подъема маятника.

а) Смоделировать на экране движение математического маятника.

б) Учесть сопротивление внешней среды (считать его пропорциональным линейной скорости движения с коэффициентом пропорциональности $k \geqslant 0$), смоделировать движение физического маятника.

15.3. Снег кружится, летает, летает. Смоделировать на экране падающий снег. Интенсивность случайного появления снежинок определяется некоторой величиной. Каждая снежинка может перемещаться по экрану, делая случайные отклонения и, даже, подъемы вверх от налетевшего ветра. Средняя скорость снижения у каждой снежинки случайна.

15.4. Бильярд. Смоделировать движение бильярдных шаров. В начале определяется начальное расположение шаров (например, в виде пирамиды), сила и направление удара. При столкновении шаров или ударе о стенку теряется часть энергии, зависисящая от скорости. При движении учитывать трение. Шары могут попадать в лузы.

15.5. Погоня за зайцем. Заяц убегает от собаки по прямой l, начиная от точки A со скоростью v_1. Собака пытается догнать зайца, двигаясь от точки B все время в направлении зайца со скоростью v_2. Расстояние $AB = d$, точка B может принадлежать, а может и не принадлежать

прямой l. Смоделировать траектории движения зайца и собаки до того момента, когда собака поймает зайца или безнадежно отстанет.

15.6. Опишите *предметную область* задачи о выигрыше в лотерею.

15.7. Опишите *вычислительную модель* солнечных часов.

15.8. Является ли медицинский анализ инкубационного характера *имитационным моделированием*?

15.9. Что такое информационная система?

15.10. В чём проблема в известном парадоксе про Ахиллеса [21], который не может догнать черепаху?

15.11. В чём проблема в известном парадоксе про стрелу [21], которая никак не вылетит из лука?

15.12. Что такое *семафор*?

15.13. Что такое *системы массового обслуживания*?

15.14. Приведите пример системы массового обслуживания, которая не может быть смоделирована только семафорами?

15.15. Какие системы моделирования Вы знаете?

15.16. Промоделируйте распределение оперативной памяти на Вашем компьютере. Попробуйте определить, сколько памяти на самом деле необходимо для нормальной работы.

15.17. Мощность компьютера. Под *мощностью компьютера* иногда понимают то, как быстро он работает с разными программами. Говорят, что один компьютер мощнее другого. Смоделируйте программу, которая будет оценивать *мощность компьютера*.

15.18. Идеальный органайзер. Создать программный органайзер. Он должен уметь сигнализировать о наступлении события, как минимум, текстовым сообщением. События нужно уметь удобно планировать. Вот некоторые требования к планированию:

- однократное предупреждение о наступлении события;
- предупреждение о наступлении события через определённый интервал времени;
- периодическое предупреждение (ежедневно, еженедельно, каждый месяц и т.п.);

- отмена взаимосвязанных событий;
- автоматизированное планирование связанных событий.

15.19. Развитие популяции. Популяция некоторого биологического вида разбита на n возрастных групп. В первую группу попадают особи, возраст которых не более одного года, во вторую— особи возрастом от одного года до двух лет и так далее. Считается, что особи популяции живут не более n лет. Обозначим количество особей в каждой группе $p_i, i = 1, 2, \ldots, n$. Рождаемость в каждой из групп описывается коэффициентами $s_1, s_2, \ldots, s_n$, то есть прибавка количества новорожденных особей от i-ой группы равна $p_i s_i$. Выживаемость особей (то есть переход из группы в следующую группу в следующем году) описывается коэффициентами b_1, $b_2, \ldots,$ b_n $(0 \leqslant b_i \leqslant 1)$. То есть, если в текущем году численность i группы составляет p_i, то в следующем году число особей $i+1$ группы будет составлять $p_i b_i$. Так как особи живут не более n лет, то $b_n = 0$.

Смоделировать развитие популяции для данной численности особей по группам, коэффициентов смертности и выживания за m лет. Подобрать такие коэффициенты смертности и выживания, чтобы общая численность популяции $p = p_1 + p_2 + \cdots + p_n$ изменялась бы на протяжении большого числа лет минимально.

15.20. Борьба за существование. Два конкурирующих биологических вида живут на одной территории в постоянной борьбе за своё существование. Обозначим их как 'зайцы' и 'лисы'. Зайцы являются пищей для лис, а сами питаются дарами земли. Развите популяций наблюдается по дискретным периодам времени. Пусть a_0 и b_0 — численность зайцев и лис соответственно в начальном периоде времени. k_a, k_b — коэффициенты рождаемости (отношение количества родившихся особей в периоде времени к количеству выживших особей в предыдущем периоде). m_a, m_b — коэффициенты выживания (отношение количество выживших особей и перешедших из одного периода в следующий к количеству всех особей периода). Взаимное воздействие популяций друг на друга описывается коэффициентами: $n \geqslant 0$ — коэффициент пропорционального увеличения популяции лис в зависимости от роста популяции зайцев; $t \geqslant 0$ — коэффициент пропорционального уменьшения популяции зайцев в зависимости от роста популяции лис. Все коэффициенты положительны. Смоделировать развитие популяций в течение нескольких периодов времени или до полного исчезновения популяций. Изобразить развитие графически в виде кривой на координатной плоскости, где на оси абсцисс отмечена численность зайцев, а на оси ординат— численность лис.

15.21. Жизнь. Колония живых клеток развивается на бесконечном клетчатом поле. В каждой ячейке может находиться не более одной клетки. Развитие происходит периодами по следующим правилам: клетка выживает и переходит в следующий период, если она имеет двух или трех соседей из восьми возможных; если у клетки один сосед или нет соседей вовсе, то она погибает от изоляции; если соседей больше трех, то она погибает от перенаселения; в пустой позиции, у которой три живых соседа, рождается новая клетка. Написать программу, моделирующую развитие колонии клеток. Подобрать и исследовать такие начальные состояния колонии, при которых она бесконечно увеличивается; со временем погибает; циклически повторяет свое развитие.

15.22. Обобщенная жизнь. Обобщим задачу 15.21. Правила развития колонии клеток представим в виде двух функций: $f_0(i)$ и $f_1(i)$ для аргумента $i = 0, 1, \ldots 8$.

$$f_0(i) = \begin{cases} 0, & \text{если пустая ячейка, имеющая } i \text{ соседей,} \\ & \text{в следующем периоде остаётся пустой;} \\ 1, & \text{если в такой ячейке рождается живая клетка.} \end{cases}$$

$$f_1(i) = \begin{cases} 0, & \text{если живая клетка, имеющая } i \text{ соседей, погибает;} \\ 1, & \text{если такая живая клетка переходит в следующий период.} \end{cases}$$

Например, модель развития жизни из задачи 15.21 описывается следующими функциями: $f_0 = \{0, 0, 0, 1, 0, 0, 0, 0, 0\}$, $f_1 = \{0, 0, 1, 1, 0, 0, 0, 0, 0\}$. Написать программу, моделирующую развитие колонии клеток по определенным значениям функций $f_0(i)$ и $f_1(i)$. Как и в задаче 15.21, исследовать различные варианты развития колонии в зависимости от начальных состояний колонии и функций развития.

15.23. Один шаг назад... Даны состояние колонии клеток и функции развития колонии (см. задачу 15.22). Найти одно из возможных состояний колонии, непосредственно предшествующих данному, причем количество живых клеток в этом состоянии было бы минимальным.

15.24. Жуки. На координатной плоскости в точках с координатами (x_1, y_1), $(x_2, y_2), \ldots,$ (x_n, y_n) в начальный момент времени находятся n жуков. Каждый жук с номером i гонится за жуком с номером $i + 1$ со скоростью v_i $(i = 1, 2, \ldots, n)$; жук номер n гонится за первым жуком. Модель рассматривается в течение последовательных моментов времени. Каждый из жуков передвигается в том направлении, где находился преследуемый жук в предыдущий момент времени. Если расстояние между

жуками в какой-то момент времени становится меньше величины δ, то догонящий жук съедает преследуемого и затем начинает преследовать того жука, за которым гнался съеденный. Когда остаются только два жука, то побеждает тот из них, номер которого больше.

Смоделировать движение жуков до k-го момента времени или до тех пор, когда догонять уже некому.

15.25. Круговой перекрёсток. К перекрёстку с круговым движением подъезжают автомобили по четырем подходящим дорогам, образующим крест. Каждому автомобилю нужно повернуть на одну из дорог, исключая ту, по которой он подъехал. Движение по кругу осуществляется по одной полосе против часовой стрелки, движение по подъезным дорогам осуществляется в две полосы в противоположных направлениях. Если полоса кругового движения на въезде с дороги занята , то автомобиль ожидает своей очереди въезда в круг. Скорости всех автомобилей одинаковы и равны v, радиус круга r, длина автомобиля (то есть линейное пространство, которое занимает автомобиль на дороге) равна $l (l \ll r)$. Время появления нового автомобиля на одном из четырех подъездов — величина случайная, с равномерным распределением; например, можно считать, что в среднем на каждом из подъездов в единицу времени появляется $k > 0$ автомобилей, причём k — величина совсем не обязятельно целая. Смоделировать движение автотранспорта на перекрёстке. Подобрать оптимальную (наименьшую допустимую) скорость движения по кольцу v, чтобы при данных параметрах r, l и k наибольшая длина очереди на каждой из подъездных дорог не превышала бы m автомобилей.

15.26. UML. Охарактеризуйте кратко UML.

15.27. RUP. Что такое RUP?

15.28. Нейпьютер. Дискретным пороговым элементом будем называть устройство, реализующее многоместную логическую функцию f (от логических аргументов — 0 и 1) вида: $f(x[1], ..., x[n]) = 0$, если $x[1] + ... + x[n] < k$, иначе $f(x[1], ..., x[n]) = 1$, где k — натуральное число. Число n называется числом входов элемента, k — его порогом.

Требуется, чтобы программа строила схему из пороговых элементов для вычисления заданной логической функции.

Формат входных данных (в файле *input.txt*):

```
<число аргументов функции> <значения функции>
```

Значения функции выписываются для всех допустимых последовательностей аргументов в порядке возрастания соответствующих двоичных чисел. Разделители — пробелы и переводы строк.

Формат выходных данных (в файле *output.txt*):

```
N <число входов элемента> <порог> <связи> ... <число входов
элемента> <порог> <связи>
```

Здесь N — число элементов в списке; связи — номера элементов, с которыми соединены соответствующие входы. Элементы списка считаются пронумерованными с $N+1$. Номера от 1 до N соответствуют входам всей схемы — номерам аргументов вычисляемой функции. Последний элемент вычисляет требуемую функцию.

Если решения нет, входной файл содержит одно число -1.

Время решения — 20 секунд.

Оценка

Лучшей считается программа, **правильно** обрабатывающая функции с как можно большим числом аргументов. Среди программ, правильно обрабатывающих функции с одинаковым числом аргументов лучшей считается программа, выдающая в среднем схемы меньшего размера, а при совпадении и этого параметра — схемы с меньшим числом входов у элементов.

Пример

input.txt

```
2 0111
```

output.txt

```
1 2  1  1  2
```

15.29. **Корова на привязи.** Оригинальную привязь придумал житель одной из деревень для выпаса своей коровы. Привязь состоит из двух длинных кольев и верёвки. Веревка связана в кольцо, а колья вбиваются внутри этого кольца. К этой верёвке и привязана корова. Таким образом корова может свободно есть траву на ограниченной площади.

Техническое требование

Толщиной кольев и верёвки пренебречь. Траву, которую корова сможет съесть за пределами кольца, не считаем. В этих условиях требуется вычислить площадь, которую сможет объесть корова для заданных координат кольев и длины верёвки.

Координаты кольев (в сантиметрах) — целые числа от -1000 до 1000. Длина веревки (в сантиметрах) — число от 1 до 10000.

Площадь вычислять с точностью до 1 квадратного миллиметра.

Время работы программы не более 3 секунд.

Пример

```
Координаты первого кола: 10 10
Координаты второго кола: -10 -10
Длина верёвки:  100
Площадь: 3711
```

15.30. **Оптимизация устройств.** Имеется N устройств, каждое может выполнять некоторый набор операций (для разных устройств разный), при этом набор выполняемых подряд операций имеет суммарную стоимость 1. Задание на выполнение состоит из упорядоченного пронумерованного списка операций, для каждой из которых указаны номера предшествующих операций (они должны обязательно быть выполнены до данной).

Задача: найти допустимую перестановку операций и привязку её к устройствам, при которой суммарная стоимость выполнения минимальна.

Техническое требование

На входе даны два файла *unit.dat* — файл, в каждой строке r которого перечислены через пробел коды допустимых (выполняемых) операций для устройства r, и файл *op.dat* — в каждой строке t первое число — код операции, а следующие числа — номера предшествующих операций.

На выходе должен быть файл *trans.dat* с перестановкой для *op.dat* — в каждой строке i число j — поместить в i строку результата j строку файла *op.dat*.

Пример

unit.dat

```
1 2 3
3
```

op.dat

```
1
1 1
3 2
2 1
```

trans.dat

```
1
2
4
3
```

15.31. Редактор блок-схем. Описать модель и написать программу для работы с блок-схемами. Программа должна предоставлять следующие возможности:

- рисовать;
- сохранять и продолжать при следующем запуске;
- изменять расположение блоков при автоматическом сохранении связей.

Ограничения на сложность блок-схем определите сами.

15.32. Редактор схем Данке. Наряду с блок-схемами есть ещё один способ «записывать» алгоритмы схематически — *схемы Данке*.

Схемы Данке, как и блок-схемы состоят из блоков и соединяющих их линий. Все отличие состоит в исполнителе. Для исполнителя схем Данке линии — это коридоры, а блоки — комнаты. По линиям и комнатам исполнитель всегда идет вдоль левой (для него) стены, значит на разветвлениях он сворачивает налево. Дойдя до конца линии исполнитель, как в обычном коридоре, поворачивает и идёт в обратном направлении вдоль противоположной стены.

Для выполнения блоков необходимо блок соединить линией, что даст возможность исполнителю свернуть к нужному блоку. Такая возможность даёт большее, по сравнению с блок-схемами, удобство исправления. Например, можно «перенести» часть алгоритма, перерисовав только одну линию.

В схемах Данке используют три вида блоков:

- прямоугольные, для простых действий;
- ромбы, для условий;
- растянутые по горизонтали шестиугольники, для повторений.

Прямоугольники исполнитель проходит, как коридор, вдоль левой стенки, но при этом выполняет указанное действие.

Ромбы имеют один вход и два выхода. Над один из выходов помечен знаком '+', другой — знаком '−'. Походя по такой комнате исполнитель проверяет условие и если оно верно выходит в коридор помеченный '+', иначе — в коридор помеченный '−'. Если кроме коридора ничего нет, его можно не рисовать. На обратном пути исполнитель проходит ромбические блоки без вычислений.

Шестиугольники имеют один вход и один выход. В блоке написано условие, при нарушении которого исполнителя выпустят из комнаты обратно во вход. До тех пор, пока условие верно исполнитель будет заходить в выход проделывать указанные действия, возвращаться.

Получается, что в комнатах есть двери, которые открываются и закрываются, взависимости от условий.

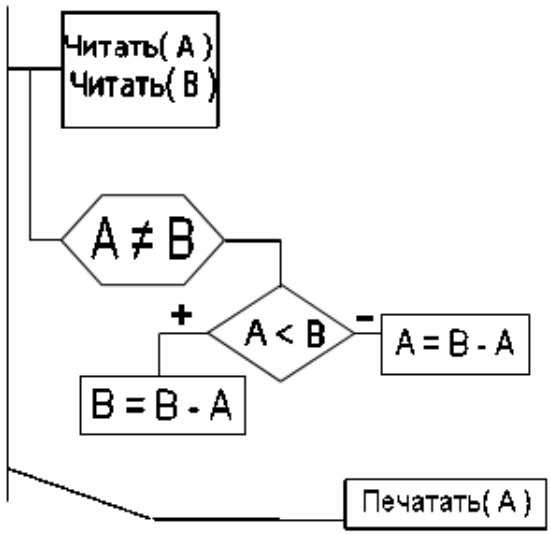


Рис. 15.1: Схема Данке алгоритма Евклида

Обычно начинают с того, что проводят вертикальную линию и считается, что исполнитель идёт по ней сверху вниз. Заканчивается работа алгоритма при выходе исполнителя из этой линии вверх.

Рисунок 15.1 на странице 203 показывает алгоритм Евклида в виде схемы Данке.

Задача. Описать модель и написать программу для работы с схемами Данке. Программа должна предоставлять следующие возможности:

- рисовать;
- сохранять и продолжать при следующем запуске;
- изменять расположение блоков при автоматическом сохранении связей.

Ограничения на сложность схем Данке определите сами.

15.33. Модель. Дайте определение понятия модели.

15.34. Модель и реальность. Сравните модели и реальность. Почему нужно использовать модели и в чём опасность их использования?

15.35. Аналог и цифра. В чём различие между аналоговым и цифровым представлениями?

Глава 16

Задачи, решаемые различными методами

16.1. Нарисовать во весь экран:

```
+--+--+--+--+--+--+--+--+--+--+
|  |  |  |  |  |  |  |  |  |  |
+--+--+--+--+--+--+--+--+--+--+
|  |  |  |  |  |  |  |  |  |  |
+--+--+--+--+--+--+--+--+--+--+
|  |  |  |  |  |  |  |  |  |  |
+--+--+--+--+--+--+--+--+--+--+
```

16.2. Ряды чисел. Для заданной высоты нарисовать без разрывов цепочки натуральных чисел от 1 до 20 включительно.

Пример

```
Высота: 3
12345678910111213141516171819 20
12345678910111213141516171819 20
12345678910111213141516171819 20
```

16.3. Сосиска в тесте. Сосиска в тесте готовится из одной сосиски и 75 грамм теста. Определить, сколько, максимум, можно сделать целых сосисок в тесте, если есть X сосисок и Y килограммов теста.

Пример

```
X = 100
Y = 2
Получится сосисок: 26
```

16.4. Проверка на число. Задана строка. Длина строки не более 100. Определить, является ли эта строка записью натурального числа.
Напоминание: числа могут содержать незначащие нули.

Пример

```
Строка: 00112233445566778899
Да, эта строка - натуральное число.
```

16.5. Написать программу, которая читает значение строки A и преобразует его как указано ниже, а затем выводит преобразованное A.

1. Если A — чётное число, то делит на 2, иначе не меняет.
2. Число больше 100 уменьшает на 50, иначе не меняет.
3. Число кратное 10 уменьшает в 5 раз, иначе не меняет.
4. Заменит пятый символ на '_'. Если такого нет, ничего не менять.
5. Заменит последний символ на '.'. Если такого нет, ничего не менять.

16.6. Вывести средние символы строки, если они есть. Иногда их два.

Пример

```
Строка: пример
им
```

16.7. Самое большое число. Прочитать N чисел и вывести самое большое из них.

Пример

```
10 -1 91 -1 15
Результат: 91
```

16.8. Сумма двух квадратов. Даны натуральные числа a и b ($a \leqslant b \leqslant 10^6$). Найти количество натуральных чисел из отрезка $[a; b]$, которые можно представить в виде суммы квадратов двух натуральных чисел по крайней мере двумя различными способами (две суммы, состоящие из одинаковых слагаемых, пусть даже с разным их порядком, считаются одинаковыми).

Пример

```
a=1
b=100
Ответ=3
```

16.9. Сумма квадратов. Дано натуральное число a. Определить, можно ли его представить в виде суммы квадратов натуральных чисел $a = {a_1}^2 + {a_2}^2 + \cdots + {a_n}^2$ так, чтобы слагаемые этой суммы не повторялись; порядок слагаемых не важен. Если можно — то выдать найденную

сумму; в случае, если таких способов несколько, то выдать любую сумму с минимальным числом слагаемых.

16.10. Двумя способами. Дано натуральное число n. Найти наименьшее число s, которое может быть представлено в виде суммы $s = a^n + b^n$ (a и b натуральные) по крайней мере двумя нетривиально различными способами.

16.11. Удаление символов. Можно ли из одной строки получить другую, убрав некоторые символы?

Пример

программа
гамма
Результат: ДА

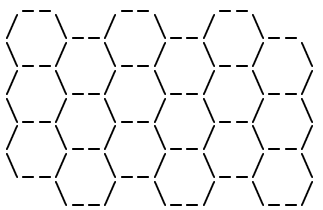
16.12. Барабан. Даны n чисел $a_1, a_2, \ldots, a_n$, записанные по окружности. Вектором x_k ($1 \leqslant k \leqslant n$) будем называть последовательность $a_k, a_{k+1}, \ldots, a_n, a_1, \ldots, a_{k-1}$. Будем считать, что вектор x_k меньше вектора x_m, если в первой неравной паре соответствующих элементов этих векторов выполняется условие $a_{k+i} < a_{m+i} (i = 0, 1, \ldots)$. Среди векторов $x_1, x_2, \ldots, x_n$ найти минимальный.

16.13. ✣ **Соты.** Нарисовать соты заданного размера. Как они устроены, догадайтесь сами.

Техническое требование Программа должна запрашивать ширину и высоту сот и выводить на экран изображение сот. Изображение возможно как текстовое, так и графическое. Весь рисунок должен полностью входить на экран. Максимальное значение высоты — 11 , ширины — 26,

Пример

Ширина: *6*
Высота: *3*



16.14. Конечная последовательность $a_1, a_2, \ldots, a_n$ состоит из цифр 0, 1 и 2. Участком последовательности назовём любые k подряд идущих членов последовательности ($1 \leqslant k \leqslant n$). Два участка называются смежными, если сразу после последнего члена одного участка идет первый член другого участка.

Дано натуральное n ($2 \leqslant n \leqslant 1000$). Найти какую-нибудь последовательность цифр 0, 1 и 2, чтобы в ней не было бы одинаковых смежных участков.

Пример

```
n=6
Ответ: 012010
```

16.15. ✣ **ТОСКА.** Заданы три слова. Последняя буква первого слова такая же, как первая буква второго, последняя буква второго слова – как первая буква третьего.

Задача.

Придумать четвёртое слово, первая буква которого будет такая же, как последняя буква третьего слова, а последняя буква — такая же, как первая первого.

Техническое требование

Все четыре слова — слова русского языка одинаковой длины, существительные, нарицательные, единственного числа, именительного падежа, присутствующие в каком-нибудь словаре русского языка (бумажном).

Длины слов будут 5 букв.

Первые и последние буквы будут только т, о, с, к, а.

Выдать ответ в виде квадрата из этих слов. То есть, нужно первое слово записать вертикально сверху вниз справа, второе внизу справа налево, третье вертикально снизу вверх, а четвёртое (Ваше) вверху слева направо. Первые и последние буквы должны стоять на пересечении. Смотрите пример.

Проверять входные слова на соответствие условию не нужно.

Пример

```
Слово 1: тоска
Слово 2: астра
Слово 3: астра
Ответ:
атлет
р   о
т   с
с   к
артса
```

16.16. Такая функция. Определим функцию $f(x)$ для целого неотрицательного аргумента x: $f(0) = 0, f(1) = 1, f(2n) = n, f(2n+1) = f(n) + f(n+1)$. Дано $N (0 \leqslant N \leqslant 10^9)$. Найти $f(N)$.

Ограничение: рекурсией и массивами не пользоваться.

16.17. Точки на прямой. Дано натуральное число n ($2 \leqslant n \leqslant 1000$) и для некоторых различных пар натуральных чисел (i, j) заданы вещественные числа a_{ij}, $(i, j \leqslant n, i \neq j)$. Можно ли на числовой прямой выбрать n точек t_i, $(1 \leqslant i \leqslant n)$ так, чтобы расстояния между различными точками t_i и t_j были равны заданному a_{ij}?

Техническое требование

В первой строке входного файла указано количество точек n. В остальных строках находятся тройки чисел: номер первой точки, номер второй точки, расстояние между ними. Номера точек — натуральные числа, не превосходящие n; расстояние задается вещественным неотрицательным числом, не большим 1000. Входные данные корректны, то есть соответствуют ограничениям задачи. Программа выдает в выходной файл слово «да», если точки можно расположить на числовой прямой, в противном случае выдается слово «нет». Точность вычислений равна $\varepsilon = 10^{-3}$.

16.18. Точки на плоскости. Дано натуральное число n ($2 \leqslant n \leqslant 1000$) и для некоторых различных пар натуральных чисел (i, j) заданы вещественные числа a_{ij}, $(i, j \leqslant n, i \neq j)$. Можно ли на координатной плоскости выбрать n точек t_i, $(1 \leqslant i \leqslant n)$ так, чтобы расстояния между различными точками t_i и t_j были равны заданному a_{ij}?

Техническое требование

В первой строке входного файла указано количество точек n. В остальных строках находятся тройки чисел: номер первой точки, номер второй точки, расстояние между ними. Номера точек — натуральные числа, не превосходящие n; расстояние задается вещественным неотрицательным числом, не большим 1000. Входные данные корректны, то есть соответствуют ограничениям задачи. Программа выдает в выходной файл слово «да», если точки можно расположить на плоскости, в противном случае выдается слово «нет». Точность вычислений равна $\varepsilon = 10^{-3}$.

16.19. Системы счисления. Множество цифр системы счисления с основанием R ($2 \leqslant R \leqslant 36$) содержит цифры $0, 1, \ldots 9$; если арабских цифр до R не хватает, то в множество добавляются прописные буквы латинского алфавита $A, B, C, \ldots, Z$. Даны две строки символов. Если эти строки могут являться записями некоторого числа в системах счисления с основаниями R_1 и R_2 соответственно, то найти минимальные значения R_1 и R_2; если таких оснований не существует, то выдать число 0.

16.20. Обращение. Обращением натурального числа n назовём такое

натуральное число $n_{\text{обр}}$, что запись числа $n_{\text{обр}}$ в двоичной системе счисления получается переворотом двоичной записи числа n. Например, для $n = 22_{(10)} = 10110_{(2)}$ число $n_{\text{обр}}$ равно $13_{(10)} = 1101_{(2)}$.

Даны два натуральных числа a и b ($a \leqslant b \leqslant 10^9$). Найти наименьшее и наибольшее из обращений натуральных чисел отрезка $[a; b]$.

16.21. Прямые. Даны натуральное число n, действительные числа $a_1, b_1, c_1, a_2, b_2, c_2, \ldots, a_n, b_n, c_n$. Найти количество областей плоскости, на которые её разбивают прямые, задаваемые уравнениями $a_i x + b_i y + c_i = 0, i = 1, 2, \ldots, n$.

16.22. Домино. Имеется n костей домино, возможно, некоторые кости одинаковы. Найти длину самой длинной цепочки, выстроенной из этих костей по правилам домино.

Техническое требование

В первой строке входного файла указано количество костей n ($1 \leqslant n \leqslant 100$). В каждой из следующих n строк находится пара целых чисел, разделенных пробелом: значения очередной кости a_i, b_i, где $0 \leqslant a_i, b_i \leqslant 6$. В выходной файл нужно поместить одно натуральное число, равное длине наидлиннейшей цепочки костей в соответствии с условиями задачи.

16.23. Удалить числа. Дана последовательность целых чисел a_1, a_2, $\ldots$, a_n, ($n \leqslant 20$). Найти минимальное количество чисел, после удаления которых из последовательности оставшиеся числа образуют возрастающую последовательность.

16.24. Найти прогрессию. Дан набор целых чисел $\{a_1, a_2, \ldots, a_n\}$, ($n \leqslant 20$). Построить из элементов набора наиболее длинную последовательность, которая является

а) арифметической прогрессией;

б) геометрической прогрессией.

16.25. Тождественность выражений. Заданы записи двух арифметических выражений с целыми числами, операциями сложения, вычитания, умножения, деления нацело, круглыми скобками и переменными. Написать программу, которая проверяет тождественность выражений, то есть совпадают ли значения выражений для всех возможных значений переменных.

16.26. Тавтология. Дана запись логического выражения в соответствии с синтаксисом:

```
<лвыраж> ::= <лтерм>|<лтерм><оп><лвыраж>
<лтерм> ::= <лприм>|<лприм> & <лтерм>
<оп>::=v|x
<лприм> ::= T|F|<имя>|(<лвыраж>)
```

где <имя> – правильно построенный идентификатор (переменная). Проверить, не является ли данное выражение тавтологией, когда для всех возможных логических значений переменных значение выражения всегда истинно.

16.27. Оптимальное размещение. На клетчатом прямоугольном поле размера $N \times M$ $(N, M \leqslant 16)$ задано несколько фигур. Каждая фигура представляет собой множество смежных клеток, причем каждая клетка фигуры соприкасаются по крайней мере одной из своих сторон с одной из соседних клеток этой же фигуры. Фигуры друг с другом не пересекаются (то есть у них нет общих клеток). Требуется указать размеры минимального по площади прямоугольника, в котором можно разместить заданные фигуры без пересечений. Разрешается их поворачивать на угол, кратный $\frac{\pi}{2}$.

Техническое требование Структура входного файла input.txt следующая:

```
<РазмерПоляN><РазмерПоляM>
<a11><a12>...<a1M>
...
<aN1><aN2>...<aNM>
```

где aij — номер фигуры (натуральное число), которой принадлежит клетка поля с координатами i и j, либо ноль, если клетка не принадлежит ни одной фигуре. Считать, что входные данные корректны. В выходной текстовый файл output.txt программа записывает одно число — площадь искомого прямоугольника (количество клеток в нём).

Пример

Входной файл:

```
4 7
0 0 1 0 0 0 0
1 0 1 2 2 2 2
1 0 1 0 0 0 2
1 1 1 0 2 2 2
```

Выходной файл: 16

16.28. Факториальное представление. Факториальным представлением целого числа N назовем набор целых значений $(d_n, d_{n-1}, \ldots, d_2, d_1)$, для которого

$$N = d_n n! + d_{n-1}(n-1)! + \cdots + d_2 2! + d_1 1!$$

Дано факториальное представление целого число N. Найти N.

Пример

```
Вход: -1 0 2 1
Выход: -19
```

16.29. Найти факториальное представление. Решить задачу, обратную 16.28. Дано целое число N. Требуется найти одно из его факториальных представлений в соответствии с одним из следующих ограничений или сообщить, что это невозможно:

а) количество ненулевых коэффициентов в представлении должно быть не менее данного натурального k;

б) $|d_i| \leqslant a$, где $1 \leqslant i \leqslant n$, $a > 0$ — данное целое число;

16.30. Факториальная система счисления. Представлением натурального числа N в факториальной системе счисления называется набор целых значений $(d_n, d_{n-1}, \ldots, d_2, d_1)$, для которого

$$N = d_n n! + d_{n-1}(n-1)! + \cdots + d_2 2! + d_1 1!, \text{ где } d_i \leqslant i,\ i = 1, 2, \ldots, n.$$

Дано натуральное число N. Найти его представление в факториальной системе счисления.

16.31. Фибоначчиева система счисления Записью натурального числа N в фибоначчиевой системе счисления называется последовательность двоичных цифр $a_n a_{n-1} \ldots a_{2(\Phi)}$ ($a_i = 0$ или $a_i = 1$), такая, что

$$N = a_n f_n + a_{n-1} f_{n-1} + \cdots + a_2 f_2,$$

где $f_2, \ldots, f_{n-1}, f_n$ — числа Фибоначчи ($f_1 = f_2 = 1, f_i = f_{i-1} + f_{i-2}, i \geqslant 3$), а f_n — наибольшее из не превосходящих N чисел Фибоначчи. Например, $30_{(10)} = 1010001_{(\Phi)}$, так как $30 = 1 \cdot 21 + 0 \cdot 13 + 1 \cdot 8 + 0 \cdot 5 + 0 \cdot 3 + 0 \cdot 2 + 1 \cdot 1$. Дано натуральное N. Найти его запись в фибоначчиевой системе счисления.

16.32. Снова фибоначчиева система счисления. Не всякая последовательность нулей и единиц может являться записью натурального числа в фибоначчиевой системе счисления (см. задачу 16.31) Например, последовательность 110 не является записью числа в фибоначчиевой системе счисления, так как число $5 = 1 \cdot 3 + 1 \cdot 2 + 0 \cdot 1$ записывается иначе:

$5 = 1\cdot5+0\cdot3+0\cdot2+0\cdot1$, то есть $5 = 1000_{(\Phi)}$ (смотрите интересную книгу о числах Фибоначчи [4]). Дано натуральное M. Найти количество фибоначчиевых записей натуральных чисел среди всевозможных последовательностей, состоящих из M двоичных цифр (нулей и единиц). Ведущие незначащие нули можно отбрасывать.

16.33. Задача о прыгуне. Прыгун может прыгать в одном направлении по разделенной на клетки полосе. За каждый прыжок он может переместиться либо в соседнюю клетку, либо через одну клетку. Изначально прыгун находится в клетке с номером 1. Сколькими различными способами он может допрыгать до клетки с номером n? Способы прыгания считаются одинаковыми, если в ходе каждого из них прыгун побывал в одних и тех же клетках.

16.34. Тройной прыжок разрешен! Решить задачу 16.33 в предположении, что за каждый прыжок прыгун может переместиться в соседнюю клетку, либо через одну клетку, либо через две клетки.

16.35. Латинский квадрат. Дана целочисленная квадратная матрица A порядка n. Проверить, является ли эта матрица латинским квадратом, то есть такой, в которой каждая строка и каждый столбец содержат все числа от 1 до n.

16.36. Магический квадрат. Дана целочисленная квадратная матрица A порядка n. Проверить, является ли эта матрица магическим квадратом, то есть такой, в которой суммы элементов в каждой строке, каждом столбце и в каждой из двух диагоналей совпадают.

16.37. Угадай число. Известна игра, когда один человек загадывает целое число от 1 до 100, а другой должен его отгадать. Отгадывающий должен называть числа, а загадывающий сообщать «больше», если загаданное число больше и «меньше», если — меньше. Ответ «равно» будет последним. Нам известны все числа, которые были загаданы и в том же порядке, кроме последнего. Мы знаем, что чисел было ровно 10, но мы знаем только 9 первых. И самое главное, загадывающий отвечал так: «больше», «меньше», «больше», «меньше», «больше», «меньше», «больше», «меньше», «больше», «меньше», «равно». Наша задача определить, какое число было загадано.

Техническое требование

Варианты ответов: либо выдать число, либо — «Только угадать», либо — «Так не может быть».

Пример

```
 1
50
25
40
30
39
32
34
31
Было загадано: 33
```

16.38. Пирожное «Семейное». Пирожное «Семейное» это три разных пирожных круглой формы в одной коробке. Нужно для каждого «семейного» пирожного (для заданных размеров трёх пирожных) определить размер коробки. Требования главного кондитера

Коробка должна быть квадратной. Все пирожные должны полностью стоять на дне коробки. Они могут соприкасаться, но не лежать друг на друге. Коробка должна быть минимального размера.

Задача. По заданным с сантиметрах диаметрам пирожных, сообщить длину стороны коробки с точностью до миллиметра.

16.39. Значение интеграла. Вычислить $f(y) = \int\limits_0^y x^x\,dx$

Техническое требование

Для заданного y вывести значение $f(y)$.

Пример

```
y= 0
f(0)=0
```

16.40. ✣ Кривая площадь. Из прямоугольного листа размерами 100×200 см вырезают окружность (или только часть) радиусом 50 см. Это делают с помощью пресса. Для указания места для отверстия вводят целые координаты центра вырезаемой окружности. Начало координат находится в левом нижнем углу прямоугольного листа. Ось OY направлена вверх, ось OX направлена вправо.

Задача

Написать программу, которая запрашивает координаты для пресса и определяет площадь фигуры, получающейся после срабатавания пресса.

Техническое требование

Ответ нужно получить с точностью до 1 см^2 в целых единицах. Координаты — целые неотрицательные числа до 1000.

Пример

```
Координаты: 0    50
Площадь: 16073
```

16.41. Двоичный палиндром. Целое число называется двоичным палиндромом, если его запись в двоичной системе счисления является палиндромом (в запись допускается добавлять нужное количество незначащих нулей). Даны целые числа N и M ($0 \leqslant N \leqslant M \leqslant 50000$), записанные в десятичной системе счисления. Найти количество двоичных палиндромов, принадлежащих отрезку $[N; M]$.

16.42. Китайская теорема об остатках. Даны натуральные взаимно простые числа m_1, m_2, ..., m_n, целые неотрицательные числа r_1, r_2, ..., r_n ($m_i > r_i$ для каждого $i \in [1; n]$). Согласно китайской теореме об остатках, существует единственное число X, такое что $0 \leqslant X < m_1 \cdot m_2 \cdot \ldots \cdot m_n$ и для каждого целого $i \in [1; n]$ остаток от деления X на m_i равен r_i. Требуется по наборам чисел $\{m_i\}_{i=1}^n$ и $\{r_i\}_{i=1}^n$ найти число X.

16.43. Игра с суммой. Квадратная таблица, в ячейках которой располагаются целые числа, имеет размеры $N \times N$, где N– нечетно. В центральной клетке таблицы стоит число 0, сумма всех чисел равна нулю. В центре находится фишка. Играют два игрока, поочередно делая ходы. За один ход игрок может переместить фишку из ячейки в любую другую соседнюю ячейку, в которую еще не было сделано ранее ходов (соседней ячейкой называется любая из не более чем восьми смежных ячеек). После хода в ячейку она закрывается и в неё больше ходить нельзя. При каждом ходе вычисляется сумма игры s, то есть число ячейки прибавляется к s. Игра заканчивается, когда нельзя сделать ни одного хода; ходы игрокам пропускать нельзя. Если в результате игры сумма s получилась отрицательной, то первый игрок выплачивает второму $-s$ рублей, если положительной, то второй игрок первому выплачивает s рублей.

Написать программу, играющую за одного из игроков по данной таблице чисел.

16.44. Даны натуральные n и m ($m \leqslant n$). Получить все последовательности, состоящие из n двоичных цифр (нулей и единиц), содержащие ровно m единиц.

16.45. Размещения. Даны натуральные n и m ($m \leqslant n$). Получить все размещения, состоящие из натуральных чисел из отрезка $[1; n]$, по m элементов в каждом размещении. Размещение — это множество из m чисел, взятых в определенной последовательности.

16.46. Даны вещественные числа $a_1, a_2, \ldots, a_n (a_i \neq 0, 1 \leqslant i \leqslant n)$. Найти набор индексов $i_1, i_2, \ldots, i_k (1 \leqslant i_1 \leqslant i_2 \leqslant \cdots \leqslant i_k \leqslant n)$ такой, чтобы сумма $a_{i_1} + a_{i_2} + \cdots + a_{i_k}$ принимала бы наименьшее по модулю значение из всех подобных сумм.

16.47. НОД чисел Фибоначчи. Даны натуральные m и n $(m \leqslant n \leqslant 2^{15})$. Найти число различных пар чисел Фибоначчи (f_i, f_j), где $m \leqslant i \leqslant j \leqslant n$, таких, что НОД$(f_i, f_j)$ — число Фибоначчи.

16.48. Требуется налить. Даны n сосудов, имеющих объемы v_1, v_2, …, v_n, где v_i — целые числа, обозначающие литры. Ответить на вопрос, можно ли с помощью указанных сосудов отмерить ровно v литров жидкости?

16.49. Требуется нарезать. Дана непустая последовательность, состоящая из двоичных цифр (длина последовательности $l \leqslant 255$) и дано натуральное число $n (n \leqslant l)$. Разбить последовательность на n непустых групп последовательных цифр так, чтобы сумма чисел, двоичная запись которых состоит из последовательности цифр групп, была бы

а) минимальной;

б) максимальной.

16.50. Посыпались. На экран компьютера выведено n-значное натуральное число, записанное в десятичной системе счисления. В результате действия вируса некоторые цифры числа 'посыпались', то есть выпали из числа, а вместо них остались пустые знакоместа. Выпавшие цифры остались на экране, но в виде неупорядоченной кучи. Сколько различных значений можно получить, вставляя выпавшие цифры обратно в число? Вставлять в одно знакоместо можно только одну цифру.

Техническое требование

На входе две строки: в первой задан шаблон числа, в котором места выпавших цифр обозначены знаками '?', длина шаблона не превосходит 12; во второй строке дан список выпавших цифр (без пробелов или других знаков), причем количество знаков '?' в шаблоне совпадает с количеством цифр во второй строке. Выдать целое число, равное количеству различных значений, которые возможно получить вставкой выпавших цифр в число вместо знаков '?'.

16.51. Луч света. На координатной плоскости задано n прямоугольников со сторонами, параллельными или перпендикулярными осям координат. Кроме этого, на плоскости находится светочувствительная фотопластинка, задаваемая в виде отрезка, концы которого имеют коор-

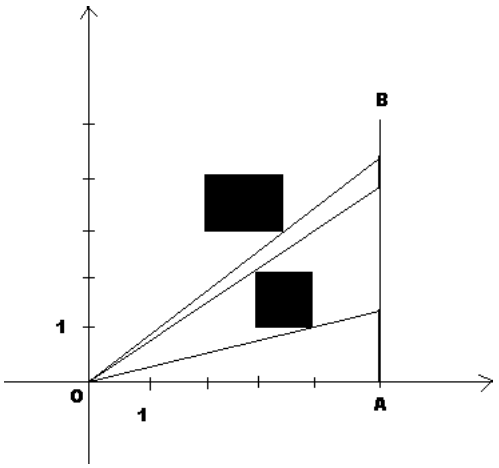


Рис. 16.1: К задаче 16.51.

динаты (x_1, y_1) и (x_2, y_2). Отрезок не пересекается ни с одним из прямоугольников; концы отрезка на совпадают. В начале координат помещен источник света. Стороны прямоугольников являются препятствием для света. Найти, какая часть длины отрезка-фотопластинки будет засвечена? Пример расположения двух прямоугольников и пластинки AB приведен на рис. 16.1.

Техническое требование

Во входном файле задаются: в первой строке — количество прямоугольников n, в следующих n строках параметры прямоугольников в виде четверок действительных чисел, первые два числа — координаты центра прямоугольника, следующие два — соответственно длина (измерение по оси OX) и ширина (измерение по оси OY). В последней строке находятся координаты концов отрезка-фотопластинки. В выходной файл записывается действительное число с тремя точными десятичными знаками, обозначающее отношение засвеченной части пластинки к размеру (длине отрезка) всей пластинки.

Пример

Входной файл:

```
2
3.5 1.5 1 1
2.75 3.5 1.5 1
5 0 5 5
```

Выходной файл:

```
0.440
```

16.52. Даны три натуральных числа a, b, c $(1 \leqslant a \leqslant b \leqslant 10^9, 1 \leqslant c \leqslant$

10). Найти количество всех чисел z, принадлежащих отрезку $[a;b]$, таких, что они могут быть представлены в виде $z = z_n c^n + z_{n-1} c^{n-1} + \cdots + z_1 c + z_0$, где $z_i \in \{0, 1\}, n \geqslant 0$.

16.53. Перевод в двоичную систему. Дано натуральное N, записанное в шестнадцатеричной системе счисления $(1 \leqslant N \leqslant 2^{8000} - 1)$. Найти количество единиц в двоичной записи этого числа.

16.54. Покрытие прямоугольника.Дан прямоугольник с размерами $a \times b$ и n кругов с радиусами $r_1, r_2, \ldots r_n$. Какое наименьшее количество из данных кругов надо взять, чтобы ими полностью покрыть прямоугольник?

16.55. Квадраты.На координатной плоскости заданы n квадратов, стороны которых параллельны осям. Требуется найти площадь области, покрытой квадратами.

Техническое требование . В первой строке входного файла находится целое число n $(0 \leqslant n \leqslant 100)$ — количество квадратов. В каждой из n следующих строк находится тройка чисел x, y, a, где x, y — координаты левого нижнего угла квадрата, a — длина его стороны; все числа целые, $-30000 \leqslant x, y \leqslant 30000$, $0 \leqslant a \leqslant 1000$. В выходной файл необходимо записать площадь области объединения квадратов в виде вещественного числа с точностью до двух знаков после запятой.

16.56. Одна строка. Дан текстовый файл, состоящий из n строк $(1 \leqslant n \leqslant 50000)$, длина каждой строки не превосходит 255 символов. В файле есть ровно одна строка, которая встречается ровно один раз, все остальные строки повторяются, причем обязательно четное количество раз. Найти эту уникальную строку.

16.57. Еще раз одна строка.Решить задачу 16.56, отменив ограничение четности вхождений повторяющихся строк, то есть только одна строка появляется единожды, другие повторяются два или более раз.

16.58. Даны натуральные m, n $(2 \leqslant m \leqslant 16; 3 \leqslant m + n \leqslant 24)$. Найти количество n-значных натуральных чисел, таких, что их запись в в системе счисления с основанием m не содержит двух подряд значащих нулей.

16.59. ◎ **Метро**[1]**.** В городе есть метрополитен, состоящий из n станций. Станции соединены друг с другом линиями, причём две станции

[1]Белорусская олимпиада по информатике 2002 г.

могут быть соединены только одной линией. В исходном состоянии система метро является связной, то есть с любой станции до любой другой можно добраться, хотя бы и через другие станции.

В связи с убыточностью, принято решение метро закрыть, причем закрывать метро будут постепенно, в год по одной станции. Естественно, после закрытия какой-то станции все линии, к ней подходящие от других станций, тоже прекращают действовать. Требуется указать такой порядок закрытия станций, чтобы после каждого закрытия оставшаяся часть метрополитена оставалась связной.

Техническое требование

В первой строке входного файла находятся два целых неотрицательных числа n и m — соответственно количество станций и количество линий между станциями ($1 \leqslant n \leqslant 100; 0 \leqslant m < 10000$). В следующих m строках находится информация о линиях между станциями, каждая линия обозначена двумя натуральными числами b_i и e_i, обозначающими номера соединяемых этой линией станций. Считать, что система станций и линий задана корректно, то есть любые две станции могут быть соединены только одной линией, номера станций не превосходят n, система связна. В выходном файле в n строках разместить номера станций в том порядке, в котором их нужно закрывать по условию задачи (в каждой строке – один номер).

16.60. По парам. Дано натуральное число $N (N \leqslant 10^5)$. Найти максимальное количество пар натуральных чисел, не превосходящих N, таких, чтобы сумма чисел пары была бы простым числом, причем каждое число может входить лишь в одну пару.

Пример

```
Введите N=8
Ответ: количество пар=4
```

16.61. Дано натуральное число a $(1 \leqslant a \leqslant 10^9)$. Найти минимальное натуральное число n, чтобы n^n делилось бы на a.

16.62. Сумма. Даны натуральные числа $a_1, a_2, \ldots, a_n$. Найти минимальное натуральное число, которое нельзя представить в виде суммы некоторых из данных натуральных чисел (каждое из a_i может входить в сумму только один раз; сумма может состоять и из одного слагаемого).

16.63. Кони, кони... Найти все возможные расстановки двенадцати коней на обычной шахматной доске, при которых каждое поле доски было бы под ударом хотя бы одного из них.

16.64. Миролюбивые магараджи. Будем считать, что «шахматная» фигура «магараджа» может ходить как ферзь и как конь. Расставить на доске 10×10 десять магараджей так, чтобы они не угрожали друг другу.

16.65. Многопроцессорность. В конфигурацию компьютера включены n одинаковых процессоров ($n \geqslant 1$). На вход последовательно поступают задания; каждое из заданий может быть направлено для выполнения на любой из свободных процессоров, причем оно занимает этот процессор до своего полного завершения. Прерываний исполнения процессором заданий не происходит. Очередное поступившее задание направляется на любой из освободившихся процессоров.

На компьютере надо выполнить m заданий, которые требуют времени $t_1, t_2, \ldots, t_m$ соответственно ($t_i > 0, i = 1, 2, \ldots, m$). Определить такую очередность последовательного поступления заданий, чтобы задания были выполнены за минимальное время.

16.66. Разложить факториал. Дано натуральное число n ($n \leqslant 1000$). Разложить число $n!$ в произведение простых чисел (если простое число входит в разложение несколько раз, то выдать его один раз и указать показатель степени – количество вхождений множителя).

16.67. Делители факториала. Даны натуральные числа n ($n \leqslant 1000$), a и b ($a \leqslant b \leqslant 10^6$). Найти количество делителей числа $n!$, принадлежащих отрезку $[a, b]$.

16.68. Игра с лунками. Имеется n лунок, расположенных в линию ($n \geqslant 3$). В каждой лунке может находиться не более, чем по одному шару. В первых k лунках находятся белые шары ($1 \leqslant k \leqslant n-2$), $(k+1)$-я лунка пустая, в остальных $n - k - 1$ лунках лежат черные шары. За один ход можно шар из любой лунки переместить либо в соседнюю пустую лунку, либо через одну лунку в пустую. За какое минимальное количество ходов можно все белые шары переместить в конец ряда лунок, а все черные — в его начало?

16.69. Многоугольник. Точки плоскости (x_1, y_1), (x_2, y_2),..., (x_n, y_n), где $n \leqslant 1000$, являются последовательными вершинами многоугольника. Найти количество точек плоскости с целочисленными координатами, лежащих внутри многоугольника, но не на его границе. Считать, что последовательность точек такова, что несмежные стороны многоугольника не пересекаются.

16.70. ◎ **Конфуз**[2]**.** Пусть a — целочисленный массив, состоящий из n элементов ($2 \leqslant n \leqslant 10^4$) $a_1, a_2, \ldots, a_n$, где $|a_i| \leqslant 2 \cdot 10^9, i = 1, 2, \ldots, n$. Вычислим сумму элементов S, $S = a_1 + a_2 + \cdots + a_n$. Заменим каждый элемент массива на разницу S и этого элемента: $a_i := S - a_i, 1 \leqslant i \leqslant n$. Такое преобразование массива a назовем операцией Confuse. Напишите программу, которая по массиву b, полученному в результате k-кратного применения операции Confuse к некоторому массиву a ($1 \leqslant k \leqslant 100$), вычислит разность максимального и минимального элементов исходного массива a.

16.71. ◎ **Абракадабра**[3]**.** Последовательность строк $s_0, s_1, s_2, s_3 \ldots$ строится следующим образом: s_0— пустая строка, а каждая последующая строка получается удваиванием предыдущей строки и приписыванием к ней слева очередной буквы латинского алфавита (начиная с буквы a). Таким образом,
$s_0 =$''; $s_1 =$'*a*'; $s_2 =$'*baa*'; $s_3 =$'*cbaabaa*'; ...
Дано натуральные числа n, m ($1 \leqslant n \leqslant 26; 1 \leqslant m \leqslant 2^n - 1$). Определить символ, который стоит в m-ой позиции в строке s_n.

16.72. **Бусы.** Сколько разных бус можно собрать из данного набора разноцветных бусинок? Бусы — это бусинки надетые на нитку связанную кольцом, поэтому у них нет начала и конца. Понятно, что как бусы не поворачивай — это одни и те же бусы. Если поменять местами бусинки одного цвета, бусы от этого не изменяться.

Техническое требование

Задана строка латинских букв, больших и маленьких. Каждая буква обозначает цвет. Причём, маленькая буква обозначает темный вариант, а большая — светлый (r — красный, R — светло красный). Бусинок одного цвета столько, сколько раз встречается эта буква в строке. Длина строки не более 50 символов.

Пример

```
Бусинки: tSst
Разных бус: 2
```

16.73. ◎ **Гвозди**[4]**.** На прямоугольном столе разложили n одинаковых листков бумаги со сторонами, параллельными краям стола. Требуется прибить листки к столу гвоздями, причем так, чтобы каждый гвоздь прикреплял бы к столу хотя бы один листок и каждый листок был бы

[2]XV Всеукраинская олимпиада по информатике.
[3]Республиканская олимпиада по информатике Удмуртской Республики, 2001 г.
[4]Республиканская олимпиада по информатике Удмуртской Республики, 2001 г.

прибит ровно одним гвоздём. Гвозди нельзя забивать в границы листков. Считать, что длина каждого листка параллельна оси Ox, а ширина параллельна оси Oy.

Техническое требование

В первой строке входного файла записаны три числа: n — количество листков бумаги ($1 \leqslant n \leqslant 100$), a и b размеры каждого листка ($1 \leqslant a, b \leqslant 1000$). В каждой из следующих n строк указано по паре чисел x_i, y_i — координаты левого нижнего угла очередного листка ($0 \leqslant x_i, y_i \leqslant 1000$). Все числа целые.

В выходной файл нужно поместить целое число, равное 0, если такого способа прибить гвозди не существует, или равное минимальному количеству гвоздей, которыми прибивается бумага в соответствии с условием задачи.

16.74. Большое число. Даны натуральные числа m, n ($1 \leqslant m, n \leqslant 10^{100}$). Найти последнюю цифру числа m^n.

16.75. Вот такие фишки. На прямоугольном клетчатом поле размера $m \times n$, где m обозначает количество горизонтальных рядов, а n — количество вертикалей, размещены n фишек, по одной в каждой вертикали; каждая фишка занимает одну клетку. За один ход можно переместить любую фишку на одну клетку вверх или вниз по вертикальному ряду клеток. Найти минимальное количество ходов, за которые все фишки можно выстроить в одну (любую) горизонталь.

Техническое требование

В первой строке входного файла указаны два натуральных числа m, n ($1 \leqslant m, n \leqslant 10^4$). Далее в файле размещены n натуральных чисел, каждое i-ое число принадлежит отрезку $[1; m]$ и обозначает, в какой горизонтали изначально находится i-ая фишка. В выходной файл необходимо записать одно натуральное число, обозначающее минимально необходимое количество ходов, за которые все фишки можно выстроить в один ряд.

16.76. Шарики. В двух ящиках находятся шарики, в первом m штук, во втором n штук ($1 \leqslant m + n \leqslant 2 \cdot 10^9$). За один ход можно из одного ящика переложить в другой ящик столько шариков, сколько в нём уже находится. Можно ли переложить все шарики в один из ящиков?

16.77. Компьютерная сеть. В некоторой организации решили создать компьютерную сеть. Для этого от единого сервера к каждому из компьютеров сети нужно проложить отдельный кабель. Но руководители организации решили сэкономить, и не закупать новый кабель, а ис-

пользовать для этого обрезки старого кабеля. Два обрезка кабеля можно спаять, получив один более длинный обрезок. Любой обрезок можно разрезать на два. Надо определить, за какое минимальное количество спаек можно сделать нужное количество соединений компьютеров с сервером. Разрешается делать неограниченное количество разрезов обрезков кабеля.

Техническое требование

В первой строке входного файла даны два числа: m — количество подключаемых компьютеров к серверу, n — количество имеющихся обезков кабеля ($1 \leqslant m \leqslant 100, 1 \leqslant n \leqslant 1000$). В следущих m строках находятся числа, по одному в каждой строке. Они обозначают расстояния от каждого из m компьютеров до сервера. В следующих n строках расположены числа, по одному в каждой строке, которые обозначают длины имеющихся обрезков кабеля. Все числа целые положительные. Программа в выходной файл должна выдавать ответ в виде целого числа, равного -1, если создать сеть в соответствии с указанными условиями нельзя, или равного минимальному количеству спаек обрезков, сделав которые, можно соединить каждый компьютер с сервером.

Примечание. Надеемся, что в реальных организациях руководство не будет принимать подобных решений о создании сети на основе таких паяных-перепаяных кабелей...

16.78. Упорядочим числа по алфавиту. Даны натуральные n, k, m ($1 \leqslant n \leqslant 10^6; 2 \leqslant k \leqslant 36; 1 \leqslant m \leqslant n$. Расмотрим последовательность десятичных записей первых n натуральных чисел. Например, для $n = 10$ получим:

$1_{(10)}, 2_{(10)}, 3_{(10)}, 4_{(10)}, 5_{(10)}, 6_{(10)}, 7_{(10)}, 8_{(10)}, 9_{(10)}, 10_{(10)},$

Запишем эти числа в системе счисления с основанием k, равного, например, 3:

$1_{(3)}, 2_{(3)}, 10_{(3)}, 11_{(3)}, 12_{(3)}, 20_{(3)}, 21_{(3)}, 22_{(3)}, 100_{(3)}, 101_{(3)}$

Упорядочим записи этих чисел лексикографически, то есть так, как принято в словарях:

$1_{(3)}, 10_{(3)}, 100_{(3)}, 101_{(3)}, 11_{(3)}, 12_{(3)}, 2_{(3)}, 20_{(3)}, 21_{(3)}, 22_{(3)}$

Переведем опять в десятичную систему:

$1_{(10)}, 3_{(10)}, 9_{(10)}, 10_{(10)}, 4_{(10)}, 5_{(10)}, 2_{(10)}, 6_{(10)}, 7_{(10)}, 8_{(10)}$

Требуется найти, чему равно m-ое число этой последовательности (то есть найти десятичную запись m-го числа последовательности натуральных чисел от 1 до n, упорядоченной лексикографически в соответствии с k-ичными записями этих чисел).

Пример

```
n=10
```

```
k=3
m=5
Ответ: это число 4
```

Примечание: считать, что буквы в алфавите идут после цифр.

16.79. Двоичное плюс-минус. Дана строка, представляющая запись натурального числа n в двоичной системе счисления ($1 \leqslant n \leqslant 2^{100}$). Найти результат сложения n с 1 и вычитания из него 1.

16.80. Двоичное минус степень. Дана строка, представляющая запись натурального числа n в двоичной системе счисления и целое число m ($1 \leqslant n \leqslant 2^{100}; 0 \leqslant m \leqslant 100$). Найти разность n и 2^m.

16.81. Уравнение в целых. Даны целые a_1, a_2, a_3, a_4, b, натуральное m. Решить уравнение $a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 = b$ в целых числах, не превосходящих по модулю m; то есть найти все четверки целочисленных значений $(x_1; x_2; x_3; x_4)$, обращающих уравнение в тождество, для которых $|x_i| \leqslant m, i = 1, 2, 3, 4$.

16.82. Из a сделать b. Даны два целых положительных числа a, b. Найти минимальное количество цифр двоичной записи числа a, вычеркнув которые можно получить двоичную запись числа, равного b или сообщить, что это сделать невозможно.

Пример

```
a=73
b=5
Ответ: количество цифр равно 2
```

16.83. Произведение цифр. Дано натуральное число $n \leqslant 10^6$. Найти наименьшее натуральное число, произведение цифр десятичной записи которого равнялось бы n или сообщить, что таких чисел не существует.

Пример

```
Введите n=120
Ответ:358
```

16.84. Несократимые дроби. Даны натуральные числа n и d ($0 \leqslant n < d \leqslant 255$). Найти количество всех обыкновенных правильных несократимых дробей, имеющих различные значения, числитель которых не превосходит n, а знаменатель не превосходит d.

Пример

```
Введите n=3
Введите d=5
Ответ: количество дробей равно 9
```

16.85. Реверси. Игра «Реверси» имеет следующие правила. Квадратная доска разбита на $N \times N$ клеток ($3 \leqslant N \leqslant 10$). Играют двое, делая по очереди ходы. В игре используются особые фишки, с одной стороны фишка имеет белый цвет (цвет первого игорка), с другой — черный (цвет второго). Каждый игрок за один ход может поместить одну фишку в любую пустую клетку доски своим цветом вверх. Например, свой ход сделал игрок белыми фишками. Если после этого хода на доске по горизонтали, вертикали или диагонали складывается последовательность фишек черного цвета, ограниченная по краям двумя белыми фишками, то все эти черные фишки переворачиваются и становятся белыми; таких последовательностей может получиться несколько. Аналогично ходит игрок черными фишками. Игра заканчивается, когда вся доска заполнена фишками. Победившим считается игрок, у которого на доске больше фишек с его цветом; возможна ничья.

Напишите программу, моделирующую «Реверси» и играющую за одного из игроков.

16.86. Параллельные вычисления. Во входном файле задана программа, состоящая из N операторов присваивания $1 \leqslant N \leqslant 16$; синтаксис:

```
<присв> ::= <идент>:=<идент><оп><идент>
<оп>::=+|-|*|/
<идент> ::= A|B|...|Z
```

Требуется распределить выполнение этих операторов между двумя процессорами таким образом, чтобы общее время выполения было минимальным, и при этом результат выполения был бы эквивалентен исходной программе. Будем считать, что для выполнения одного оператора процессору требуется один такт работы. Введем еще один оператор NULL, который ничего не изменяет, но процессор также тратит на его один такт. Очевидно, что два оператора присваивания не могут быть одновременно выполнены на разных процессорах, если в левой части одного из них содержится идентификатор переменной, которая есть в правой части другого.

Техническое требование

Во входном текстовом файле дан текст программы в соответствии с вышеприведенным синтаксисом. В каждой строке расположен ровно один оператор присваивания, пустых строк нет.

В выходной файл требуется поместить результат распределения программы на два процессора в следующем виде: в первой строке указывается целое положительное число T — время выполнения программы (в тактах работы). В каждой из T следующих строк располагаются по

два оператора, выполняемых одновременно: первый — для первого процессора, второй — для второго. Операторы разделены по крайней мере одним пробелом. В качестве оператора может выступать присваивание, либо оператор NULL.

Пример

Во входном файле:

```
A:=B+C
B:=C-D
M:=K*N
E:=M/B
```

В выходном файле:

```
3
A:=B+C M:=K*N
B:=C-D NULL
NULL  E:=M/B
```

16.87.  **Нарушенный баланс.** В тексте письма могут быть буквы, цифры, знаки препинания, круглые скобки и другие символы. Если текст составлен правильно, то в нем соблюдается баланс круглых скобок, то есть каждой открывающей скобке соответствует далее по тексту ровно одна закрывающая, причем между ними может быть сбалансированная по скобкам часть текста. Текст или часть текста, не содержащая ни одного символа или не содержащая круглых скобок, считается сбалансированной по скобкам.

При передаче правильно составленного письма по электронной почте в результате сбоя из текста исчезло некоторое количество скобок. Необходимо определить, сколько существует различных способов вставки недостающего минимального числа скобок в поврежденный текст так, чтобы восстановить баланс скобок. Вставлять скобки можно между любыми символами текста, а также перед первым и после последнего символа.

Техническое требование

Вход: Входной текст читается из текстового файла с именем *INPUT.TXT*. Максимальная длина текста — 100 символов. Пробелы и символы окончания строк не считаются частью текста и должны быть проигнорированы.

Выход: на экран выводится целое число, равное

а) количеству различных способов вставки недостающего минимального числа скобок, если баланс скобок нарушен и его вставкой скобок можно восстановить, или

б) 0, если баланс скобок не нарушен, или

в) -1, если баланс скобок в тексте нельзя восстановить ни при какой

вставке скобок

Максимальное время работы программы — 15 секунд.

Пример

x)a(;-)(x()(

Выход: 10

16.88. **Дроби.** Дано действительное число в десятичной записи. Необходимо указать равное ему смешанное число с выделенной целой частью, дробная часть которого является правильной несократимой обыкновенной дробью.

Техническое требование

Вход: Входной текст читается из текстового файла с именем *INPUT.TXT*. В файле задано действительное число (возможно, отрицательное; возможно, с дробной частью; возможно, с периодом) согласно синтаксиса:

```
<Действительное> ::= [-]<Целая часть>[.<Дробная часть>]
<Целая часть>    ::= <Целое>
<Дробная часть>  ::= <Целое>(<Целое>)
                   | <Целое>
                   | (<Целое>)
```

Здесь <Целое> означает любую непустую последовательность цифр 0, 1, 2, ..., 9. В круглых скобках указывается период числа. Максимальное количество цифр в целой части равно 9. Максимальное количество цифр в дробной части равно 9.

Выход: Выдать на экран ответ в виде смешанного числа с целой частью и дробной частью. Дробная часть должна быть правильной несократимой обыкновенной дробью. Если или целая часть равна нулю, или дробная часть равна нулю, то ее выводить не нужно. Если число равно нулю, то вывести только целую часть числа. Целую часть от дробной части отделить одним пробелом, числитель дробной части от знаменателя отделить знаком '/'.

Максимальное время работы программы — 15 секунд.

Пример

Вход: *-1.2(34)*

Выход: -1 116/495

16.89. Решить задачу, обратную 16.88: дана запись смешанного числа (в соответствии с правилами из задачи). Найти десятичную запись этого числа в виде, указанном в задаче 16.88.

16.90. Сапер. Известная игра «минер» устроена так: минное поле представляет собой клетчатый прямоугольник, в клетках которого могут быть мины. Каждая клетка содержит либо мину, либо число мин в соседних клетках (у каждой клетки может быть до восьми соседей). Все клетки изначально скрыты. Каждый шаг игрока заключается в попытке вскрыть одну из клеток поля. Если там оказывается мина, то игра прекращается, игрок считается проигравшим, если нет — то в клетке отображается количество соседних мин. В случае, когда мин рядом нет, то вскрывается данная клетка, а также рекурсивно все соседние клетки. Игрок может помечать клетки, где могут находиться мины, не вскрывая их. Цель игрока — обнаружить все мины, то есть вскрыть все клетки без мин.
Требуется написать программу «сапер», играющую за игрока. Ясно, что не всегда удается определить ход, гарантированно не приводящий к проигрышу. Однако, можно вычислить наиболее вероятный успешный из всех возможных ходов.

16.91. Игра Ним. В игре участвуют три игрока. Даны три кучки камешков. Ход игрока состоит в том, что он забирает из какой-то одной (произвольной) кучки положительное число камешков. Игроки ходят по очереди. Выигрывает тот, кто заберет последний камешек. Написать программу, моделирующую игру Ним, в которой участвуют два человека, а за третьего играет программа.

16.92. Сама себя. Написать программу, выводящую свой собственный текст.
Ограничение: тривиальные программы, использующие открытие файла с исходным текстом или некоторые особенности конкретного языка (например, в Бейсике программа `10 LIST`) решением не считаются. Это ограничение делает задачу более точной, в отличие от формулировки из книги [18].

16.93. Все соседи разные. Дано натуральное число N ($2 \leqslant N \leqslant 10$). Заполнить квадратную матрицу порядка N целыми числами $0, 1, 2, 3$ таким образом, чтобы у любого элемента матрицы все его соседи были различны (соседом элемента считается другой элемент, у которого либо номер строки, либо номер столбца отличается на единицу).

16.94. Пересечение отрезков. Дано множество из n отрезков числовой прямой, задаваемых координатами концов a_i, b_i, где $a_i \leqslant b_i, i = 1, 2, \ldots, n$, a_i и b_i — действительные числа, $1 \leqslant n \leqslant 100$. Дано действительное положительное число l. Найти наименьшее число m ($0 \leqslant m < n$),

такое, что при удалении некоторых m отрезков из множества, пересечение оставшихся будет иметь длину, не меньшую чем l, либо сообщить, что такого числа не существует.

Пример

```
Введите n=4
Введите попарно концы 4 отрезков:
0 2
1 3
2 4
0 4
Введите l=2
Ответ: минимальное количество отрезков, которые надо удалить
равно 2
```

16.95. Объединение отрезков. Дано множество из n отрезков числовой прямой, задаваемых координатами концов a_i, b_i, где $a_i \leqslant b_i, i = 1, 2, \ldots, n$, a_i и b_i — действительные числа, $1 \leqslant n \leqslant 100$. Дано действительное положительное число l. Найти наименьшее число m $(0 \leqslant m < n)$, такое, что при удалении некоторых m отрезков из множества, объединение оставшихся будет иметь длину, не большую чем l, либо сообщить, что такого числа не существует. Длиной объединения множества отрезков будем считать сумму длин отдельных частей, из которых состоит это объединение. Например, длина объединения отрезков $[0; 3]$, $[1; 2]$ и $[4; 5]$ равна 4.

Пример

```
Введите n=4
Введите попарно концы 4 отрезков:
0 2
1 3
2 4
0 4
Введите l=2
Ответ: минимальное количество отрезков, которые надо удалить
равно 3
```

16.96. Ориентирование на местности. Соревнования по ориентированию проходят на некоторой местности. Местность имеет форму квадрата размером 2×2 километра. На местности введена система координат, начало которой расположено в левом нижнем углу квадрата, оси направлены соответственно слева направо и снизу вверх. Квадрат разбит на два равных прямоугольных участка размерами 2×1 километр,

обозначенные буквами A и B; A расположен внизу, B расположен вверху квадрата.

У участника имеется карта местности, на которой помечены n контрольных пунктов, в каждом из которых он должен отметиться. Путь участника начинается в начале координат. Между двумя любыми пунктами можно выбирать путь в виде линии любого вида, выходить за пределы квадрата местности нельзя. Так как характер местности на каждом из участков различен, то скорости, с которыми может участник двигаться по ним, могут отличаться: обозначим их v_A и v_B соответственно, скорость измеряется в метрах в минуту. При движении по границе между участками скорость движения равна скорости более 'быстрого' участка из двух.

Требуется найти минимальное время, за которое участник, выйдя из начала координат, сможет обойти все контрольные пункты.

Техническое требование

В первой строке входного текстового файла указано одно натуральное число — количество контрольных пунктов n ($1 \leqslant n \leqslant 16$). Во второй строке входного файла указаны два натуральных числа — скорости передвижения участника по участкам v_A и v_B соответственно ($1 \leqslant v_A, v_B \leqslant 500$). В каждой из следующих n строк расположено по два целых неотрицательных числа x_i, y_i, обозначающих координаты контрольных пунктов в метрах ($0 \leqslant x_i, y_i \leqslant 2000$).

В выходной текстовый файл требуется записать одно целое число — минимальное время в минутах, которое потребуется участнику для обхода всех пунктов, начиная с начала координат. Действительное значение времени нужно округлить до целого. Допускается ошибка не более чем на одну минуту.

16.97. Общая точка треугольников. На плоскости даны координаты n различных точек $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$, где x_l, y_l — действительные числа, $1 \leqslant l \leqslant n \leqslant 100$. Некоторые точки являются вершинами m треугольников, каждый треугольник задается номерами трех его различных вершин v_i, v_j, v_k ($1 \leqslant v_i, v_j, v_k \leqslant n$). Найти координаты любой точки плоскости, принадлежащей наибольшему числу треугольников. Точка считается принадлежащей треугольнику, если она лежит внутри него или на его границе. Числа результата вывести с тремя точными знаками после запятой.

16.98. Волшебные векторы. Назовем упорядоченный по неубыванию набор из N натуральных чисел волшебным N-вектором, если сумма этих чисел равна их произведению. Задано число N. Найдите все вол-

шебные N-векторы.

Пример

```
Введите N=5
Все волшебные 5-векторы:
1 1 1 2 5
1 1 1 3 3
1 1 2 2 2
```

16.99. ✣ **Системы счисления.** Запись $(X)_Y$ обозначает, что X есть запись числа в системе счисления по основанию Y. Написать программу, которая решает уравнения вида: $(A)_X = B$, где A и B (целые положительные числа не больше 2000000000) заданы, а X — переменная. A и B — заданы десятичными цифрами.

Пример

```
A= 160
B= 112
X=8
```

16.100. Разложение на различные простые. Даны натуральные числа n, m. Найти количество натуральных чисел, не превосходящих n, таких, которые равны произведению ровно m различных простых чисел.

Пример

```
Введите n=20
Введите m=2
Ответ: таких чисел 4
```

16.101. Минимальный квадрат. Даны координаты n точек плоскости $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$, где x_i, y_i — действительные числа, $1 \leqslant i \leqslant n \leqslant 1000$. Найти сторону минимального квадрата, которому принадлежат все данные точки. Точка считается принадлежащей квадрату, если она лежит внутри него либо на его стороне.

16.102. Гипотеза Гольдбаха. В гипотезе Гольдбаха утверждается, что любое четное число, не меньшее 4, можно представить в виде суммы двух простых чисел. Даны натуральные числа a, b $(4 \leqslant a \leqslant b \leqslant 10^6)$. Проверить гипотезу Гольдбаха для всех четных чисел из отрезка $[a; b]$.

16.103. Теорема Л.Г. Шнирельмана. Теорема Шнирельмана утверждает, что в любую замкнутую кривую на плоскости можно вписать квадрат. Если область, ограниченная кривой невыпукла, то квадрат может выходить за границы области. Таким образом, на любой замкнутой кривой можно найти четыре точки, являющиеся вершинами квадрата.

Дана последовательность координат n различных точек плоскости $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$, являющихся последовательными вершинами замкнутой ломаной линии без самопересечений. Найти квадрат максимальной площади, вершины которого принадлежат данной ломаной (то есть вершины квадрата принадлежат отрезкам ломаной).

16.104. Треугольные числа. Определим последовательность треугольных чисел $t_1, t_2, \ldots, t_i, \ldots$ следующим образом:

$$t_i = \begin{cases} 1, & \text{если } i = 1; \\ t_{i-1} + i, & \text{если } i > 1. \end{cases}$$

Таким образом, начало треугольной последовательности выглядит так: $1, 3, 6, 10, 15, 21, \ldots$

Дано натуральное число N ($N \leqslant 10^9$). Определить, является ли оно треугольным?

16.105. Удалить цифру. Дано натуральное число $N \leqslant 10^6$. Найти основание системы счисления R ($2 \leqslant R \leqslant 36$), чтобы при удалении одной наибольшей цифры из записи числа N в системе счисления с основанием R, число уменьшилось бы максимально.

16.106. Дополнить до квадрата. Дано натуральное число $N \leqslant 10^9$. Найти минимальное простое число, прибавив которое к N получается полный квадрат.

16.107. Вычеркните из числа

12345...99100101...200220032004

N цифр, чтобы оставшееся число было максимальным из возможных.

16.108. Вычеркните из числа

12345...99100101...200220032004

k-значное число (незначащие нули входят), чтобы оставшееся число было максимальным из возможных.

16.109. Вычеркните из числа

12345...99100101...200220032004

N штук k-значных чисел (незначащие нули входят), чтобы оставшееся число было максимальным из возможных.

16.110. Произведение каких зеркальных чисел равно N? Два числа назовём *зеркальными*, если одно получается из другого перестановкой цифр в обратном порядке.

Пример

```
N= 252
12*21 = 252
```

16.111. Сумма каких зеркальных чисел равна N? Два числа назовём *зеркальными*, если одно получается из другого перестановкой цифр в обратном порядке. Достаточно одного варианта.

Пример

```
N= 55
23+32 = 55
```

16.112. Сколько цифр в числе X^Y? X и Y — натуральные числа.

Пример

```
X= 2
Y= 10
4
```

16.113. ✣ **Конструкции фигурок.** Задана строка не более, чем из 1000 символов. Одинаковыми символами в ней задаются конструкции фигурок. *Конструкции фигурок* — это только расстояния, на которых расположены друг от друга символы, но не сами символы. Таким образом, в строке могут быть записаны одинаковые конструкции фигурок разными символами.

Техническое требование

Написать программу, которая по заданной строке выводит число разных конструкций фигурок.

Для записи строки будут использованы только обычные текстовые символы. Входная строка находится в файле *string.txt*. Время работы — 5 секунд.

Пример

В *string.txt*:

abc␣abc␣abc

Ответ:

2

16.114. **RGB в HSV.** Смит предложил построить модель субъективного восприятия в виде объемного тела HSV (цветовой тон, насыщенность, светлота).

Насыщенность меняется от 0 до 1. Отметим, что насыщенность зависит от цветового охвата т.е. от расстояния от оси тела до его границы для каждого V. При $S = 1$ цвета или их дополнения полностью насыщенны. Ненулевая линейная комбинация 3 основных цветов не может быть полностью насыщенна. Если $S = 0$, то тон H неопределен, т.е. на центральной оси находятся ахроматические, серые цвета.

Модель HSV соответствует тому, как составляют цвета художники. Чистым пигментам отвечают значения $S = 1, V = 1$; разбелам — цвета с увеличенным содержанием белого, т.е. с меньшим S; оттенкам — цвета с уменьшенным V, которые получаются при добавлении черного. Тон изменяется при уменьшении как V так и S.

Напишите программу перевода цвета из режима RGB в режим HSV.

16.115. HSV в RGB. До начала решения этой задачи должна быть решена задача 16.114. Напишите программу перевода цвета из режима HSV в режим RGB.

16.116. НУИНУ. Здание НУИНУ[5] представляет собой залы, соединённые коридорами. Найти нужный зал с непривычки очень сложно, так же сложно выйти из института. Чтобы посетители не затруднялись выйти, достаточно в каждом зале повесить табличку «Выход» над коридором ведущим к выходу. Поскольку может оказаться, что путей к выходу несколько, было решено выбирать самый короткий.

Задача

По заданной схеме коридоров указать, где нужно повесить таблички «Выход».

Техническое требование

Структуру входных и выходных данных выбирайте ту, которая удобнее.

16.117. Экономия в НУИНУ. В задаче 16.116 описан способ развешивать таблички «Выход». Но здание большое и табличек нужно много. Руководство института приняло решение развесить таблички не в каждом зале, А только там, где это действительно необходимо.

Оказывается, если есть «кольцо» из залов, то табличку «Выход» можно повесить только в одном зале «кольца». Посетитель всё равно к ней придёт.

[5] Научный универсальный институт необыкновенных услуг. Этот институт фигурирует в фильме «Чародеи», снятый по мотивам книги Стругацких «Понедельник начинается в субботу». Там как раз и возникла проблема с выходом посетителя из здания института.

Техническое требование

Структуру входных и выходных данных выбирайте ту, которая удобнее.

16.118. Арбитр охоты на лис. Написать программу-арбитра для игры «Охота на лис» (см. задачу 16.119).

Арбитр поочерёдно выполняет программы-игроки и выполняет действия предусмотренные правилами.

Арбитр должен проверять и диагностировать следующие ошибки:

- запись, отличная от формата; в файл можно писать ровно 100 текстовых символов, которые описаны выше, ровно по 10 в строке, среди которых не более одного '?';
- замена символов, отличных от '*', в первом режиме, отличных от '?', во втором, замена '?' на '*';
- запись управляющих символов; например, перевод строки, конец файла и т.п.

При обнаружении этих ошибок арбитр засчитывает поражение нарушителю.

Насколько это возможно, арбитр пытается предотвратить или обнаружить следующие нарушения:

- читать и изменять файлы, отличные от forest.txt и своего временного файла;
- хакерские приёмы, например, изменение значений таймера.

Определение «жульничества» происходит следующим образом: программа-арбитр поочерёдно запрашивает информацию обо всех клетках у подозреваемой программы и, если обнаружится нарушение правил, то подозреваемой программе будет засчитано поражение. Если нарушения не обнаружено, то подозреваемой программе засчитывается выигрыш.

По окончанию игры арбитр сообщает, кто победил и по какой причине.

16.119. ❐ **Охота на лис.** Игра «Охота на лис» ведётся на квадратном поле размером 10 на 10 клеток. В пяти клетках поля спрятаны пять лис, по одной в клетке. Цель игры найти всех лис. Для поиска разрешается указывать по одной клетке поля за один ход. В ответ в клетке появляется изображение лисы, если она спрятана здесь. Если лисы в указанном месте нет, появляется число от 0 до 5. Это число указывает, сколько лис расположено в одной вертикали, горизонтали и диагоналях с указанной клеткой. Уже найденные лисы считаются тоже.

Вариант игры для двоих

В эту игру можно играть и вдвоём. Для этого каждый игрок задаёт своё поле и своё расположение лис. Ходы выполняют по одному и пооче-

рёдно. Игрок, нашедший лису, ходит ещё раз. Каждый игрок ищет лис на поле противника. Выигрывает тот, кто первым найдет всех лис.

Задача

Написать программу для игры в «Охоту на лис» для двоих.

Техническое требование

Поле задаётся в файле forest.txt. Файл состоит из 10 строк по 10 символов в каждой. Значения символов следующие: '*' — поле не открыто, '@' — найденная лиса, '0', '1', '2', '3', '4', '5' — количества видимых лис. '?' обозначает клетку, на которую делается ход.

Программа может находиться в двух режимах работы. Первый режим соответствует случаю, при котором на поле нет символа '?'. Второй — случаю, при котором этот символ есть.

В первом режиме в файле forest.txt находится поле противника и Вы должны ходить. Ход заключается в том, что в файл forest.txt записывается исходное поле, где один из символов поля заменён символом '?'.

Во втором режиме в файле forest.txt находится Ваше поле с символом '?', поставленным противником. В этом случае ход заключается в замене знака вопроса нужным символом.

Программа должна делать только один очередной ход и после этого завершать работу. Время на ход — не более 20 секунд для Pentium-100. Всего ходов — не более 100. Если за 100 ходов игра не закончена — ничья.

Программа может использовать временный файл, который имеет имя *tmp.fox*.

Хитрость игры

Поскольку расположение лис противник не знает, разрешается жульничество. Лис можно переставлять на другие клетки и вообще врать. Но если противник заметит, что информация стала противоречивой, то Вам записывается немедленный проигрыш. Если нет: «Не пойман — не вор».

Если Вы заметили, что противник обманывает, то можете записать в начало файла forest.txt строку "Error:", а во второй строке написать причину. Диагностика причины формально оцениваться не будет, но поможет жюри. Но если противник вёл себя честно, то проиграли Вы.

Можно сдаваться. Для этого нужно в файл forest.txt вывести строку "@@@@@".

Программы запускались жюри Олимпиады с программой-арбитром. Арбитр проверял и сам диагностировал следующие ошибки:

- Запись, отличная от формата. В файл можно писать ровно 100 текстовых символов, которые описаны выше, ровно по 10 в строке, среди которых не более одного '?'.
- Замена символов, отличных от '*', в первом режиме, отличных от '?', во втором, замена '?' на '*'.

- Запись управляющих символов. Например, перевод строки, конец файла и т.п.

При обнаружении этих ошибок засчитывалось поражение.

Не допускается:

- Читать и изменять файлы, отличные от forest.txt и своего временного файла. Нарушение этого пункта влечет снятие программы с конкурса.
- Хакерские приёмы, например, изменение значений таймера. Нарушение этого пункта может повлечь дисквалификацию не только на весь тур, но и на весь конкурс.

Оценка программы

Первоначально программы были испытаны на нескольких тестах.

Далее среди программ, корректно работавших на предыдущем этапе, был проведён турнир. Ничья приносит по 1 очку, выигрыш — 2, проигрыш — 0. Каждая игра турнира заключалась в поочерёдном запуске двух программ по правилам игры для двоих.

Пример

Приводятся состояния после работы программ

```
Ход 1  Программа 1   Программа 2  Программа 1   Программа 2
       **********    **********   **********    **********
       *?********    *@********   *@********    *@********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   *?********    *5********
       **********    **********   **********    **********
```

```
       Программа 2   Программа 1
       *********?    *********0
       **********    **********
       **********    **********
       **********    **********
       **********    **********
       **********    **********
       **********    **********
       **********    **********
       **********    **********
       **********    **********

Ход 2  Программа 1   Программа 2  Программа 2   Программа 1
       **********    **********   *********0    *********0
       *@********    *@********   **********    **********
       *********?    *********0   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       *5********    *5********   **********    **********
       **********    **********   *********?    *********1

Ход 3  Программа 1   Программа 2  Программа 2   Программа 1
       **********    **********   *********0    *********0
       *@********    *@********   **********    **********
       *********0    *********0   **********    **********
       **********    **********   **********    **********
       **********    **********   **********    **********
       **********    **********   ****?*****    ****2*****
       **********    **********   **********    **********
       **********    **********   **********    **********
       *5?*******    *55*******   **********    **********
       **********    **********   *********1    *********1

Ход 4  Программа 1
       Error:
       Лис больше 5??!

 Программа 1 выиграла!
```

16.120. **Спрячь лису.** Лиса считается видимой из данной клетки, если она находится с ней в одной и той же вертикали, горизонтали или диагонали.

Сколько, максимум, можно поставить лис на прямоугольном поле M строк на N столбцов, так чтобы из каждой клетки было видно не более трёх лис (включая клетки с лисами)? Лиса, находящаяся в данной клетке, тоже считается (как следует из предыдущего абзаца).

Техническое требование

Программа должна запрашивать размеры поля и вводить в файл output.fox расположение лис. $1 \leqslant M, N \leqslant 100$.

Формат выходного файла

```
<N - Количество лис>
<Горизонталь1> <Вертикаль1>
<Горизонталь2> <Вертикаль2>
...
<ГоризонтальN> <ВертикальN>
```

Если решения нет, вывести 0.

Пример

```
M= 4
N= 5
4
```

Если входные данные неправильны, программа должна выдать осмысленное сообщение об ошибке и нормально завершить работу. Единственное, что можно предполагать: они являются числами.

16.121. **Круглый трек.**

Велосипедисты ездят с постоянной скоростью по круглому треку с длиной окружности 1 км. Трек достаточно широк, чтобы разъехаться двум велосипедистам. При встрече в одной точке трех или более происходит столкновение, и гонки завершаются. Даны начальные позиции и скорости велосипедистов. Определить время до первого столкновения.

Техническое требование

Исходные данные находятся в файле velos.txt. Они заданы в следующем формате:

<Число велосипедистов n $(3 \leqslant n \leqslant 1000)$>

<Начальное положение и скорость первого>

...

<Начальное положение и скорость последнего>

Начальные положения задаются как рациональные числа $0 \leqslant a \leqslant 1$ в км.

Скорости также как рациональные числа в м/сек.

Рациональное число задается парой целых чисел: числитель, знаменатель. И числитель, и знаменатель меньше по модулю 2^{15}. Таким образом, в каждой строчке, кроме первой, по четыре числа.

Результат расчета выдается на экран как число с 12 знаками до запятой и 12 знаками после запятой.

Если гонки продолжаются вечно, то выдайте строчку

```
infinity
```

Если знаменатель одной из дробей 0 или неправильны другие исходные значения, программа должна выдать осмысленное сообщение об ошибке и нормально завершить работу.

Примеры

1) входной файл

```
3 0 1 1 1 1 2 2 1 3 4 -3 2
```

Ответ:

```
1500
```

2) входной файл

```
2000 ...
```

Ответ (примерный)

```
Слишком много велосипедистов.
```

16.122. **Задача коммивояжера.** *Коммивояжер* (франц. commis voyageur), разъездной представитель торговой фирмы, предлагающий покупателям товары по имеющимся у него образцам, каталогам и т. п. (мегаэнциклопедия «Кирилл и Мефодий»[6]).

Наш коммивояжер ездит по городам с целью сбыта товаров разного рода. Он всегда начинает и заканчивает свой путь в одном и том же городе. На проживание во всех городах коммерсант тратит одинаковую сумму денег. Поэтому существенна только сумма на проезд из города в город. Есть список городов и стоимость проезда между некоторыми парами городов.

Задача: побывать во всех городах (можно не по разу) и при этом потратить как можно меньше денег на проезд и вернуться обратно.

[6] http://mega.km.ru

Техническое требование

Считаем, что:

1) внутри города проезд ничего не стоит;
2) проезд между двумя городами напрямую стоит одинаково в оба конца;
3) стоимость — целое число от 1 до 10000;
4) городов не более 100.

Стоимости проезда между парами городов записаны в файле cost.txt в следующем формате:

```
<Количество стоимостей>
<Начальный город>
<Город> <Город> <Стоимость>
<Город> <Город> <Стоимость>
<Город> <Город> <Стоимость>
...
```

Город задается названием без пробела. Длина названия города не более 30 символов.

Программа должна реагировать на клавишу ESC. Если ESC была нажата, то в течении 10 секунд программа должна записать результат и закончить работу.

Результат записывается в файл way.txt в следующем формате:

```
<Количество городов в пути>
<Город>
<Город>
<Город>
...
```

Пример

В файле cost.txt:

```
8
Ижевск
Москва Ижевск 5
С.-Петерб.(Ленинград)  Ижевск 6
Челябинск  Ижевск 7
Комсомольск-на-Амуре  Ижевск 8
С.-Петерб.(Ленинград) Москва 10
Челябинск С.-Петерб.(Ленинград) 20
Комсомольск-на-Амуре Челябинск 30
Москва Комсомольск-на-Амуре 40
```

Ответ (в файле way.txt):

```
8
Ижевск
Москва
С.-Петерб.(Ленинград)
Ижевск
Челябинск
Ижевск
Комсомольск-на-Амуре
Ижевск
```

16.123. Великая Теорема Ферма. Великая Теорема Ферма звучит так: уравнение

$$x^n + y^n = z^n$$

для натуральных x, y, z, n при $n > 2$ решений не имеет.

Требуется проверить это утверждение для максимального количества троек $\langle x, y, z \rangle$. Желательно, чтобы количество проверенных x, y и z в этих тройках отличалось не более, чем на 10.

Техническое требование

Написать программу для решения данной задачи. Нужно учесть возможность продолжать вычисления с места остановки и распараллеливания вычислений.

16.124. Печатные платы. Для изготовления электронного оборудования на одном предприятии выпускаются различные типы печатных плат. Печатная плата представляет собой систему проводников, размещенных на изолирующем материале, изготовленном в виде плоской прямоугольной пластины. Проводники соединяют между собой контактные площадки, к которым припаиваются электронные компоненты. Требуется выяснить, можно ли одну печатную плату заменить другой. Считается, что одну печатную плату можно заменить другой, если любое размещение электронных компонент на первой плате можно заменить размещением тех же компонент на второй плате, причем компоненты будут соединены точно таким же способом.

Платы задаются прямоугольниками, проводники соединяют точки с целочисленными координатами и прокладываются вдоль осей координат. Проводники не имеют пересечений (плата является однослойной). Контактные площадки расположены группами в количестве не более, чем по 4 в точках с целочисленными координатами. Количество контактных площадок совпадает с количеством проводников, входящих в данную точку.

Техническое требование

Входные данные задаются в файле INPUT.TXT в следующем виде (все числа натуральные):

```
<Число групп контактных площадок>
<Ширина первой платы (по оси X)>
<Высота первой платы (по оси Y)>
<X> <Y>      - координаты групп
контактных площадок
...
<X> <Y>
<Число вертикальных отрезков проводников (длины 1)>
<X> <Y>      - координаты верхних концов проводников
...
<X> <Y>
<Число горизонтальных отрезков проводников (длины 1)>
<X> <Y>      - координаты нижних концов проводников
...
<X> <Y>
<Ширина второй платы по оси X>
<Высота второй платы по оси Y>
<X> <Y>      - координаты групп контактных площадок
...
<X> <Y>
<Число вертикальных отрезков проводников (длины 1)>
<X> <Y>      - координаты верхних концов проводников
...
<X> <Y>
<Число горизонтальных отрезков проводников (длины 1)>
<X> <Y>      - координаты левых концов проводников
...
<X> <Y>
```

Результат — слово ДА или НЕТ — выдается на экран.

Пример

Содержимое входного файла:

```
2
1
1
0 1
1 0
```

```
2
0 1
1 1
2
0 0
0 1
1
2
1 2
0 0
4
0 1
1 1
0 2
1 2
2
0 2
0 0
Ответ: ДА
```

16.125. **Кладоискатель.** В лабиринте лежат клады. Кладоискатель должен собирать их. Но лабиринт охраняет злой Сторож. У Кладоискателя есть только 100 минут на сбор кладов. Через 100 минут Кладоискателя заберёт вертолёт. Отсюда у Кладоискателя две задачи. Первая — собрать как можно больше кладов. Вторая — не попасться сторожу.

Техническое требование

Написать программу, которая читает файл с лабиринтом (*maze.txt*) и выводит в файл *path.txt* последовательность ходов Кладоискателя. Если Кладоискателя ловит Сторож, то строка будет содержать меньше символов.

Кладоискатель и Сторож делают по одному ходу в минуту. Ходы выполняются поочерёдно. Первым ходит Кладоискатель, вторым — Сторож.

Программа должна работать не дольше 50 секунд.

ЛАБИРИНТ

Лабиринт задается в текстовом файле *maze.txt*. Стены обозначены символами '#', проходы — '␣' (символ пробел), клад — '*', Кладоискатель — 'S', Сторож — 'D'. Лабиринт всегда прямоугольного размера 12×12. Сторож и Кладоискатель всегда по одному. Первая и последняя строки файла всегда состоят из символов '#'. Каждая строка начинает-

ся и заканчивается символом '#'. Таким образом, лабиринт всегда имеет по контуру стену.

ПОВЕДЕНИЕ КЛАДОИСКАТЕЛЯ

Кладоискатель может ходить по 4 направлениям: вправо, влево, вверх и вниз. Эти направления обозначаются заглавными буквами латинского алфавита: 'R', 'L', 'U', 'D'. Кладоискатель может стоять на месте это обозначается буквой 'S'. Ходить можно только по символам '␣' и '*'. После прохождения по '*' она заменяется на '␣'.

ПОВЕДЕНИЕ СТОРОЖА

Сторож не отличается интеллектом и всегда гонится за Кладоискателем.

Сторож может ходить по 4 направлениям: вправо, влево, вверх, вниз или стоять на месте. Он стремиться сократить сумму расстояний по вертикали и горизонтали между собой и Кладоискателем. При этом он пытается сократить то из расстояний, которое больше. При одинаковом расстоянии он выбирает горизонтальное и, если нельзя, то вертикальное (но только при одинаковых расстояниях). Расстояния считаются по прямой, без учёта препятствий. Сторож стоит на месте лишь когда перед ним находится препятствие.

Пример

В файле maze.txt:

```
############
# * * * * *#
#* * * * * #
# * * * * *#
#* * * * * #
# * # * * *#
#* * * * * #
# * * * * *#
#* * * * * #
#S* * * * D#
############
############
```

В файле path.txt (один из ответов, может не лучший):

```
UUUUURURDURSSSSSS...SS
```

Так Кладоискатель соберёт 6 кладов.

ОЦЕНКА ЗАДАЧИ

Чем больше кладов набирает Кладоискатель, тем больше баллов получает программа. Если Кладоискателю невозможно прятаться от Сторожа 100 ходов, то выигрывает тот, кто дольше прятался.

16.126. Квадратная спираль.

Квадратная таблица размером 40000×40000 заполнена последовательными натуральными числами начиная с 1 по спирали. Спираль начинается в левом верхнем углу с координатам (1, 1) и закручивается по часовой стрелке вовнутрь. Например, спираль 3×3 выглядит так как показано на рисунке 16.2.

1	2	3
8	9	4
7	6	5

Рис. 16.2: Спираль 3×3

Техническое требование

Написать программу, которая по заданным координатам (строка, столбец) выводит число в клетке с этими координатами.

Пример

```
Координаты: 2   10
160005
```

16.127. Кладоискатель II.

Эта задача является модификацией задачи 16.125

В лабиринте лежат клады. Кладоискатель должен собирать их. Но лабиринт охраняет злой Сторож. У Кладоискателя есть только 100 минут на сбор кладов. Через 100 минут Кладоискателя заберёт вертолёт. Отсюда у Кладоискателя две задачи. Первая — собрать как можно больше кладов. Вторая — не попасться сторожу.

Техническое требование

Написать программу, которая читает файл с лабиринтом (maze.txt) и выводит на экран количество кладов собранных Кладоискателем.

Если Кладоискателя ловит Сторож, то выводить нужно 0.

Кладоискатель и Сторож делают по одному ходу в минуту. Ходы выполняются поочерёдно. Первым ходит Кладоискатель, вторым — Сторож.

Программа должна работать не дольше 5 секунд.

ЛАБИРИНТ

Лабиринт задается в текстовом файле maze.txt. Стены обозначены символами '#', проходы — '␣' (символ пробел), клад — '*', Кладоискатель — 'S', Сторож — 'D'. Лабиринт всегда прямоугольного размера 12×12. Сторож и Кладоискатель всегда по одному. Первая и последняя строки файла всегда состоят из символов '#'. Каждая строка начинается и заканчивается символом '#'. Таким образом, лабиринт всегда имеет по контуру стену.

ПОВЕДЕНИЕ КЛАДОИСКАТЕЛЯ

Кладоискатель может ходить по 4 направлениям: вправо, влево, вверх и вниз. Кладоискатель может стоять на месте. Ходить можно толь-

ко по символам '␣' и '*'. После прохождения по '*' она заменяется на '␣'.

ПОВЕДЕНИЕ СТОРОЖА

Сторож отличается идеальным интеллектом и всегда гонится за Кладоискателем не давая ему собирать клады.

Сторож может ходить по 4 направлениям: вправо, влево, вверх, вниз или стоять на месте. Сторож всегда выбирает такой ход, что бы в конце Кладоискатель собрал наименьшее количество кладов.

Пример

В файле maze.txt:

```
############
# * * * * *#
#* * * * * #
# * * * * *#
#* * * * * #
# * # * * *#
#* * * * * #
# * * * * *#
#* D * * * #
#S* * * * *#
############
############
```

```
Ответ: 4
```

16.128. **Сложение.** Число от 0 до $p_1p_2\ldots p_n-1$, где p_i — различные простые числа, однозначно представляется массивом своих остатков от деления на p_i.

Написать программу, которая по заданному числу оснований $1 \leqslant n \leqslant 10$, массиву простых чисел p и двум массивам остатков x, y находит массив остатков для суммы $x + y$.

Техническое требование

Файл *input.txt* в первой строке содержит n, во второй — массив p, в третьей и четвертой, соответственно, x и y. Все числа в массивах — неотрицательные, целые, не превосходящие $2^{15}-1$, разделены пробелами.

В файл *output.txt* должны быть записаны три строки. В первой стоит 0, если сумма не превзошла $p_1p_2\ldots p_n - 1$ и 1 — в противном случае.

Во второй – последовательность остатков для $x+y$, разделенных пробелами. В третьей — слово *End.*

Пример

input.txt:

```
2
3 5
```

```
 2 4
 1 3
```
output.txt:
```
 1
 0 2
 End.
```

16.129. **Дискет должно хватить.** Для того, чтобы переписать новую компьютерную игру, нужно много дискет. Но все же хочется использовать дискет поменьше. Обычно используют архивацию с разбиением на тома, но сейчас нет доступного архиватора. Кроме того, для защиты от плохих кластеров используют не все пространство дискеты.

Техническое требование

Дан список файлов, полученный командой `dir>filelist` в операционной системе Windows 95/98.

Программа должна создать bat-файл *put.bat*, который будет копировать заданные файлы на дискеты в устройстве `A:`, причем дискет используется наименьшее количество. Дискеты — стандартного размера. Дискет понадобится не более 20. Размеры файлов — не более 1Гб. Для смены диска bat-файл должен содержать команду `pause`. Для записи файлов на дискете есть 1300Кб. Все файлы расположены в одном каталоге и их количество не более 500.

Примечания

Файлы нельзя разделять на части. Команда dir выполняется в правильной файловой системе. Стандартные дискеты используют разбиение на кластеры по 512 байт и файл всегда использует целое количество кластеров.

Пример

В файле filelist:
```
 Том в устройстве D имеет метку FILESYS
 Серийный номер тома: 3410-2CE0
 Содержимое папки D:\GAMES\Lines98

.              <ПАПКА>      21.08.00  11:50 .
..             <ПАПКА>      21.08.00  11:50 ..
LINES98  HLP         8 444  03.09.99  13:19 lines98.hlp
LINES98  EXE        58 880  30.11.99  21:10 lines98.exe
LINES98  SKN       883 451  03.09.99  13:53 lines98.skn
LEADERS  DAT           558  28.09.00  21:51 leaders.dat
BALLS    BMP       939 254  21.06.99  18:53 BALLS.BMP
BKG      BMP        40 890  03.09.99  12:00 BKG.BMP
```

```
CANTMOVE WAV        29 914  23.05.99  19:32 cantmove.wav
DESTROY  WAV           800  07.10.98  14:17 destroy.wav

Итоговые данные
         8 файлов      1 962 191 байт
         5 папок     365 105 152 байт свободно
```

в put.bat:

```
echo Вставьте дискету и нажмите <ПРОБЕЛ>
pause
copy lines98.skn a:
copy lines98.hlp a:
copy lines98.exe a:
copy leaders.dat a:
copy BKG.BMP a:
copy cantmove.wav a:
copy destroy.wav a:
echo Вставьте дискету и нажмите <ПРОБЕЛ>
pause
copy BALLS.BMP a:
echo копирование окончено, занято 2 дискеты
```

Как оценивалось задание на соревновании

Оценивались программы, работающие не более 20 секунд на компьютерах класса P-200. Далее программы тестировались на примере и на достаточно простом тесте жюри. Между программами, прошедшими эти тесты, было устроено соревнование на нескольких тестах. Программа выигрывала, если для копирования ей понадобится меньшее число дискет. Если требовалось одинаковое количество дискет поровну, выигрывала та, у которой осталось больше места на последней дискете.

16.130. **Пополнение архивов.** Другой заголовок — Дисков все равно должно хватить!

Написать программу пополнения архивов на машинных носителях информации (дисках, и т.п.). Основная задача — разместить изменившиеся и новые архивы, использовав наименьшее число новых носителей информации и сделав при этом наименьшее число перемещений архивов с одного носителя на другой.

Входные данные задаются в виде файла INPUT.TXT с числами в десятичной записи, расположенными в следующем порядке:

```
емкость носителя>
<число носителей (m)>
```

```
<число архивов на 1-м носителе (n)>
<начальный размер 1-го архива> <изменившийся размер 1-го архива>
...
<начальный размер n-го архива> <изменившийся размер n-го архива>
... ... ...
<число архивов на m-м носителе>
<начальный размер 1-го архива> <изменившийся размер 1-го архива>
...
<начальный размер n-го архива> <изменившийся размер n-го архива>
```

Все размеры даются в некоторых единых условных единицах.

Выходные данные задаются в виде файла OUTPUT.TXT с числами в десятичной записи, расположенными в следующем порядке.

```
<число архивов, перемещаемых с 1-го носителя>
<номер архива> <номер нового носителя>
...
<номер архива> <номер нового носителя>
... ... ...
<число архивов, перемещаемых с m-го носителя>
<номер архива> <номер нового носителя>
...
<номер архива> <номер нового носителя>
```

В качестве номеров новых носителей можно использовать как числа в диапазоне от 1 до m, так и бо̀льшие числа, если нужны новые носители.

Размеры архивов и носителей даются в диапазоне от 0 до 1000000000, номера носителей и архивов — от 1 до 10000.

Среди программ устраивается соревнование на тестах. Лучшими признаются программы, не сделавшие ошибок, обрабатывающие бо̀льшие количества архивов, добившиеся наиболее плотного хранения архивов, сделавшие при этом наименьшее количество перемещений. На работу программы отводится 10 секунд (процессор Pentium-200).

Пример

Пример входных данных (input.txt):

```
1440
2
2
300 600
800 900
2
300 300
400 500
```

Это означает, что размер носителя — 1440, число носителей — 2; на первом носителе было 2 архива, первый из которых увеличился с 300 единиц до 600, второй — с 800 единиц до 900; на втором носителе было 2 архива, первый, в 300 единиц, не изменил свой размер, а второй увеличился с 400 единиц до 500 единиц.

Пример выходных данных (output.txt):

```
1
1 2
0
```

Это означает, что первый архив с первого носителя нужно переместить на второй носитель, а архивы со второго носителя не перемещать.

16.131. **Синие и красные многоугольники.** Дан ряд многоугольников на плоскости с непересекающимися границами. Требуется раскрасить эти границы в 2 цвета (синий и красный) так, чтобы если один многоугольник находится непосредственно внутри другого, то их границы были бы раскрашены в разные цвета. Границы всех внешних многоугольников красятся только в синий цвет.

Требуется написать программу раскраски границ многоугольников.

Формат входных данных

Входные данные заданы в виде файла *INPUT.TXT*, содержащего

- число многоугольников,
- для каждого многоугольника:
  - число вершин
  - для каждой вершины (в порядке соединения вершин) — ее координаты (x и y), последняя вершина соединяется с первой

Числа разделены пробелами и/или переводами строк.

Ограничения:

Число многоугольников — не более 100.

Общее число вершин — не более 10000.

Координаты — целые неотрицательные числа, меньшие 100.

Формат выходных данных

Файл OUTPUT.TXT с последовательностью кодов цветов границ (в порядке ввода).

Код синего цвета — 0, код красного цвета — 1.

Коды разделены пробелами.

Если нет решений, файл содержит единственное число 2.

Если входные данные содержат ошибку, то файл содержит единственное число 3.

Время решения не более 20 секунд.

Пример

Входные данные

```
3
4
0 0
0 5
5 5
5 0
4
1 1
1 4
4 4
4 1
4
2 2
2 3
3 3
3 2
```

Выходные данные

```
0 1 0
```

16.132. **Игра в ящик.** Игра проходит на прямоугольном поле шириной 31 и длиной 19. Таким образом, все поле разделено на квадраты площадью 1. В центре поля находится квадратная яма площадью 1 и глубиной 1. У каждого игрока есть ящик, представляющий собой прямоугольный параллелепипед с квадратным основанием площадью 1. Высота ящика выбирается игроком до начала игры произвольно, но не меньше 1. Ход представляет собой кантование (переваливание через ребро) ящика. Причём края ящика всегда должны совпадать с границами квадратов. При этом ящик не должен выставляться за пределы поля и ложиться на ящики противников. Начальное положение ящиков может выбираться жюри.

Цель игры

Поставить свой ящик основанием в яму. Очевидно, это сможет только один игрок, он и выигрывает.

Задача

Для решения задачи понадобится написать две программы. Первая программа должна будет выбрать высоту ящика. Вторая — делать ход.

Техническое требование

Первая программа должна прочитать файл *square.txt* и дописать ту-

да очередную строчку из первых четырех (см. формат *square.txt*). Если файл пуст, то первый ход Ваш и программа должна вписать строку из единиц.

Вторая программа должна прочитать файл *square.txt*. Результатом работы программы должен быть изменённый файл *square.txt* с положением ящиков после хода игрока записанного в первой строке *square.txt*. Измениться должны так же первые четыре строки. Первая строка должна оказаться последней, а все остальные должны сдвинуться вверх без изменения порядка.

Начальное положение ящиков на поле будет всегда такое, как показано в примере. Играть будут четыре игрока и их ящики будут стоять по углам.

Игрок может пропускать ход, только если ему некуда сходить, не нарушая правил.

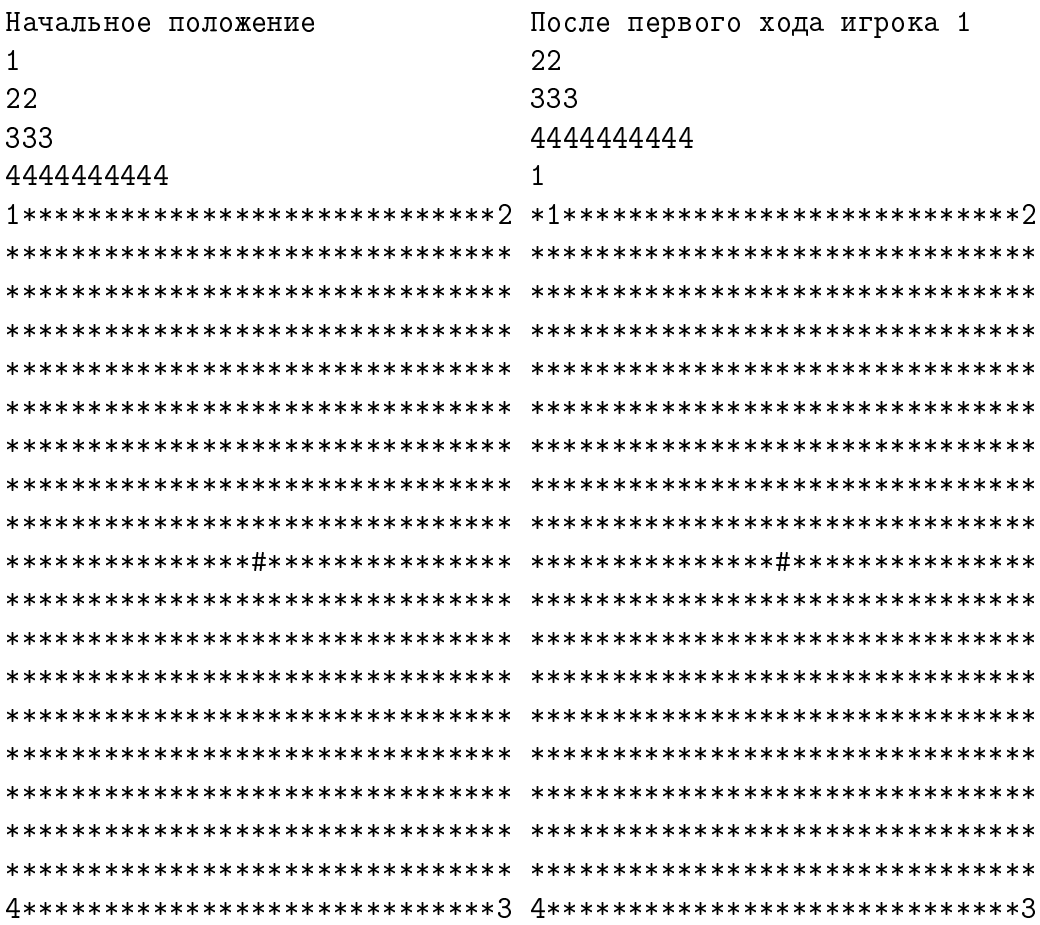
Игра продолжается не более 100 ходов.

Время на один ход не более 30 секунд.

Формат файла *square.txt*

Первые четыре строки содержат информацию о размерах ящиков и очерёдности хода четырёх игроков. Как видно из примера, Номер строки — очерёдность хода, цифра — номер игрока, количество символов в строке — высота ящика.

Далее 19 строк по 31 символу описывают поле с ящиками.


Начальное положение
1
22
333
4444444444
После первого хода игрока 1
22
333
4444444444
1

```
После первого хода игрока 2      После первого хода всех игроков
333                              1
4444444444                       22
1                                333
22                               4444444444
*1**************************** *1****************************
****************************2 ****************************2
****************************2 ****************************2
**************************** ****************************
**************************** ****************************
**************************** ****************************
**************************** ****************************
**************************** ****************************
**************************** 4***************************
***************#************ 4**************#************
**************************** 4***************************
**************************** 4***************************
**************************** 4***************************
**************************** 4***************************
**************************** 4***************************
**************************** 4***************************3
**************************** 4***************************3
**************************** 4***************************3
4***************************3 ****************************
```

Так оценивалась задача на соревновании

Среди программ был устроен турнир в два этапа. Первый — отборочный с программами жюри, разыгрывалось 10 баллов. Программы, не нарушающие правил и выигравшие хотя бы одну программу жюри допускались ко второму этапу. Во втором этапе разыгрывались 90 баллов. Второй этап — турнир всех программ. Итого можно было набрать 100 баллов.

16.133. Новая жизнь.

Краткое описание задачи

Требуется найти короткие циклы в игре в «жизнь по новым правилам».

Игра «жизнь» проводится на двумерном поле, разбитом на равные квадратные клетки. В каждой клетке может находиться организм в одном из своих состояний. Состояния обозначаются цифрами и латинскими буквами (строчные и прописные — различные состояния). Отсутствие

организма в клетке обозначается пробелом. Состояния в клетках меняются одновременно в выделенные моменты времени, обозначаемые целыми числами. Начальные состояния клеток задаются в момент времени 0. Все следующие состояния клеток определяются по правилам игры. Каждое правило игры ставит новое состояние клетки в соответствие состояниям этой клетки и ее соседей в предшествующий момент времени. Соседями каждой клетки считаются ближайшие к ней 8 клеток. Если какой-то комбинации состояний клеток не ставится в соответствие ни одно новое состояние, то считается, что организм умирает, и его состояние заменяется пробелом. Правила не должны противоречить друг другу.

Входные данные

Файл *rules.txt* содержит список правил в виде десятисимвольных строк: 1-й символ — новое состояние, далее 9 символов АБВГДЕЖЗИ, описывающих предшествующие состояния клетки ('Д') и ее соседей, расположенных в виде конфигурации

```
АБВ
ГДЕ
ЖЗИ
```

Файл *init.txt* содержит некоторое число N N-символьных строк, составленных из пробелов, цифр и латинских букв. Этот массив изображает начальные состояния клеток (в остальных клетках — пробелы).

Выходные данные

Выходной файл *loop.txt* содержит два различных номера моментов времени в интервале от 0 до N, на которых возникает первое повторение всех состояние клеток. Если таких чисел нет, то выдаются два нуля. Между числами ставится пробел.

Дополнительные условия

Время работы программы — 10 секунд. Лучшими считаются программы, правильно обрабатывающие за это время наиболее длинные тесты (по суммарной длине входных файлов).

Пример

(пробелы заменены знаком '-')

rules.txt

```
1----1----
1-1-111-1-
```

init.txt
```
-----
--1--
-111-
--1--
-----
```
loop.txt
```
1   2
```

16.134. **Трис.** В квадратный двумерный стакан со стороной размером в 3 «клетки» падают плоские фигурки, каждая из которых составлена из трех квадратов размером в одну клетку. Квадратики в фигурках соединены в сторонами так, что эти стороны полностью совпадают. Таким образом, имеются две формы фигурок:

```
[][][]
```

и

```
[][]
[]
```

Внутри стакана они могут падать только вертикально вниз. При этом стороны квадратов, из которых составлены фигурки, должны быть направлены параллельно сторонам стакана. Фигурка может упасть вдоль любых вертикальных рядов клеток в стакане. При этом ей нельзя выходить за боковые края стакана. Фигурка падает до столкновения с дном стакана или с занятым квадратом. После падения фигурки должны умещаться в стакане (не выходить за верхний край). Если после падения очередной фигурки один из горизонтальных рядов стакана оказывается полностью занят квадратиками фигурок, то этот ряд квадратиков удаляется, а все расположенные выше квадратики сдвигаются на одну клетку вниз. Этот процесс при «удачном» падении фигурок может продолжаться сколь угодно долго.

Последовательность фигурок с их ориентацией и занимаемыми ими вертикальными рядами при падении в стакан называется «способом падения фигурок». Количество упавших фигурок называется «длиной» этого способа. Способ называется «чистым», если в конце процесса остается пустой стакан. Число чистых способов падения фигурок с заданной длиной называется «разнообразием способов падения», обеспечиваемой этой длиной.

Задача

Задача состоит в определении наименьшей длины, обеспечивающей не менее, чем заданное разнообразие способов падения.

Техническое требование

Заданное разнообразие способов падения находится в файле *input.txt*. Оно не превышает двух миллиардов.

Результат следует поместить в файл *output.txt*.

Время решения — одна секунда на компьютере с процессором Pentium-200.

Пример

```
input.txt: 3
output.txt: 2
```

16.135. **Выбор ящика.** Игра в ящик проходит на прямоугольном поле шириной N и длиной M. Точно в центре поля находится квадратная яма площадью 1 и глубиной 1. У каждого игрока есть ящик, представляющий собой прямоугольный параллелепипед с квадратным основанием площадью 1. Высота ящика выбирается игроком до начала игры произвольно, но не меньше 1. Ход представляет собой кантование (переваливание через ребро) ящика. При этом ящик не должен выставляться за пределы поля и ложиться на ящики противников. Начальное положение ящиков может выбираться жюри.

Цель игры

Поставить свой ящик основанием в яму. Очевидно, это сможет только один игрок, он и выигрывает.

Задача

По заданным размерам поля (M и N) определить размер ящика и один из вариантов ходов для передвижения ящика из левого верхнего угла в яму. Программа считается лучшей, если ящик оказывается в яме за меньшее количество ходов. На поле нет ни одного препятствия и противника.

Техническое требование

Первая программа должна читать файл square.txt, где в первой строке записано число $0 < s < 40$ ($N = 2s + 1$) , а во второй — $0 < w < 10$ ($M = 2w + 1$). После этого туда нужно дописать с новой (третьей) строки длину ящика (не более 100). В четвёртой строке должно быть число ходов, необходимое для попадания ящика в яму. В пятой строке — последовательность символов: 'L', 'R', 'U' или 'D', — направления перекантовывания ящика, обозначают «влево», «вправо», «вверх» и «вниз», соответственно.

Время работы программы не более 30 секунд.

Пример

В *square.txt* до работы программы:

```
1
3
```

В *square.txt* после работы программы:

```
1
3
2
3
DRD
```

16.136. **Охрана склада.** Склад представляет собой площадь 20 на 30 метров. Каждый квадратный метр склада либо полностью пуст, либо полностью занят тюком. Тюк выглядит как прямоугольный параллелепипед с основанием 1×1 метр. Этим основанием тюк устанавливается точно на один квадратный метр склада. Стоящие рядом тюки стоят так плотно, что через промежуток ничего не видно, даже если тюки стоят по диагонали. По периметру склад огорожен забором.

В ночное время склад охраняется при помощи устройств «ОКО».

Устройство «ОКО» (дальше ОКО) реагирует на движение в одном из направлений строго по прямой. Таким образом, это устройство способно обеспечить охрану одного ряда квадратных метров склада начиная со следующей клетки от места своего расположения и до первого препятствия. Сами устройства небольшие и препятствием не являются. Для эффективной работы ОКО направляют только вдоль или поперёк склада, но никогда по диагонали. На один квадратный метр можно устанавливать только одно ОКО.

Требуется

Написать программу, которая по расположению тюков укажет, как нужно установить ОКО.

Входные данные (pack.txt)

Расположение тюков хранится в файле *pack.txt* в виде 20 строк по 30 символов в каждой. Занятое тюком место обозначается '#', свободное — '␣'.

Выходные данные (oko.txt)

Результат работы программы записывается в файл *oko.txt* формат такой же, как у файла *pack.txt*, но корме тюков на пустых местах должны быть указаны расположения и направления ОКО. Символы для их обозначения следующие. '<' — направлено влево, '>' — вправо, 'V' — вниз, '^' — вверх.

Пример

```
pack.txt                          oko.txt
##############################  ##############################
##############################  ##############################
####                        ##  ####>         V           V##
#### ########## ########## ##  #### ########## ########## ##
#### ########## ########## ##  #### ########## ########## ##
#### ########## ########## ##  #### ########## ########## ##
#### ########## ########## ##  #### ########## ########## ##
#### ########## ########## ##  #### ########## ########## ##
#### ########## ########## ##  #### ########## ########## ##
####                        ##  ####^                     <##
##############################  ##############################
############ ################  ############ ################
############ ################  ############ ################
############ ################  ############ ################
############ ################  ############ ################
############ ################  ############V################
############ ################  ############ ################
############ ################  ############ ################
############ ################  ############ ################
############ ################  ############^################
```

Оценка

Верной программой считается та, которая будет расставлять ОКО так, что всё свободное пространство склада будет находиться под контролем. Среди верных программ лучшей считается та, которая затратит меньше устройств.

Техническое требование

Программа должна работать не более 50 секунд на компьютерах класса Pentium-200. Для работы программа может рассчитывать на 1 Мб оперативной памяти и 2 Мб дисковой.

16.137. **Графический конвертор.** Конвертировать файл из формата команд графопостроителя в формат XPM.

Формат входных данных: текстовый файл *input.gpc*, содержащий последовательность команд графопостроителя. Каждая команда представляет собой последовательность символов, среди которых могут присутствовать только прописные буквы, цифры, запятая и знаки '+' и '-'. Команда всегда завершается символом «точка с запятой» (';'). Между командами могут присутствовать одиночные переводы строк. Конвертор

должен интерпретировать только описанные ниже команды. Все остальные команды игнорируются.

Синтаксис интерпретируемых команд:

```
<команда>::=<относительное перемещение>
           |<абсолютное перемещение>
           |<опустить перо>
           |<поднять перо>
<относительное перемещение>::= PR<сдвиг по горизонтали>,
                                <сдвиг по вертикали>
<абсолютное перемещение>::= PA<горизонтальная координата>,
                             <вертикальная координата>
<опустить перо> ::= PD
<поднять перо> ::= PU
<сдвиг по горизонтали> ::= <сдвиг>
<сдвиг по вертикали> ::= <сдвиг>
<сдвиг> ::= +<число без знака>|-<число без знака>
<горизонтальная координата> ::= <число без знака>
<вертикальная координата> ::= <число без знака>
<число без знака> ::= <цифра>|<две цифры>|<три цифры>
                     |<четыре цифры>
<цифра> ::= 0|1|2|3|4|5|6|7|8|9
<две цифры> ::= <цифра><цифра>
<три цифры> ::= <цифра><две цифры>
<четыре цифры> ::= <цифра><три цифры>
```

Первая интерпретируемая команда — «абсолютное перемещение», задающее начальное положение пера графопостроителя. Начало координат $(0, 0)$ считается размещенным в верхнем левом углу рисунка. Отрицательные координаты недопустимы. Положительными считаются сдвиги вниз по вертикальной оси и вправо по горизонтальной оси. Считается, что изображение наносится на белом поле черным цветом опущенным пером. Поднятое перо не рисует.

Пример входного файла:

```
PA2,3; PD;PR+2,+0; PD;PR+0,-1; PU;PA2,2;
```

Формат выходных данных:

Текстовый файл *output.xpm* вида

```
#define output_format 1
#define output_width <ширина изображения>
#define output_height <высота изображения>
#define output_ncolors 2
#define output_chars_per_pixel 1
static char *output_colors[] =
{ "a", "#000000", "b", "#ffffff" };
static char *output_pixels[] =
{ "<строка>", "<строка>", ... "<строка>", "<строка>" };
```

где каждая «строка» состоит из букв 'a' и 'b', 'a' соответствует черной точке, 'b' соответствует белой точке. Длина строки равна ширине изображения, число строк равно высоте изображения.

Примечание: размер выходного изображения должен быть минимизирован: изображение (черные точки) вписывается в наименьший прямоугольник со сторонами, параллельными осям координат (без поворотов изображения). Линии должны не иметь разрывов. Движению вправо соответствует переход к следующим позициям строк, движению вниз — переход к следующим строкам.

Пример выходного файла:

```
#define output_format 1
#define output_width 3
#define output_height 2
#define output_ncolors 2
#define output_chars_per_pixel 1
static char *output_colors[] = {
   "a", "#000000", "b", "#ffffff"
};
static char *output_pixels[] = {
   "bba", "aaa"
};
```

16.138. **Осчастливливание числа.** Как известно, счастливым числом называется натуральное число, у которого сумма цифр первой половины равна сумме цифр второй половины. К числам с нечетным количеством цифр будем приписывать один незначащий ноль, что сделает количество цифр чётным.

Задача. Прибавить к заданному числу минимально возможное неотрицательное число так, чтобы сумма получилась счастливой.

Техническое требование

Числа до 80 десятичных цифр. Ввод исходного числа с клавиатуры и вывод получившегося счастливого числа на экран.

ВРЕМЯ РЕШЕНИЯ — 5 секунд.

Пример

```
Число: 12340004
Сумма: 12340019
```

16.139. Ломаный график.

Формулировка задачи

Записать формулу для функции, график которой задан ломаной линией, используя сложение, вычитание, умножение, деление и модуль.

Входные данные находятся в файле *input.txt* в следующем формате:

```
<число вершин n>
<абсцисса 1> <ордината 1> ... <абсцисса n> <ордината n>
```

где <число вершин>, <абсцисса> и <ордината> — натуральные числа, заданные конечными последовательностями десятичных цифр (не более четырех) без знаков.

Выходные данные следует поместить в файл *formula.h* в следующем формате:

```
f(x)=<выражение>
```

где

```
<выражение>::=<терм>|<выражение>+<терм>|<выражение>-<терм>
<терм>::=<множитель>|<терм>*<множитель>|<терм>/<множитель>
<множитель>::=x|<число>|(<выражение>)|abs(<выражение>)
<число>::=<цифры>.<цифры>
<цифры>::=<цифра>|<цифры><цифра>
<цифра>::=0|1|2|3|4|5|6|7|8|9
```

Ограничения:

1) общее число цифр в числе должно не превышать 7;
2) глубина вложенности скобок должна не превышать 3;
3) число лексем выражения не может превышать число вершин ломаной линии более чем в 20 раз (лексемой называется число, переменная, знак операции (в том числе, и abs) или скобка);
4) формула должна давать значение функции с точностью не менее 0.001.

Пример

Файл *input.txt*:

```
3 0 1 1 0 2 1
```

Файла *formula.h*:

```
f(x)=abs(1.0-x)
```

16.140. **СТРОКА-2** Лаборатория по обработке текстов разработала язык программирования СТРОКА-2. Этот язык предназначен для обработки строк. Он устроен так же, как язык СТРОКА (см. задачу 14.17). Для данного языка создан интерпретатор ***string***.

Синтаксис языка СТРОКА-2

Все команды записываются в виде: A r B, где A, B — строки или переменные, r — операция языка. Команда означает, что нужно вычислить результат операции r для операндов A и B, после чего записать этот результат в переменную Z.

Переменными языка являются только заглавные латинские буквы. Переменные A и Z имеют специальные значения. A — хранит входную цепочку. Значение Z будет выведено после окончания работы интерпретатора и считается результатом.

Константами языка являются любые строки. Константы заключаются в кавычки. Строки в тестах будут текстовые, без управляющих символов и одинарных кавычек.

Команды этого языка записываются по одной в строке.

В отличие от языка СТРОКА в языке СТРОКА-2 имеются только одна операция

$x < y$ — заменить первую слева подстроку y в z на x.

Задача

Написать программу для построения программы на языке СТРОКА-2 по известным тестам. Тесты будут находится в файле *test.txt*. Построенная программа записывается в файл *program.str*. Эта программа должна проходить как можно больше тестов.

Формат записи тестов в файле *test.txt*

В файле с тестами каждая нечетная строка — это входные данные, а четная — выходные.

Техническое требование

Тестов в файле будет не больше 10000, т.е. строк до 20000. Длины строк до 100 символов. Размер файла с построенной программой не должен превышать 100 Мб. Каждая строка до 200 символов. Количество строк не ограничивается. Константы будут текстовые, без управляющих символов. Вторая одинарная кавычка обозначает конец константы. Время работы программы не должно превышать 50 секунд на Р-200. Памяти

дисковой и оперативной будет выделено не менее 1 Мб.

Пример

Тесты

```
abc
b
aaaaa
aaa
123456
2345
```

Программа

```
Z < A
'abc' < 'b'
'aaaaa' < 'aaa'
'123456' < '2345'
```

ОЦЕНКА ЗАДАЧИ

Программы будут оцениваться по количеству пройденных тестов созданной ими программы.

16.141. **Забор яблоневого сада.** К этому времени должна быть решена задача 14.24.

Отличие этой задачи 14.24 описаны ниже.

Техническое требование

Расположение узла с деревом задаётся двумя числами, номерами линий, при пересечении которых получается этот узел. Расположение всех яблонь задается в файле *orchard.txt* следующим образом. В первой строке записано одно число — количество яблонь (не менее одной). Начиная со второй строки, записаны пары чисел — расположение узлов. Пар столько же, сколько и яблонь. Числа разделены пробелами. Некоторые яблони могут быть указаны несколько раз. В файле fence.txt записаны пары чисел — узлы, где должны быть углы забора в произвольном порядке. Нужно написать программу, которая читает файлы *orchard.txt* и *fence.txt*. Результатом работы программы не позже чем через 10 секунд должен быть квадрат длины одного пролёта забора. Если в *fence.txt* записаны сведения нарушающие условия 1–4, но нужно выдать число 0. Ответ выдавать на экран.

Пример

1)

```
B orchard.txt     в fence.txt
 4                 0   0
 1   1             14  17
 5   6             30  30
30  30             17  14
 5   6
```

Ответ: 0

2)

```
B orchard.txt     в fence.txt
3                   0   0
 1   1             14  17
 5   6             17  14
30  30             31  31
 5   6
```

Ответ: 485

16.142. **Цветная линия.** Для игры в цветную линию используются шарики семи цветов.

Несколько шариков выстраиваются в линию. Если рядом оказались шарики одинакового цвета, они убираются (все сколько есть). Если есть несколько групп одноцетных шариков, то убирается самая левая. Оставшиеся шарики сдвигаются так, чтобы не осталось пустого места. Если после сдвига опять окажется, что есть рядом одинаковые шарики, то операция повторяется. После того как не осталось одинаковых шариков рядом, выполняется ход. Можно переставлять любой шарик из линии в её конец. Шарики сдвигаются. После чего всё продолжается с получившейся линией.

Цель игры

Оставить как можно меньше шариков за минимум ходов.

Техническое требование

В файле *line.txt* записаны две строки. Первая число — длина исходной линии (не более 80). Вторая — последовательность цветов шариков в этой линии. Цвета обозначются буквами: 'B' (синий), 'G' (зелёный), 'C' (голубой), 'R' (красный), 'M' (малиновый), 'Y' (жёлтый), 'W' (белый).

Задача. Написать программу, которая по заданной начальной линии запишет в файл moves.txt последовательность ходов для получения короткой линии шариков. Последовательность должна содержать не более 100 чисел. Файл moves.txt устроен следующим образом. В каждой строке записано одно число — номер шарика от начала, который нужно

переставить в конец.

Оценка задачи

Между программами будут устроены соревнования. В первую очередь будет оцениваться длина оставшейся линии, и при равных длинах — количество ходов.

Пример

В line.txt

```
10
BGCRCMYWWY
```

В moves.txt

```
3
4
```

Сама игра

```
BGCRCMYWWY
```

Перед ходом

```
BGRCMC
```

Ход 1

```
BGRCMC
```

Ход 2

```
BGRMCC
```

Результат:

```
BGRM
```

Итого: за два хода удалось получить длину 4.

16.143. **Отремонтирована ли стиральная машина?** Вспомните задачу 14.23.

Вася починил свою стиральную машину. Но он боится, что он ее неправильно собрал. В случае ошибки может произойти короткое замыкание и некоторые из блоков могут снова выйти из строя. К счастью, Вася знает, какие соединения блоков стиральной машины ведут к аварии, то есть что с чем не должно быть соединено одновременно. Вася знает электрическую схему, которую он получил при сборке стиральной машины. Помогите Васе проверить, не приведет ли его сборка к аварии. Ваша задача состоит в написании программы, определяющей по схеме сборки машины и ее программе приводит ли эта сборка к аварии, а если приводит, то на каком шагу работы программы.

Исходные данные содержатся в файле *input.txt* в следующем виде:
t — длина программы машины (число моментов времени в ней),
c[1] — число соединений в момент времени 1,
x[1,1] y[1,1] ... x[1,c[1]] y[1,c[1]] — информация о соединениях в момент времени 1 (точка номер x[1,1] соединена с точкой номер y[1,1], ..., точка

номер x[1,c[1]] соединена с точкой номер y[1,c[1]]),
...
c[t] — число соединений в момент времени t,
x[t,1] y[t,1] ... x[t,c[t]] y[t,c[t]] — информация о соединениях в момент времени t,
k — число запрещенных сочетаний соединений,
b[1] — число соединений в запрещенном сочетании номер 1,
z[1,1] u[1,1] ... z[1,b[1]] u[1,b[1]] — информация о соединениях в запрещенном сочетании номер 1 (нельзя одновременно соединить точку z[1,1] с точкой u[1,1], ... и точку z[1,b[1]] с точкой u[1,b[1]]),
...
b[k] — число соединений в запрещенном сочетании номер k,
z[k,1] u[k,1] ... z[k,b[k]] u[k,b[k]] — информация о соединениях в запрещенном сочетании номер k.

Замечание: если две точки соединены с третей, то они считаются соединенными между собой.

Выходные данные записываются в файл *output.txt* в следующем виде: 0, если аварии не происходит, иначе файл содержит номер момента времени, в который авария происходит ВПЕРВЫЕ.

Технические ограничения: номера точек для соединения могут принимать значения от 1 до 200, число соединений — не более 40000, число моментов времени и число k — не более 10000. Все числа в *input.txt* разделяются пробелами и/или переводами строк, после последнего числа — конец файла с возможными предшествующими пробелами и/или переводами строк. Время решения ограничено 20 секундами. Вспомогательные файлы использовать запрещается. Доступная оперативная память — 16M.

Пример

Правильное содержимого *input.txt*:

```
1
1
1 2
1
1
1 2
```

Здесь t=1, c[1]=1, x[1,1]=1, y[1,1]=2, k=1, b[1]=1, z[1,1]=1, u[1,1]=2.

Пример соответствующего содержимого output.txt:

```
1
```

16.144. **Водопровод.** Водопроводная сеть состоит из труб разной

пропускной способности.

Вода по любой трубе может течь в любую сторону. Все места соединения труб учтены и пронумерованы целыми числами от 0 до 10000. Особо выделены места с номерами 0 и 1. На самом деле, 0 это не соединение труб, а место, в котором подаётся вода в трубы, а 1 — место, откуда вода вытекает. У каждой трубы известна пропускная способность — максимальное количество воды протекающей по трубе, за единицу времени, при котором она ещё не лопнет.

Задача

Цель: добиться того чтобы в узел 1 текло как можно больше воды. Определить необходимое количество воды, протекающей по каждой трубе (напор) для достижения цели. Превышать пропускную способность трубы нельзя, а то труба может лопнуть.

Техническое требование

Поток воды в трубе должен быть целым числом. В файле *pipes.txt* находится описание соединения. Ответ нужно записать в файл *water.txt* в указанном формате.

Формат pipes.txt

```
<Номер трубы> <узел 1> <узел 2> <пропускная способность>
<Номер трубы> <узел 1> <узел 2> <пропускная способность>
...
<Номер трубы> <узел 1> <узел 2> <пропускная способность>
```

Формат water.txt

```
<Номер трубы> <узел 1> <узел 2> <напор>
<Номер трубы> <узел 1> <узел 2> <напор>
...
<Номер трубы> <узел 1> <узел 2> <напор>
```

Пример

pipes.txt:

```
1 0 2 1
3 3 2 10
5 3 1 9
2 0 3 10
4 2 3 10
6 3 1 9
```

water.txt:
```
1 0 2 0
2 0 3 10
3 2 3 1
4 3 2 1
5 3 1 9
6 3 1 1
```

Оценка задачи

Задача оценивались жюри соревновательно. Максимальное количество баллов получали решения с максимальным количеством воды поступающей в узел (приток) 1. Из решений с одинаковым притоком считалось лучшим то, в котором присутствовало больше труб с нулевым напором. Максимальное количество баллов за задачу: 100.

16.145. **Игра «Три кучки».** Двое игроков играют в следующую игру. Перед ними — три кучки камней (в каждой кучке — не менее 2 камней). Каждый из игроков во время своего хода может удалить любую кучку, а камни любой другой разложить в виде прямоугольника со сторонами в M и N камней (M и N больше единицы), и заменить эту кучку на две: первая — из M камней, вторая — из N камней. Игроки ходят по очереди. Проигрывает тот, кто не может сделать указанное действие.

Требуется написать программу, которая делает очередной ход в игре.

Техническое требование

Имя входного файла: *input.txt*

Имя выходного файла: *output.txt*

Ограничение по времени: 1 секунда.

Формат входных данных

Входной файл содержит три целых положительных числа — начальную игровую позицию. Каждое число не превышает 32767. Числа записаны в десятичной форме и разделены пробелами.

Формат выходных данных

Выходной файл должен содержать в первой строке цифру 1, если ход возможен, или цифру 2, если это не так. Если ход возможен, то во второй строке должны содержаться три числа: номер удаляемой кучки, номер заменяемой кучки и число камней в первой из заменяющих кучек.

Пример

input.txt
```
2 2 4
```
output.txt
```
1
```

```
1 3 2
```

16.146. **Водопроводчик.** Перед решением этой задачи нужно решить задачу 16.144.

Изменение в предыдущей формулировке

Написать программу, которая будет проверять ответы, полученные для входных данных для первого тура, на правильность. Если ответ правильный, вычисляется приток в 1 узел. Если ответ содержит ошибку, то указать, в каком узле или трубе и что не верно. Формат файла *water.txt* будет верным, это можно не проверять.

Время работы программы 1 секунда.

16.147. **Три кучки-2.** Перед решением этой задачи полезно решить задачу 16.145 и воспользоваться её решением.

Двое игроков играют в следующую игру. Перед ними — три кучки камней (в каждой кучке — не менее 2 камней). Каждый из игроков во время своего хода может удалить любую кучку, а камни любой другой разложить в виде *контура прямоугольника* со сторонами в M и N камней (M и N больше единицы), и заменить эту кучку на две: первая — из M камней, вторая — из N камней. Игроки ходят по очереди. Проигрывает тот, кто не может сделать указанное действие.

Требуется написать программу, которая делает очередной ход в игре.

Техническое требование

Имя входного файла: *input.txt*

Имя выходного файла: *output.txt*

Ограничение по времени: 1 секунда.

Формат входных данных

Входной файл содержит три целых положительных числа — начальную игровую позицию. Каждое число не превышает 32767. Числа записаны в десятичной форме и разделены пробелами.

формат выходных данных

Выходной файл должен содержать в первой строке цифру 1, если ход возможен, или цифру 2, если это не так. Если ход возможен, то во второй строке должны содержаться три числа: номер удаляемой кучки, номер заменяемой кучки и число камней в первой из заменяющих кучек.

Пример

input.txt

```
2 2 4
```

output.txt

```
1
```

```
1 3 2
```

Примечание

Число набранных баллов определяется в результате турнира между программами участников и жюри.

16.148. **Пейджинговый роуминг.** Известна проблема с роумингом для пейджеров. Нужно, чтобы абонент получал назначенное для него сообщение, находясь рядом с любой антенной, но у пейджера нет обратной связи, и определить, где он находится, невозможно. Приходится передавать сообщение по всем станциям и передавать его в эфир на каждой станции. Проблема состоит в том, что две станции могут передавать сообщения одновременно третьей. И если третья станция окажется на одинаковом расстоянии от этих двух, могут возникнуть дублирование. Здесь предлагается подход для решения этой проблемы.

Давайте сразу строить станции так, чтобы не было двух, расположенных на одинаковом расстоянии от третьей. Тогда описанной ситуации не возникнет.

Техническое требование

Наша задача — поставить максимально возможное число станций на прямоугольной территории. Все вычисления будем вести в целых метрах. Тогда любое место для станции может иметь координаты — целые числа X от 0 до N и Y от 0 до K ($K, N < 1000$).

Программа считывает из файла map.txt два числа N и K и записывает в файл antenna.txt информацию о расположении станций в следующем виде:

```
<S - количество станций>
<Координата X первой станции> <Координата Y первой станции>
...
<Координата X S-ой станции> <Координата Y S-ой станции>
```

Время работы программы 10 секунд.

Пример

map.txt

```
3 4
```

antenna.txt

```
4
0 0
0 1
0 3
3 4
```

Задача соревновательная. Максимальное количество баллов получает программа, разместившая максимальное количество станций.

16.149. **Квадрат чисел.** Задан квадрат нечетного размера из чисел. Движение начинается из центра. Двигаться можно вправо, влево, вверх и вниз. По ходу движения суммируются числа, записанные в клетках пути следования. Движение заканчивается на стороне квадрата.

Написать программу, которая сообщит минимальную сумму.

Техническое требование

Заданный квадрат хранится в файле. Файл устроен следующим образом:

```
<Размер квадрата>
<Число> <Число> ... <Число>
<Число> <Число> ... <Число>
...
<Число> <Число> ... <Число>
```

Все числа натуральные. Разделителями являются пробелы. Количество чисел в строке и количество строк совпадает в размером квадрата. Размер квадрата не больше 256.

Программа должна запросить имя файла с описанием квадрата и напечатать длину кратчайшего пути.

Пример

В файле:

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

Ответ: 23

16.150. **Шашки по Чапаеву и Петьке.** Формулировка задачи полностью совпадает с формулировкой задачи 10.35 за одним исключением. Исключение составляет добавленное правило: «(©Петькин) нельзя ходить одной шашкой вперед на столько же, на сколько вторая ходит назад».

16.151. **Постъядерный конь.** На прямоугольной доске некото-

рые клетки заражены. Добраться шахматным конем из начальной клетки в конечную за минимальное число ходов, не вступая на зараженные клетки.

Формат входного файла (строки соответствуют строкам текстового исходного файла):

```
Размеры доски (2 натуральных числа не больше 100).
Координаты начальной точки.
Координаты конечной точки.
Координаты первой из зараженных точек.
    ...
Координаты последней из зараженных точек.
```

Число зараженных клеток не задается.

Выходом программы является число ходов, необходимых для прохода из начальной клетки в конечную, либо -1, если проход невозможен.

Выход программы, запущенной с ключом /g, может задавать участник, как ему вздумается.

Об оценке задачи

Полное число баллов — 60. 30 баллов дается за прохождение стандартной системы тестов в обычном режиме. Время на решение теста не более 10 сек.

Для программ, прошедших все стандартные тесты, жюри пыталось подобрать индивидуальные, если этого не удавалась — начислялась премия в 15 очков.

Участник имел право задать графический либо более красивый символьный вывод решения. Этот вариант вывода должен работать при запуске программы с ключом /g, и проверяется лишь для программ, прошедших все стандартные тесты. За наиболее красиво оформленные решения, в которых графический вывод помогает понять обоснование либо ход решения, начислялось до 15 дополнительных очков. Отсутствие графического вывода не каралось.

16.152. Модуль hex. Острый угол в $(1,1)$. Стороны $(1,1:n)$ $(m,1:n)$ раскрашены в красный цвет, две другие — в синий. Шестиугольники доски также раскрашены в два цвета: синий и красный. Пример расположения ячеек доски 4×5 приведён на рисунке 16.3.

Написать модуль с названием *hex* на разных языках программирования. В него должны входить:

Процедура **getsize(i,j)**. Выдает в качестве значений i, j — размеры доски.

Булевская функция **getdata(i,j)**. Два ее параметра — целые числа. Вызов ее с параметрами $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$ выдает true, если шестиугольник с данными координатами синий, false — если он красный.

Булевская функция **checkend(p)**. Ее параметр — указатель на массив path[1..10000, 1..2] of integer. path должен содержать пары целых чисел, каждая пара в нем должна быть соседом предыдущей. Функция выдает Вам в качестве булева значения, верен ли найденный путь. После её вызова программа должна быть завершена.

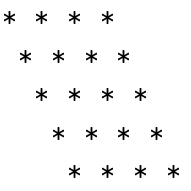


Рис. 16.3: Доска 4x5

Способы задания и хранения информация о доске остаются на Ваше усмотрение.

16.153. **Пёстрый ящик.** Дана гексагональная доска в виде параллелограмма размера $m \times n$. Острый угол в $(1,1)$. Стороны $(1,1:n)$ $(m,1:n)$ раскрашены в красный цвет, две другие — в синий. Шестиугольники доски также раскрашены в два цвета: синий и красный. Известно, что тогда обязательно имеется путь из клеток одного цвета, связывающий две соответствующие противоположные стороны ромба. Ваша задача — найти этот путь.

Техническое требование

Вы пользуетесь модулем *hex* (см. задачу 16.152).

Программа считается лучшей, если она запросила минимальное число данных; при равном количестве запросов — нашедшая более короткий путь.

16.154. **Ящик российских цветов.**

ЧАСТЬ ПЕРВАЯ

Дана доска с шестиугольными клетками в виде параллелограмма размера $m \times n$. Острый угол в $(1,1)$. Стороны $(1,1:n)$ $(m,1:n)$ раскрашены в красный цвет, две другие — в синий.

Известно, что, если шестиугольники доски также раскрашены в два цвета: синий и красный, то обязательно имеется путь из клеток одного цвета, связывающий две соответствующие противоположные стороны ромба. Ваша задача — найти этот путь.

Решение несколько облегчается тем. что некоторые точки доски могут быть не раскрашены (остаются белыми), и Вы сами можете раскрасить их в нужный Вам цвет.

Техническое требование

Вы пишете паскалевский юнит[7] *student*.

Основная Ваша процедура, которая будет вызываться тестирующей программой — процедура без параметров *student_ main*. Каждый её вызов должен полностью проделать решение задачи для данной доски.

Вы пользуетесь юнитом *hex*. В нем имеются:

```
Типы
type hexcolor=(white,red,blue);
     TheXGrid=object(TObject);
...
```

Объект
hexgrid типа со следующими методами:

Процедура getsize(i,j). Выдает в качестве значений i, j-размеры доски.

Функция getdata(i,j) типа hexcolor. Два ее параметра — целые числа. Вызов ее с параметрами $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$ выдает цвет данного. шестиугольника.

Булевская функция checkend(p). Её параметры: массив path[1..10000, 1..2] of integer; целое число pathlength: 1..10000.

Массив *path* должен содержать пары целых чисел, каждая пара в нем должна быть соседом предыдущей. Функция выдает Вам в качестве булева значения, верен ли найденный путь. После ее вызова необходимо дописать в текстовый файл *results.txt* осмысленное сообщение о результатах своей работы и закрыть данный файл. Любые другие действия после её вызова запрещены.

Предупреждение. Во избежание хакерских трюков юнит *hex.tpu*, предоставленный во время тура для отладки Ваших программ, будет отличаться от тех, которые будут действовать во время проверки, так что можно пользоваться лишь спецификацией данных программ.

Для облегчения отладки getdata, предоставленная во время тура, будет считывать данные из символьного массива вида (массив — примерный)

board.txt(1 — синяя клетка, 0 — красная, -1 — белая):

```
4
1 0 1 1
1 1 0 -1
0 0 1 1
1 -1 0 0
```

[7] В языке Turbo Pascal Unit подключается к основной программе в разделе Uses.

Вы можете сами модифицировать данный массив для проверки Вашей программы. Откуда будут брать данные getdata в тестах жюри — не Ваше дело.

Среди участников, прошедших некоторый тест, максимальное число очков будет даваться запросившему минимальное число данных, при равном количестве запросов — нашедшему более короткий путь. Если некто запросил цвета всех клеток доски, он получает не более 1 очка за данный тест. Если несколько тестов решены подобным образом, то вторая часть решения участника не рассматривается.

ЧАСТЬ ВТОРАЯ

Написать юнит *hex.tpu* самим и таким образом, чтобы программа getdata по возможности удлиняла Вашим друзьям-соперникам поиск пути.

Юнитом *student* пользоваться запрещается. Вы пользуетесь юнитом *arbitre*.

При вызове программы getsize Вы должны дописать в файл *results.txt* Ваш регистрационный номер в форме:
`Nasty things from #####`.

Программа checkend должна: выдать в *results.txt* количество запрошенных Вашим оппонентом данных в форме `#### queries made!`; закрыть его и вызвать процедуру

```
confirmation(queries: integer; # количество сделанных запросов.
  checked: array[1..10000,1..2]; # запрошенные точки
  responses: array[1..10000] of hexcolor; #Ваши ответы
  correct: Boolean # решена ли задача
);
```

Если некто сделал больше запросов, чем клеток в доске, можно вызвать confirmation и раньше, с соблюдением всех вышеуказанных условий. Последнюю запрошенную точку в этом случае можно не выдавать.

Ваша getsize должна вызвать процедуру `official_size(m,n)` и выдать предписанные ею размеры доски. Так что размеры доски определяете не Вы, а жюри.

Ваша getdata должна вызывать функцию `official_data(i,j) : hexcolor` и имеет право изменять белый цвет на нужный Вам, а красный и синий getdata должна, как бы Вам ни было обидно, передавать сопернику в точности. Впрочем, функцию `official_data` Вы можете вызывать и в другом месте, проанализировав доску заранее.

Оценка второй части будет производиться лишь для участников, наиболее успешно справившихся с первой частью. Чем больше запросов понадобится Вашим соперникам для поиска путей, тем выше будет оценка

Вашей программы. Она в любом случае 0, если Ваша программа некорректно отвечает либо некорректно оценивает решения оппонентов.

ЧАСТЬ ТРЕТЬЯ

Цель та же, что и во второй части, но вместо процедуры getdata Вы пишите процедуру `Changecolor(i,j,i1,j1,Newcolor)` i, j — входные, i1, j1, Newcolor — выходные, которая по запросу соперником `(i,j)` окрашивает в нужный Вам цвет одно из белых полей доски `(i1,j1)`. Если $(i1, j1) = (0, 0)$, то это означает, что Вы отказываетесь закрашивать поле. Так что прямо мешать противнику Вы не сможете, и остаётся делать гадости на будущее.

`Changecolor` находится в том же юните *hex* и является методом объекта `HexGrid`.

Процедура `Confirmation` остается без изменения.

16.155. **Шестиугольный Брезенхем.** В последнее время всё чаще в экранах стали использовать шестиугольный растр вместо прямоугольного. Этого можно добиться если смесить каждую чётную строку на половину пиксела влево, а строки немного приблизить друг к другу. Этот подход даёт большее разрешение при том же размере экрана. На рисунке 16.4 приведён пример.

Естественно, классические алгоритмы рисования для такого растра уже не годятся. Отсюда задача: придумать и реализовать алгоритм рисования отрезка прямой на шестиугольном растре.

```
 * * * *
* * * *
 * * * *
```

Рис. 16.4: Растр 4x3

Техническое требование

Считаем размер экран постоянным: 1000×1000. Система координат обычная прямоугольная. Координаты считаются так, как если бы растр оставили прямоугольным. Экран отображает первую четверть. Координаты считаются с 0. Строка — чётная, если ордината нечётная (строка с нулевой ординатой — нечётная).

Программа должна считывать координаты концов отрезков из файла *lines.txt* и строить эти отрезки. Файл *lines.txt* начинается с числа отрезков, затем написаны четвёрки чисел через пробелы (один или несколько) — координаты концов отрезка. Первая пара — начало, X и Y соответственно, вторая — конец. $0 \leqslant X, Y < 1000$.

Рисование происходит в файл с именем *screen.hex*. В файл записываются пары координат точек линии, по одной точке в строке. Координаты разделены пробелами. Первая точка должна совпадать с началом линии, последняя — с концом. Линии проводятся независимо.

Дополнение

Структура программы должна состоять из, минимум, двух модулей. Один из них — include-файл с именем *pset.inc* должен содержать **только** подпрограмму pset(X,Y) — Вашу процедуру, которая «ставит» точку на экран, а точнее записывает её в файл *screen.hex*. При тестировании она модифицируется жюри. Для записи в файл *screen.hex* другими средствами пользоваться запрещается. Остальные модули — на усмотрение автора.

Оценка программы

Отрезок считается построенным неправильно:

- если предыдущий пиксел не является соседом следующего;
- если отклонение центра пиксела от действительного отрезка больше половины диаметра пиксела по прямой.

Оценивается скорость работы алгоритма и количество рисуемых пикселов.

Пример

lines.txt

```
3
0 0 0 3
3 0 0 0
0 3 3 0
```

screen.hex

```
0 0
0 1
0 2
0 3
3 0
2 0
1 0
0 0
0 3
0 2
1 2
2 2
2 1
3 1
3 0
```

16.156. **N N N.** Даны три набора действительных чисел. Количество чисел во всех наборах одинаково и равно N. *Набор сумм* образуется из всевозможных сумм чисел, где каждая сумма получается сложением трёх чисел, ровно по одному из каждого набора.

Задача

Все числа из *набора сумм* разбить на N групп при условии, что количество чисел в каждой группе будет одинаково, каждое число из *набора сумм* используется ровно один раз и суммы всех чисел группы одинаковы для всех групп.

Техническое требование

N<100. Три начальных набора содержатся в файле *input.txt*.

Формат файла *input.txt*

```
N
<1-е число первого набора>
<2-е число первого набора>
...
<N-е число первого набора>
<1-е число второго набора>
<2-е число второго набора>
...
<N-е число второго набора>
<1-е число третьего набора>
<2-е число третьего набора>
...
<N-е число третьего набора>
```

Формат файла *output.txt*

```
<номер числа набора 1 группы 1> <ном.ч.наб.2 гр.1> <ном.ч.наб.3 гр.1>
<номер числа набора 1 группы 1> <ном.ч.наб.2 гр.1> <ном.ч.наб.3 гр.1>
    ...
<номер числа набора 1 группы 2> <ном.ч.наб.2 гр.2> <ном.ч.наб.3 гр.2>
    ...
<номер числа набора 1 группы N> <ном.ч.наб.2 гр.N> <ном.ч.наб.3 гр.N>
```

Программа должна читать данные из файла *input.txt* и записывать результат в *output.txt*.

Пример

input.txt:

```
2
0.1
0.2
1
```

```
2
10
20
```

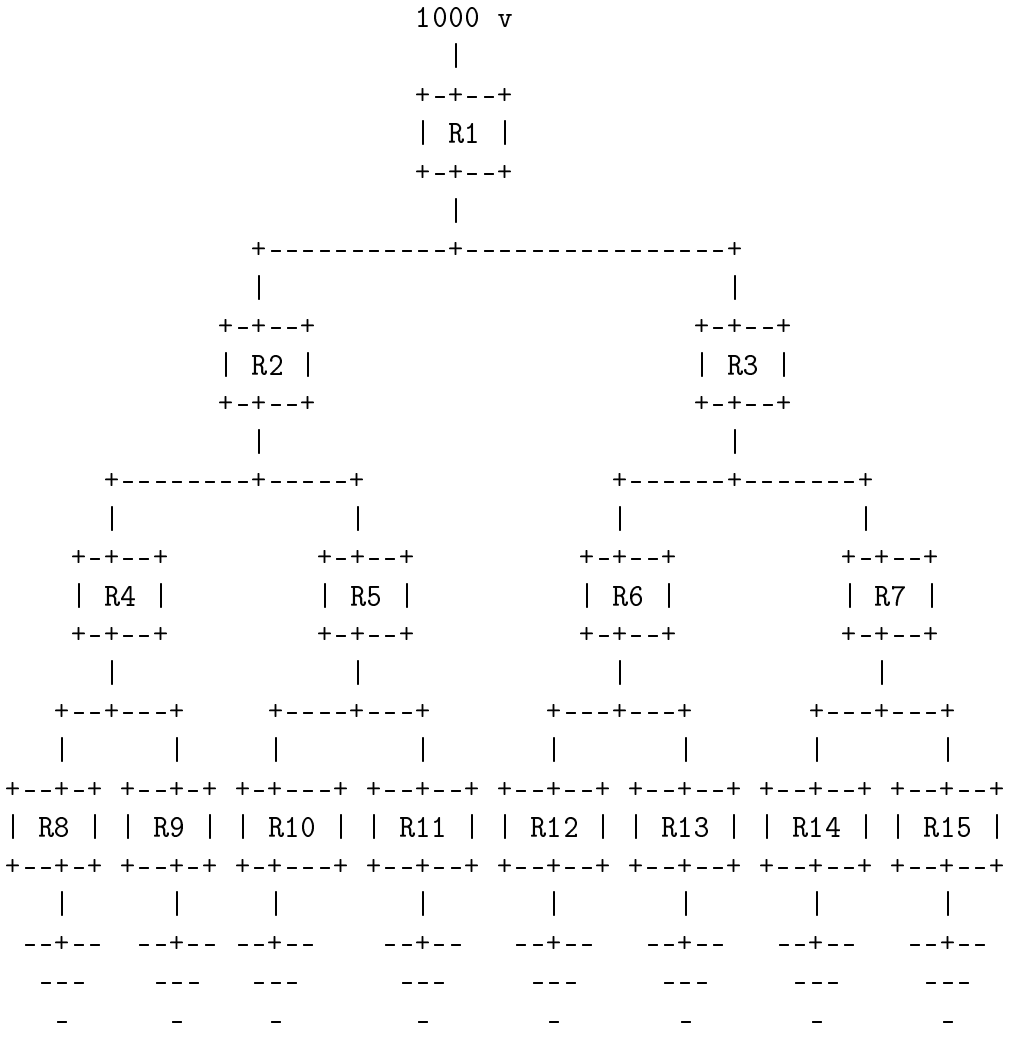
output.txt:

```
1 1 1
2 2 2
1 1 2
2 2 1
1 2 1
2 1 2
1 2 2
2 1 1
```

16.157. **Диагностика.** Дана электрическая схема, составленная из резисторов с номинальным сопротивлением в 1 ом, соединенных в виде полного бинарного дерева глубины N, листья которого соединены с землей, а на корень подано напряжение +1000 вольт.

Например, для дерева глубины $N = 4$ схема выглядит следующим

образом:



Все резисторы пронумерованы сверху вниз слева направо: первый резистор от корня — R1, следующий ряд — R2, R3 и т. д.

Известно, что схема может быть неисправна — некоторые резисторы, не соединенные непосредственно с землей, могут быть закорочены (фактически имеют нулевое сопротивление). Других неисправностей нет. В схеме примера повреждены могут быть резисторы от R1 до R7.

Имеется вольтметр, которым можно измерять напряжение в любой точке схемы относительно земли. Точки обозначаются номерами резисторов, начинающими путь от этой точки до корня. В данном примере это номера от 1 до 7.

Требуется создать программу диагностики данной схемы.

Программа должна получать данные через стандартный входной поток (например, для языка C/C++ это stdin) и выдавать данные в стандартный выходной поток (для языка C/C++ — stdout). Первое входное данное — глубина проверяемой схемы, число N. Далее программа выдает номер проверяемой точки в схеме — целое число в десятичной системе счисления, за которым следует символ перевода строки. Затем программа читает результат измерения напряжения в данной точке - целое число в десятичной системе счисления, выражающее значение напряжения в микровольтах. Значение напряжения округляется до ближайшего целого числа. Результат тестирования выдается в виде списка обозначений неисправных резисторов в порядке возрастания их номеров, разделенных символами перевода строки.

Между диагностическими программами проводится соревнование на специальном наборе из 10 схем. На каждой схеме выявляются победители: побеждают программы, правильно нашедшие все неисправности за наименьшее число измерений. Они получают по 10 баллов. Программы, выполнившие наибольшее число измерений, получают по 1 баллу. Оценки остальных программ распределяются равномерно в зависимости от числа проведенных испытаний между этими крайними значениями с округлением до ближайшего целого числа баллов.

Требуется, чтобы программа работала в режиме реального времени: время реакции не должно превышать 1 секунды для процессора Pentium с тактовой частотой 100 мегагерц. «Зависающие» программы снимаются с соревнований. Программам, не нашедшим всех неисправностей схемы или «обнаружившим» не существующие неисправности, засчитывается поражение на данной схеме.

Пример

Вход	Выход
3	1
1000000000	2
1000000000	3
1000000000	R1
	R2
	R3

16.158. **Действия в остаточных классах.** Число k представляется в системе остаточных классов следующим образом. Задается n оснований: различных простых чисел p_i, и k представляется как система остатков k_i. Например, 12 в системе остаточных классов 3, 5, 7 представляется как 0, 2, 5.

Часть 1

Написать программу для четырех арифметических действий в остаточных классах.

Часть 2

Остаточные классы используются для машинного представления чисел в спецпроцессорах. При этом числа, большие $max/2$, где max — произведение всех оснований, рассматриваются как отрицательные. Соответственно, появляется понятие переполнения. Написать программу, учитывающую переполнения, и реализующую дополнительно к четырем арифметическим действиям ввод чисел в десятичной системе, вывод их (в той же системе), сравнение двух чисел.

Техническое требование

Основания — 32-разрядные целые числа без знака.

Программа должна начинаться с однозначно читаемого комментария, в котором указывается, какой вариант задачи решен — часть 1 или часть 2.

Если входные данные некорректны, программа должна нормально завершить работу и выдать осмысленное сообщение об ошибке.

Входной файл *inp1.txt* для первой части имеет вид:

число оснований n ($1 \leqslant n \leqslant 100$),

n целых чисел — оснований представления.

Строчка ***

$2 * n$ целых чисел, являющихся представлениями первого и второго операндов.

Выходной файл *res1.txt* для первой части содержит $4*n$ целых чисел, расположенных по одному в строке, и являющихся представлениями в перечисленном порядке

Суммы

Разности

Произведения

Частного

операндов. Если происходит деление на 0, вместо последних n результатов пишется одна строчка `overflow`. Результаты разделяются строчками ***

Входной файл *inp2.txt* для второй части имеет вид

число оснований n ($1 \leqslant n \leqslant 100$),

n целых чисел — оснований представления.

Строчка ***

Строка цифр, задающая в десятичной системе первый операнд. Первым символом в строке может быть знак '−'.

n чисел, задающих в системе остаточных классов второй операнд.

Выходной файл *res2.txt* для второй части содержит шесть строк.

Первые пять задают в десятичной системе
Сумму
Разность
Произведение
Частное
Остаток от деления
операндов. Если происходит переполнение, то вместо соответствующего результата пишется строчка `overflow`. Остаток от деления имеет тот же знак, что и делитель.

В последней стоит один из символов $>$, $=$, $<$, в зависимости от результата сравнения операндов.

Пример

Входной файл для части 1

```
3
3 5 7
***
2 4 6
2 3 5
```

Выходной файл для части 1

```
1
2
4
***
0
4
6
***
1
3
5
***
2
2
2
```

Входной файл для части 2

```
3
3 5 7
***
45
1 3 3
```

Результат для части 2

```
7
overflow
overflow
-1
7
>
```

Пример ошибки.

```
4
5 7 8 11
...
```

Сообщение в выходном файле (примерное):

```
Основание 8 не простое!
```

Примечание. Результаты в примерах могут быть неправильными, во избежание жульничества.

Программа должна давать ответ за 5 сек.

16.159. **Палиндром перестановками.** За какое минимальное число перестановок символов можно превратить заданную строку в палиндром. Перестановка — обмен местами двух разных символов.

Техническое требование

Программа должна читать символы из текстового файла *string.txt* и печатать минимальное число перестановок. Во входном файле все символы записана подряд. Символов не больше 10000. Если строку в палиндром превратить невозможно, то напечатать -1. Время будет ограничено.

Пример

Во входном файле: `ппоот`

```
Перестановок: 2
```

16.160. **Шарик.** Дана комната в форме прямоугольного параллелепипеда. Комната разделена тонкими плоскими вертикальными перегородками, параллельными одной стене. Каждая перегородка имеет форму прямоугольника, длина ее совпадает с длиной комнаты. Перегородка стоит на полу и имеет свою индивидуальную высоту. Перегородки расставлены на равном расстоянии друг от друга и от параллельных стен. Со стороны одной из стен, параллельной перегородкам, из-под потолка в комнату вбрасывается маленький абсолютно упругий шарик. Шарик вбрасывается вдоль потолка (без взаимодействия с ним) перпендикулярно перегородкам. Шарик вбрасывается так, что если бы не было

перегородок, то он под действием силы тяжести упал бы точно в угол, образованный противоположной стеной и полом. Считается, что стены комнаты строго вертикальны. Для каждой перегородки задается *время* достижения шариком высоты ее верхней кромки при падении от потолка. Шарик абсолютно упруго отражается от пола, потолка, стенок и перегородок. Размерами шарика и сопротивлением воздуха следует пренебречь. При пролете точно через верхнюю кромку перегородки шарик с ней не взаимодействует. При попадании точно в угол шарик отражается в противоположном направлении. Нулевая высота перегородки эквивалентна ее отсутствию. «Верхняя кромка» такой перегородки достигается шариком одновременно с полом. От пола шарик отражается точно заданное число раз N. На N+1-й шарик прилипает к полу.

Требуется написать программу, которая по числу N и временам достижения шариком высот верхних кромок перегородок вычисляет номера перегородок, между которыми шарик прилипнет.

Входные данные

Входной файл *input.txt* содержит натуральные числа в следующем порядке:

1. Число перегородок (целое от 0 до 100).
2. Время достижения шариком пола (в миллисекундах, целое число от 1 до 10000).
3. Времена достижения шариком высот верхних кромок перегородок в порядке следования перегородок от стенки вбрасывания шарика (в миллисекундах, целые неотрицательные числа, не превышающие время достижения пола).
4. Число отражений шарика от пола (N, целое число от 0 до 1000).

Выходные данные

Выходной файл *output.txt* содержит номер перегородки, перед которой шарик прилипнет в порядке следования перегородок от стенки вбрасывания шарика. Если шарик прилип за последней перегородкой, то результат — номер последней перегородки, увеличенный на 1. Если шарик попадает точно в место расположения перегородки, то следует выдать ее номер. Если шарик прилип в углу пола и стены вбрасывания, то следует выдать 0.

Время на вычисление — не более 5 сек.

Пример

Во входном файле:

```
3
10000
1000
2000
```

```
3000
10
```
В выходном файле:
```
0
```

16.161. **Бутылка и бублик.** Гомеоморфизм двух фигур означает возможность непрерывно и взаимно-однозначно преобразовать каждую из них в другую (таком образом, здесь не может быть разрывов и отождествлений). Например, треугольник гомеоморфен квадрату и кругу, но не кольцу либо сфере.

Двумерная поверхность (многообразие) задана следующим образом. Она разбита на куски, гомеоморфные замкнутым треугольникам (симплексы). У каждого симплекса некоторые из сторон могут быть склеены со стороной другого либо того же симплекса. Каждая из сторон может участвовать лишь в одной склейке.

Склейка может производиться таким образом, что склеиваемые стороны ориентированы в одну и ту же сторону либо противоположно. Направления сторон в каждом симплексе: первая от третьей ко второй, вторая — от первой к третьей, третья — от второй к первой. Если для склейки требуется отождествить другие вершины, за исключением тех, которые отождествляются непосредственно при склеивании сторон, задание многообразия считается ошибочным.

Число симплексов в многообразии не более 100.

Многообразие задается записью следующего вида.

```
<Число симплексов> <Число отождествлений>
<Номер симплекса> <Номер стороны> <Номер симплекса>
    <Номер стороны> <Ориентация склейки>
```

Пример

```
2 3
1 1 2 2 +
1 2 2 3 +
1 3 2 1 +
```

Это означает, что в многообразии два симплекса, первая сторона первого склеена со второй стороной второго в том же направлении, вторая сторона первого — с третьей стороной второго, а третья — с первой стороной второго. Легко видеть, что данное многообразие гомеоморфно сфере.

Ориентация задается символами '+' или '−'.

Входной файл *input.txt* содержит последовательно заданные два многообразия. Нужно вывести на экран три текстовых строчки. В первой

говорится, гомеоморфны ли они, во второй — какому из стандартных многообразий эквивалентно первое многообразие, в третьей — какому эквивалентно второе. Если не удается установить гомеоморфность чему-то стандартному, то вторая (третья) строчка должна содержать символы '???'. Если заданное многообразие просто невозможно, то это нужно записать в качестве текстовой строчки.

Программа должна распознавать следующие стандартные многообразия.

Круг
Сфера
Кольцо
Кольцо Мебиуса
Тор
Бутылка Клейна
Проективная плоскость

Не запрещается в некоторых случаях распознавать и другие многообразия, вместо того, чтобы писать «???».

Пример

Вывод не обязательно должен быть в точности такой и не обязательно правильный.

```
2 3
1 1 2 2 +
1 2 2 3 +
1 3 2 1 +
1 0
```

Вывод

```
Негомеоморфны
Сфера
Круг
```

```
2 3
1 1 2 2  -
1 2 2 3 +
1 3 2 1 +
2 1
1 1 2 3  -
```

```
Негомеоморфны
Невозможно
Круг
```

Примечание. Невозможность означает некорректность определения данного многообразия вообще, а не невозможность вложить его без само-

пересечений в наше трехмерное пространство (например, бутылку Клейна в него не вложишь без самопересечений). Также и возможность преобразовать многообразия друг в друга не означает существование непрерывной деформации нашего пространства, их преобразующей. Узлов мы не рассматриваем.

Для желающих сообщаем, что для вложения без самопересечений любых двумерных многообразий достаточно четырехмерного пространства.

16.162. **Задача о знакомых.** В телеконференции принимают участие M людей, причем некоторые из них знакомы друг с другом. Сколько, минимум, должно быть участников, чтобы хотя бы N из них либо были попарно знакомы друг с другом, либо попарно незнакомы.

Техническое требование

Написать программу, которая по заданному в файл стандартного входа числу N выводит в файл стандартного выхода число M.

Ограничения

$N \leqslant 163$.

Пример

```
N= 2
M= 2
```

16.163. **Гонки на бутылке.**

К этому моменту должна быть решена задача 16.161. Вы можете использовать идеи либо фрагменты из нее для решения следующей задачи.

В файле *bottle.txt* находится определение многообразия в том же виде, что и на первом туре: через склейки сторон треугольников, например

```
4 4
1 2 2 1 -
1 3 4 1 -
2 3 3 3 -
3 2 4 2 -
```

В первой строке — количество треугольников и количество склеек.

В файле *init.txt* находится начальная позиция Вашего игрока (охотника) и жертвы. Позиция задается следующей строкой:

$$3 + 15 - 21$$

Первое число — номер треугольника, в котором находитесь Вы, далее идет его ориентация (если '+', то нумерация его сторон идет для Вас против часовой стрелки, т. е. в положительном направлении). Далее —

те же данные для жертвы. Предпоследнее число — Ваша скорость, последнее – скорость жертвы.

Внимание! Скорость жертвы может быть 0.

Ваша задача — за минимальное число шагов нагнать жертву. Жертва поймана, если Вы оказались на одной и той же поверхности одного и того же треугольника. Если Вы выяснили, что многообразие несвязно или Вы на разных сторонах двусторонней поверхности (и только в этом случае) Вы первым ходом выдаете в файл *move.txt* строку `"Surrender"`. Если же скорость Ваша недостаточна для поимки, Вы все равно должны пытаться, надеясь на тупость жертвы. На поимку отводится максимум 100 ходов.

Ваше движение состоит в переходах из треугольника в треугольник *через общую вершину*. Движение жертвы — *через общую сторону*. Вы можете не использовать все разрешенные Вам ходы и даже стоять на месте, жертва использует все ходы обязательно.

Ваши данные Вы можете хранить в файле *hunter.txt*. Формат данного файла и его содержимое остается на Ваше усмотрение. Перед Вашей работой жюри не обязано очистить данный файл от «мусора», оставленного Вами либо другими.

Ваша программа компилируется как файл *hunter.exe* и будет вызываться для очередного хода другой программой. При ее втором и последующем вызовах в файле *move.txt* находится номер и ориентация треугольника, в котором находится жертва. На поимку вызывающая программа среагирует сама.

В файл *move.txt* Вы выдаете после очередного вызова последовательность строк вида (например):

```
2
2 -
4 +
```

где первое число — число Ваших ходов (не больше Вашей скорости, может быть нулем), следующие строки — последовательные треугольники, проходимые Вами, если в первой строке не ноль.

Внимание! Задание многообразия удовлетворяет следующим ограничениям.

Никакие три стороны не склеиваются друг с другом.

Не склеиваются две стороны одного и того же треугольника.

Никакие два треугольника не склеиваются больше, чем одной стороной.

Дополнительные условия

Для получения премиальных очков Вы имеете право представить программу *eluder.exe*, которая играет за жертву.

Файл, в котором Вы храните вспомогательную информацию, *eluder.txt*.

Забота о контроле сохранности файлов *hunter.txt* или *eluder.txt* лежит на Вас самих. Если Вы поймали противника на изменении Вашего файла, Вам засчитывается максимальное число очков за данный тест. В этом случае Вы должны выдать в файл *move.txt* осмысленное сообщение о жульничестве, желательно с информацией, которая поможет жюри доказать нечестную игру противника. Программа жюри нечестностей проявлять не будет, но будет контролировать корректность Ваших ходов (и Вашего противника, если он бьется за дополнительные очки). За ложное обвинение Вы получаете штраф.

Использование Вами фрагментов из Вашей программы первого тура никак не регламентируется и близость Вашей программы к программе первого тура никак не оценивается.

16.164. **Задача о рисовании графа.** Дано: неориентированный граф задан своими дугами, представленными в виде таблицы (номеров вершин первого и второго концов).

Построить программу, которая считывает текстовый файл (*graph.txt*), каждая строка которого представляет дугу и содержит два номера вершин, которые эта дуга соединяет.

По этим входным данным программа должна построить изображение графа на плоскости. Программа должна выдать текстовый файл (*graph.xml*), содержащий XHTML 1.0 (http://www.w3c.org), который с точностью до цвета и шрифта одинаково просматривается и текстовым (lynx, links) и графическим (Netscape, IE) броузером. Все вершины графа должны быть гипертекстовыми ссылками (HREF="<номер вершины>.xhtml").

Техническое требование Граф содержит не более 1000 дуг. Можно рассчитывать на Pentium 200, 1 Мб оперативной памяти и 2 Мб на диске. Но возможно будет выделено и больше ресурсов. Программа должна работать не дольше 10 секунд.

Пример входных данных

```
7 1
```

Пример изображения выходных данных в браузере Lynx:

7-1

16.165. **Освещение лабиринта.** В плоском лабиринте размером

до 100×100 нужно расставить светильники так, чтобы не осталось неосвещённых мест. Как известно, свет распространяется по прямой. В нашем лабиринте стены такие, что они поглощают весь свет, попавший на них, то есть рассчитывать на отражение и просвечивание не приходится. Мощности светильников хватает на то, чтобы осветить коридор от одного края лабиринта до другого. Светильники в стены ставить нельзя.

Светильник представляет собой точечный источник света и устанавливается в центре клетки. Клетка считается освещённой, если освещена её часть не нулевой площади.

Техническое требование

На вход программе будет подаваться файл *maze.txt* с лабиринтом. В результате работы программа должна создать файл *lamp.txt*, в котором содержится исходный лабиринт с расставленными в нём светильниками.

Формат файлов вполне понятен из примера (см. ниже).

Пример

```
maze.txt          lamp.txt
6                 6
7                 7
#######           #######
# # ###           # # ###
# #  ##           # # *##
# ## ##           # ## ##
#    ##           #*   ##
#######           #######
```

ОЦЕНКА РЕШЕНИЯ

Правильные решения сравнивают по минимуму затраченных светильников.

16.166. **Головоломка «Бесконечность».** Головоломка состоит из двух колец, пересекающихся в одной точке. В каждом кольце — три шарика. Один из шариков — в точке пересечения. Так что всего — пять шариков. Шарики пронумерованы цифрами от 1 до 5. Стандартное положение шариков, обозначаемое кодом 12345 (пятизначное число), изображено на рисунке:

```
 1--\ /--2
<    3    >
 4--/ \--5
```

Если в указанных позициях находятся другие шарики, то положение описывается соответствующей последовательностью, например, код 35124 описывает конфигурацию:

```
 3--\ /--5
<    1    >
 2--/ \--4
```

Кольца можно вращать: левое — налево (против часовой стрелки), правое — направо (по часовой стрелке). Эти повороты обозначаются буквами L и R соответственно. Таким образом, поворот L преобразует конфигурацию 12345 в 32415, а поворот R преобразует 12345 в 13542.

Требуется написать программу, выдающую по коду конфигурации последовательность поворотов, которая переводит эту конфигурацию в стандартное положение (12345).

Формат входных данных (в файле input.txt):

```
<число исходных конфигураций N>
<код конфигурации 1> ... <код конфигурации N>
```

Разделители — пробелы и переводы строк. Внутри кодов пробелов нет.

формат выходных данных (в файле output.txt):

```
<последовательность поворотов 1>
...
<последовательность поворотов N>
```

Внутри последовательностей пробелы отсутствуют. Если решения нет выдается буква N. Разделители — переводы строк.

Время решения — 10 секунд.

Оценка

При прохождении одинаковых тестов лучшей считается программа, выдающая в среднем наиболее короткие правильные последовательности поворотов.

Пример

input.txt

```
3
32415
12345
13542
```

output.txt

```
LL

RR
```

Обратите внимание — вторая последовательность пустая!

16.167. **GIF-сообщение.** В графическом файле (формат — анимационный gif (GIF89)) записано сообщение, по одному символу на каждой картинке файла, причём весь символ будет одного цвета, а фон другого. Но от картинки к картинке эти цвета могут меняться.

Техническое требование

Написать программу, которая читает файл *message.gif* в формате GIF и записывает в файл *message.txt* сообщение, закодированное *message.gif* Символами, которые будут использоваться для записи сообщения, будут: Латинские буквы (большие и маленькие), цифры, знаки препинания и пробел. Знаки препинания: '.', ',', '!', '?', '-'. Шрифты для написания букв не ограничиваются. Так же не гарантируется сохранение шрифта и размера на протяжении всего сообщения. Единственное ограничение — человек в состоянии однозначно идентифицировать каждый символ сообщения даже по отдельности. Длина сообщения не более 80 символов.

Пример

в message.gif (все картинки показаны вместе):

в message.txt: `It is a test!`

16.168. **Кратчайший делитель файла.** Файл — конечная последовательность (цепочка) символов. Конкатенация файлов — сцепление этих последовательностей в одну. Умножение файла на число n — n-кратная конкатенация файла с самим собой. Кратное файла — результат его умножения на натуральное число. Делитель файла — файл, по отношению к которому исходный файл является кратным. Кратчайший делитель — наиболее короткий делитель.

Задача — найти кратчайший делитель исходного файла, если он есть.

Входные данные

Файл *text.txt* — исходный файл.

Выходные данные

Файл *div.txt* — выходной файл. Если делителей нет, сообщение об ошибке «делителей нет» выдается на монитор.

Ограничения

Входной файл может содержать только цифры и латинские буквы (строчные и прописные). Время работы — не более 10 секунд. Объем обрабатываемого (за это время) файла — предмет конкурса.

Пример

Если *text.txt* — это цепочка

aabaabbaabaabbaabaabb

то *div.txt* — это
`aabaabb`

16.169. **Тип файла.** Со времён операционной системы MS-DOS стало традицией заканчивать имя файла несколькими символами, характеризующими внутреннюю структуру файла. Обычно этих символов от 1 до 3 и идут они после точки. Называют такое окончание расширением.

Задача

Написать программу, которая по заданному файлу укажет традиционное для его структуры расширение.

Техническое требование

Имена файлов будут записаны в файле *list.txt* по одному в каждой строке. Имена файлов будут состоять только из строчных латинских букв, цифр и одной точки.

Каждое имя файла устроено следующим образом: вначале идут буквы или цифры количеством от 1 до 8. Затем может быть одна точка. Потом, но только если есть точка, от 1 до 3 символов.

Сами файлы расположены в том же каталоге, что и программа, обрабатывающая список.

Размеры файлов до 1 Мб.

Результат нужно записать в файл *list.ext*. Расширения должны быть записаны по одному в каждой строке в том же порядке, что и имена характеризуемых файлов в *list.txt*. Если типичное расширение указать невозможно, то в ответ нужно записать строку из трёх знаков вопроса.

Время работы программы составляет 1 секунду на каждый файл в списке. Объём используемой памяти (оперативной, виртуальной или дисковой) не ограничивается.

Пример

В list.txt
```
task.txt
task.exe
task.000
task.001
```
В list.ext
```
txt
com
???
xpm
```

Оценка решений

Лучшей считается программа, определяющая большее количество структур файлов.

16.170. **Удаление от файлов.** Файл — последовательность байтов. Байт — последовательность из 8 бит, т. е. двоичных цифр, каждая из которых может быть равна 0 или 1. Позиция в файле — номер байта и номер бита в этом байте. Расстояние между двумя файлами — число позиций, в которых у этих файлов стоят разные двоичные цифры.

Пусть дано некоторое множество *исходных* файлов одной длины. Требуется создать новый файл той же длины, наиболее удаленный от этих ИСХОДНЫХ файлов.

Входные данные

В файле с именем *names* перечислены имена *исходных* файлов, разделенные пробелом (каждое имя — последовательность от 1 до 8 цифр). Длины всех файлов с указанными именами одинаковы.

Выходные данные

Требуется построить файл той же длины, что и *исходные* файлы, с именем *new*, так, чтобы минимум расстояний от построенного файла до ИСХОДНЫХ файлов был максимален.

Пример

Входные файлы (в первой строке — имя файла, далее — содержимое файла до пустой строки; каждый байт каждого *исходного* файла изображен отдельной строкой из 8 двоичных цифр):

names

```
1 2
```

1

```
00000000
11111111
```

2

```
00000000
11110000
```

Выходной файл с именем *new*:

```
11111111
00001100
```

Оценка программ

Соревнование между программами проводилось по мере усложнения тестов (увеличения числа и длины файлов, число файлов — важнее). К следующим этапам допускались лучшие программы по предыдущим этапам. На каждом этапе проводился один тест. Соревнование проходило до

выявления наилучшей программы (но не более, чем в 10 этапов). Число баллов за каждый тест назначалось обратно пропорционально числу прошедших его программ (с округлением до ближайшего целого числа, но не меньше 1). Общее число баллов — 100. Все баллы, которые не удастся распределить в соответствии с указанным правилом, назначаются за прохождение последнего теста.

Ограничения

Время работы программы — не более 10 секунд (для тактовой частоты процессора P II 600 МГц).
Объем памяти — не более 10 Мб.
Число файлов и их длина — предмет конкурса.

16.171. **Электронный микроскоп.** При изучении кристалла под электронным микроскопом получены его три плоские проекции. Требуется восстановить объемную структуру кристалла.

Техническое требование

Предполагается, что атомы кристалла находятся в узлах прямоугольной трехмерной решетки с целочисленными координатами, не более чем по одному в каждом узле. Проекции проводятся вдоль осей координат. Электронный микроскоп в состоянии определить число атомов, попавшее на пути прохождения луча.

Входные данные

Данные задаются в файле *input.txt*. Предполагается, что кристалл размещается в кубе с ребром N (натуральное число). Структура исходных данных:

```
N
<проекция вдоль оси X>
<проекция вдоль оси Y>
<проекция вдоль оси Z>
```

Каждая проекция имеет вид:

$$\begin{matrix} A(1,1) & \cdots & A(1,N) \\ \vdots & \ddots & \vdots \\ A(N,1) & \cdots & A(N,N) \end{matrix}$$

где $A(i,j)$ — число атомов встретившееся на пути луча с координатами i,j (для проекции вдоль оси X $i=Y$, $j=Z$, для проекции вдоль оси Y $i=Z$, $j=X$, для проекции вдоль оси Z $i=X$, $j=Y$). Числа $A(i,j)$ — целые, в десятичной записи.

Все числа во входном файле разделяются пробелами и/или переводами строк. В конце файла также может размещаться пробел и/или перевод строки. Переводы строк могут быть двойными.

Выходные данные

Выходные данные следует записывать в файл *output.txt*. Если решения нет, то выходной файл делается пустым. Иначе в нем помещаются натуральные числа:

$$\begin{array}{ccc} B(1,1,1) & \cdots & B(1,1,N) \\ \vdots & \ddots & \vdots \\ B(1,N,1) & \cdots & B(1,1,N) \\ \vdots & \vdots & \vdots \\ B(N,1,1) & \cdots & B(N,1,N) \\ \vdots & \ddots & \vdots \\ B(N,N,1) & \cdots & B(N,N,N) \end{array}$$

где $B(X,Y,Z) = 1$, если в узле с координатами (X,Y,Z) есть атом, и $B(X,Y,Z) = 0$, если в узле с этими координатами нет атома.

После каждого числа ставится пробел, после каждых N чисел после этого пробела ставится перевод строки, после каждых N строк с числами и в конце файла ставится дополнительный перевод строки (пустая строка).

Если возможно несколько решений, то надо выдать одно из них.

Условия конкурса

Программы-решения тестируются при увеличении N. Программы, не прошедшие тесты для меньших N, не допускаются к тестированию для больших N. Число баллов за каждый тест (кроме последнего) пропорционально (с точностью до целого округления) числу не прошедших его (и не дошедших до него) программ. Программы, прошедшие все тесты, получают полное количество баллов, предусмотренное за одну задачу.

Пример

Пример файла *input.txt*

```
3

3 3 3
3 2 3
3 3 3

3 3 3
3 2 3
```

```
3 3 3

3 3 3
3 2 3
3 3 3
```

Пример файла *output.txt*

```
1 1 1
1 1 1
1 1 1

1 1 1
1 0 1
1 1 1

1 1 1
1 1 1
1 1 1
```

16.172. ❐ **Задача риэлтера II.** Перед решением этой задачи должна быть решена задача 13.60.

Оказывается, что выходные данные в виде номеров обменов слишком неудобны для риэлиера. Поэтому, пожалуйста, скорректируйте выходные данные. Теперь в файле *output.txt* нужно записать информацию о том, какая квартира меняется и на какую. Квартиры идентифицируются по двум параметрам: номер обмена и номер квартиры в обмене.

Формат *output.txt*

`<номер обмена> <номер продаваемой квартиры> <номер обмена> <номер желаемой квартиры>`

Если квартира не для обмена, а для продажи (в желаемой части только деньги), то вторая пара чисел — нули.

Изменение в примере из задачи 13.60

В файле output.txt

```
10 3 20 1
20 1 12 1
10 2 12 2
20 2 12 3
 1 1  0 0
```

16.173. ❐ **Туманный лабиринт.** Задан лабиринт 10×10, в каждой

клетке которого может находиться или отсутствовать стена. Клетки нумеруются слева направо и сверху вниз числами от 0 до 99 (рис 16.5). Вокруг лабиринта — сплошная стена. Программа за один вопрос может узнать, что находится в клетке с заданным номером. Из одной клетки можно переходить в другую, если они имеют общую сторону и ни в одной из них нет стены.

```
 0  1  2  3  4  5  6  7  8  9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 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 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99
```

Рис. 16.5: Нумерация клеток

Требуется, задав не более 100 вопросов узнать, как пройти из клетки 0 в клетку 99, переходя от одной клетки к другой.

Техническое требование

Программа во время своей работы должна сделать ход не более чем за 10 секунд. При каждом запуске программа может читать файлы: *maze.txt* и *temp.tmp*. *maze.txt* хранит текущий путь до какой-то клетки лабиринта. Начало этого пути всегда в клетке с номером 0. В пути все клетки должны быть соседними и все, кроме, может быть, последней, должны быть пустыми. Номера клеток разделены пробельными символами (пробелы, табуляции, переводы строк). Перед запуском программы будет проверена последняя клетка пути в *maze.txt* и если там стена, она будет убрана из *maze.txt*, иначе *maze.txt* не изменится. После окончания работы программы в *maze.txt* должен находится путь, последняя клетка которого будет проверена перед очередным запуском Вашей программы. Если номер последней клетки в пути окажется равен 99, то будет проверен весь путь и, если он окажется верным, то задача считается решённой.

Программа может иметь только один временный файл temp.tmp размером не более 10 байт.

Цель

Найти путь за минимальное количество запусков Вашей программы.

Пример

<запуск проверки пути>
В *maze.txt*: 0
<запуск программы участника>
В *maze.txt*: 0 1 2 3 4 5 6 7 8 9
<запуск проверки пути>
В *maze.txt*: 0 1 2 3 4 5 6 7 8
<запуск программы участника>
В *maze.txt*: 0 1 2 3 4 5 6 7 8 18 19 29 39 49 59 69 79
<запуск проверки пути>
В *maze.txt*: 0 1 2 3 4 5 6 7 8 18 19 29 39 49 59 69 79
<запуск программы участника>
В *maze.txt*: 0 1 2 3 4 5 6 7 8 18 19 29 39 49 59 69 79 89 99
<запуск проверки пути>
Количество ходов на поиск пути: 3

Задание 1

Написать программу для выполнения цели, которая описана выше.

Задание 2

Предоставить файл *maze.txt*. В файле *maze.txt* записаны номера всех клеток, где находятся стены, т.е. хранится лабиринт. Эти лабиринты будут участвовать в тестировании программ участников. Сами тесты баллов не приносят, но могут снизить количество баллов у соперников.

16.174. **Многогранник.** Найти объем многогранника с целочисленными координатами вершин. Все грани многогранника — треугольники. Не треугольные грани могут задаваться несколькими треугольниками, лежащими в одной плоскости. Объем требуется выдать точно в виде несократимой дроби. Многогранник может быть и невыпуклым. Форма многогранника не ограничена. Например, он может быть подобием бублика, или состоять из двух несвязанных частей, или быть подобием двух продетых друг в друга звеньев цепи, или быть подобием бублика, завязанного узлом. Многогранник может даже содержать внутренние пустоты, в которых могут содержаться отдельные части многогранника (типа «матрешки»). Выдать в этом случае нужно объем тела многогранника за вычетом всех пустот.

Техническое требование

Имя входного файла: *input.txt*

Имя выходного файла: *output.txt*

Ограничение времени тестирования: 1 секунда на 1 тест

Формат входных данных

Входной файл содержит запись целых неотрицательных чисел в де-

сятичной системе счисления. Последовательно задаются все грани многогранника. Каждая грань задается последовательно тремя тройками неотрицательных целых чисел — координатами вершин (сначала — координата x, потом — y, и, наконец, — z). Вершины всех граней при этом обходятся обязательно в одном и том же направлении: все — по часовой стрелке или все — против часовой стрелки, если смотреть на грань снаружи многогранника (то есть из пустоты, а не из тела многогранника) и так, чтобы эта грань не была заслонена другими гранями. Таким образом, если считать, что при задании каждой грани проходятся три ребра в определенном направлении, то в итоге каждое ребро проходится дважды: в одном направлении и в противоположном. Каждая вершина проходится столько раз, сколько граней (а, значит, и ребер) к ней примыкает. Грани между собой могут соприкасаться только своими ребрами. Например, ни одна грань не может рассекать другую посередине, находиться внутри другой грани или соприкасаться с другой гранью вершиной. Все числа не превышают 15, разделяются одиночными или кратными пробелами или переводами строк. Число граней не превышает 1000.

Формат выходных данных

Выходной файл содержит одну строку, в которой записаны только два целых числа в десятичной системе счисления: числитель и знаменатель несократимой дроби, выражающей объем многогранника. Числа разделяются одним пробелом. Знаменатель должен быть положительным. Никаких других символов в строке быть не должно. Числитель может быть больше знаменателя. Незначащих нулей в начале чисел быть не должно.

Пример

input.txt

```
0 0 1  0 1 0  1 0 0
0 0 0  0 0 1  1 0 0
0 0 0  0 1 0  0 0 1
0 0 0  1 0 0  0 1 0
```

output.txt

```
1 6
```

16.175. Помощник риэлтера.

Первоначальный вариант задачи риэлтера см. 14.23

Затем, (см. 16.172) формулировка была изменена.

Задача

Составить программу, которая по файлам *exchange.txt* и *input.txt* (*output.txt* первоначальной формулировки) выдает *output.txt* из изменённой формулировки.

Техническое требование

Ограничение времени 10 секунд. Максимальное количество квартир (продаваемых и желаемых) не более 10. Максимальная стоимость квартиры — целое число от 1 до 10000. Максимальное количество комнат в квартире от 1 до 9. Максимальное количество обменов (количество строк) в exchange.txt — 100000. Все цены (на квартиры, доплаты и выручки) — целые числа от 0 до 10000.

16.176. Гексагональные шашки. Написать программу для игры в гексагональные шашки. Единственное, чем они отличаются от обычных русских шашек, так это то, что доска состоит не из квадратов, а из шестиугольных ячеек. Таким образом, у каждой гексагональной ячеки может быть до 6 соседних ячеек. Корректировка правил для этого случая является дополнительной задачей, подумайте.

16.177. Экспансия. Территория страны разбита на N районов. Военная база, расположенная в произвольном районе, контролирует собственный район и те, которые имеют с ним общую границу. Найти минимальное количество военных баз, контролирующих всю территорию, и указать районы, в которых они расположены.

Техническое требование

$N \leqslant 1000$. Данные о взаимном расположении районов хранятся в файле *country.txt* Других районов, кроме перечисленных в этом файле, нет. Ответ нужно записывать в файл *base.txt*.

Формат country.txt

```
<N-Число пар районов с общей границей>
<Номер одного района 1> <Номер другого района 1>
...
<Номер одного района N> <Номер другого района N>
```

Формат base.txt

```
<S-Число баз>
<Номер базы 1>
...
<Номер базы S>
```

Пример

В country.txt

```
4 1 2 3 1 1 4 5 4
```

В base.txt

```
2 1 5
```

ОЦЕНКА

Программа считается лучше, если для полного контроля использует меньше баз.

16.178. Топологическая карта. Топологическая карта задаётся следующим образом. В файле input.map указываются регионы с общей границей, все регионы пронумерованы числами от 1 до 20, т.е. регионов всегда 20.

Формат input.map

```
<Число пар>
<Номер региона 1> <Номер региона 2>
...
<Номер региона n> <Номер региона m>
```

Пример

В input.map

```
3
1 10
10 20
20 1
```

Задача. По заданной топологической карте вывести её рисунок.

Техническое требование

Рисунок выводится на графический экран. Регионы на экране представляются в виде целой (без дыр и разрывов) области. Для различения регионов с общей границей эти области должны быть покрашены в разные цвета. Все области должны иметь номер изображаемого региона. Фон должен быть чёрным, а номера – белые. Всего цветов на экране должно быть не более 16.

16.179. Вам предложили написать программу, решающую уравнения вида $f(x) = 0$, где $f(x)$ — произвольная функция, а x — действительное число. Сколько времени Вам на это понадобится?

16.180. Вам предложили написать программу, решающую уравнения вида $P(x) = 0$, где $P(x)$ — произвольный многочлен, а x — действительное число. Сколько времени Вам на это понадобится?

16.181. Установите взаимосвязи между следующими понятиями программирования: язык, синтаксис, семантика, прагматика, стиль, метод, технология, модель, тип, тестирование, алгоритм, качество, теория и политика.

16.182. Бесполезные комментарии. Часто преподаватели требуют, чтобы программа была снабжена подробными комментариями. Некоторые «умные» студенты пишут в комментарий пояснения действий, а не сути.

Пример

```
main()                // начало программы
{int i,s;             // описание i и s
  for(i=1;i<=10;i++) // цикл для i
    s+=i;             // увеличение s на i
  printf(s);          // печать значения s
}
```

Напишите программу для такого комментирования программы.

Решите эту задачу для языков разных стилей программирования. Интересны следующие языки: Pascal, РЕФАЛ, Prolog, Lisp.

16.183. Тест к уравнению $\mathbf{x + 2 \cdot x + 3 \cdot x + ... + x \cdot x = N}$. Для заданного K найти максимальное $N < K$ такое, чтобы уравнение

$$x + 2 \cdot x + 3 \cdot x + ... + x \cdot x = N$$

имело решение. Вводится K, выводится N.

Пример

```
K= 20
N=18
```

16.184. ✠ Счастье в одной цифре. Число называется *счастливым*, если сумма правой половины его цифр равна сумме левой половины. Если число состоит из нечётного числа цифр, нужно добавить слева один ноль.

Задача

В заданном числе заменить одну цифру так, чтобы получившееся число оказалось счастливым. Написать программу для нахождения получившегося счастливого числа.

Техническое требование

Десятичная запись числа содержит не более 20 знаков. Если вариантов несколько, то выдать тот, где число получается максимальным.

Пример

```
Число: 9112004
Результат: 9117004
```

16.185. ✣ **Частые числа.** Строится последовательность чисел.

$$\begin{aligned} X_1 &= (N+1)\#(N+1), \\ &\dots \\ X_i &= (X_{i-1}+i)\#(X_{i-1}+i), \\ &\dots \\ X_N &= (X_{N-1}+N)\#(X_{N-1}+N). \end{aligned}$$

Выдать число, которое встречается чаще других в этой последовательности. Если таких чисел несколько, то выдать самое маленькое из них. Число $0 < N < 100\,000$ задаётся пользователем.

Операция # берёт от левого числа последние две цифры, от правого — первые две и соединяет их в одно число. Если цифр не хватает, то берут незначащие нули (слева). Например, $12345\#12345 = 4512$, $1\#2 = 0102$

Пример

```
N= 10
Ответ:  1111
```

16.186. ✣ **Перемешанная строка.** Под *перемешиванием строки* будем понимать следующий процесс:

1) берём заданную строку;
2) если символов нечётное количество, то добавляем в начало строки пробел, получаем исходную[8] строку;
3) делим строку пополам, получаем правую и левую строки;
4) новая строка получается сцеплением правой, исходной и левой строк (правая оказалась слева, левая справа);

При повторении результат берётся в качестве исходной строки. Например, для строки «строчка» последовательность такова:

1) строчка
2) ␣строчка
3) ␣стр очка
4) очка␣строчка␣стр

Описанную последовательность выполняем 2004 раз. Определить в получившейся строке символ, который будет находиться на 2004 месте.

Техническое требование

Длина входной строки не превышает 100 символов. На вход подаются только непустые строки.

Пример

```
Строка: строчка
На 2004 месте находится символ: а
```

[8] Обратите внимание, что в этой задаче «исходная строка» и «заданная строка» означают разные понятия.

16.187. Мат. Заданные фигуры расставить на классической шахматной доске, так чтобы чёрному королю был объявлен мат. Фигуры нужно использовать все. Напишите программу, которая решает эту задачу.

16.188. Ошибки-описки. Найдите ошибки, описки, неточности и прочие изъяны в данном задачнике.

Часть II

Решения, ответы, подсказки, тесты и намёки

Глава 1

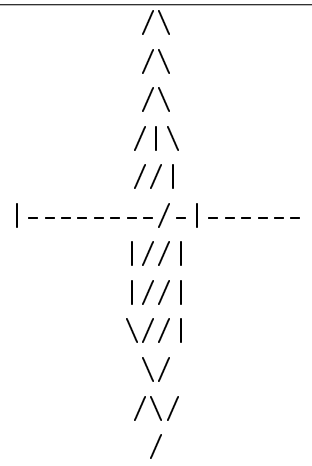
Введение в систему понятий программирования

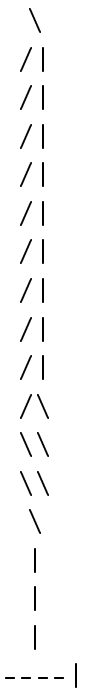
1.3. Нужно перевозить в такой последовательности: туда козу, обратно мужик возвращается один, туда капусту, обратно козу, туда волка, обратно один, туда козу.

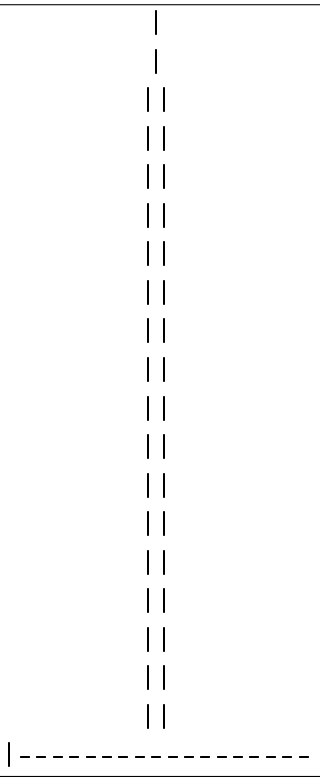
Очевидно, эту задачу лучше решать в сентенциальном стиле. Например, на языке Prolog.

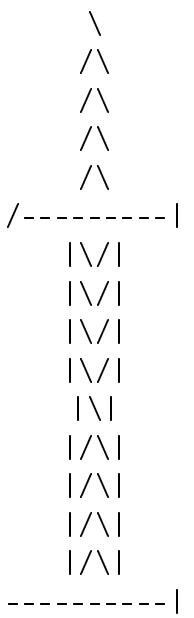
1.6.

N	Вход	Выход

0	0310r13R8rR20ll4Lr04L9rrr4LL1lRlRR3	

1	0101R6L5l4R11R2r10l03	

2	2020LL20llllll19rLrrr18Rllllll10	

3	0502RL11R11rR11Ll11lL11R6R6r11	

4	`2020R21rrr41Rr41R02`	--
Время работы программы 1 секунда		

1.7.

N	Входные данные	Верный ответ
0	N=100	60
1	N=20	12
2	N=1999	1680
3	N=1	1
4	N=720720	720720
5	N=1000000000	821620800
6	N=2000000000	1643241600
7	N=205405605	205405200

1.8.

N	Входные данные	Верный ответ
0	`-(-(X+(-50))-(X-6))`	`+28/1`
1	`X+1+X+2+X+3+X+4`	`-5/2`
2	`-(-(-(-(-(-(X)))))))+987`	`+987/1`
3	`X+(X+X+X+X)+X-2 000 000 002`	`+1 000 000 001/3`
4	`(((X+1)+X+1)+X+1)+X`	`-3/4`
5	`(X+1)-(1+X)`	`X-любой`
6	`(X-2002)+(2003-X)`	`Нет решений`
7	`-X+1-X+12-X+123`	`+136/3`
8	`X-(10-0-(-5+(X+X)-6)+X-123)`	`-51`

1.10.

N	Входные данные	Верный ответ
0	Res= 1999	771
1	Res= 200	100
2	Res= 9876	Таких нет
3	Res= 99	99
4	Res= 222 222 222	Нет таких
5	Res= 26975	10000
6	Res= 111 111 110	41079055
7	Res= 1 999 999 995	739422455
Время работы программы 5 секунд		

1.12.

N	Входные данные		Верный ответ
0	M= 1	N= 2	4
1	M= 1	N= 123	125
2	M= 0	N= 1000	1001
3	M= 2	N= 250	503
4	M= 3	N= 3	61
5	M= 4	N= 0	13
Время работы программы 1 секунда			

1.13.

N	Вход	Выход
0	X=6	8
1	X=10	52
2	X=2	1
3	X=1	1
4	X=40	102334155
Время работы программы 1 секунда		

1.14. Основа решения на языке Pascal, требующего порядка $log(n)$ умножений[1]:

```
Y := 1;
while n>0 do
begin if n MOD 2 = 1
      then begin Y :=Y * X; n := n - 1 end;
    n := n DIV 2; X := X * X
end
```

Тесты:

N	Вход		Выход
0	X=2	n=10	1024
1	X=10	n=2	100
2	X=3	n=3	27
3	X=10	n=9	1000000000
4	X=1	n=1234567890	1
5	X=1234567890	n=1	1234567890
Время работы программы 1 секунда			

[1] Традиционно для $log_a b$ используют сокращения: при $a = 10$ записывают $lg\ b$, при $a = 2$ — $log\ b$.

Глава 2

Синтаксис, семантика и прагматика языка программирования

2.7. Семантическая эквивалентность имеется для случаев б), в), к) и л). Во всех остальных случаях ответ отрицательный, так как выражения, обозначаемые заглавными буквами, могут вызывать подпрограммы, меняющие контекст выполняемых фрагментов.

2.50. Два фрагмента не эквиваленты по нескольким причинам. Для нахождения одной из них дадим подсказку: когда вычисляются (и как изменяются) начальное и конечное значения параметра цикла for?

Глава 3

Стили программирования, или программирование с птичьего полета

3.6.

N	Вход	Выход
0	-1␣2␣3+␣X␣=␣-␣1␣␣2␣␣␣3␣␣␣␣4␣	X = -1111
1	987+X=1230987	X = 1230000
2	␣5600␣+X␣=␣5612␣	X = 12
3	-␣123␣␣4+X=␣-1234	X = 0
4	-␣123␣␣4+X=␣-123	X = 1111
5	␣0␣+X=␣-9	X = -9
Время работы программы 3 секунды		

3.8.

В данном наборе тестов «визуально» означает, что надо посмотреть, действительно ли числа подходят.

N	Вход		Выход
0	11	2	Визуально
1	2	2	Визуально
2	999999	999998	Визуально
3	3	999999999	Визуально
4	-2	1	Визуально
5	7	-3	Визуально
6	1	0	НЕТ ЧИСЕЛ
Время работы программы 3 секунды			

3.11.

0	Вход	`abababABabAbbccccccccccc`
	Выход	`4ab1ABabAbb11c` или `4ab1ABabA2b11c`
1	Вход	`ccccccccccbbAbaBAbababababa`
	Выход	`11c2b1AbaBA4ba` или `11c1bbAbaBA4ba`
2	Вход	`XxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxX`
	Выход	`1X30x1X`
3	Вход	`QWERTYUIOPASDFGHJKLZXCVBNM`
	Выход	`1QWERTYUIOPASDFGHJKLZXCVBNM`
4	Вход	`AhhhhhhhhhHhHhHhHhHhHhHhH`
	Выход	`1A9h10hH`
5	Вход	$\underbrace{aabb\ldots aabb}_{15}$
	Выход	`15aabb`
6	Вход	$\underbrace{M\ldots M}_{80}$
	Выход	`80M`
7	Вход	`abababcababababcababababc`
	Выход	`3abababc`
8	Вход	`abababababababcabababababababcabababababababc`
	Выход	`7ab1c7ab1c7ab1c`
9	Вход	`aaaaaaaaaababababababababababbbbbbbbbb`
	Выход	`9a10ab9b`
Время работы программы 3 секунды		

3.13.

Ошибка состоит в том, что пять заместителей могут действовать независимо друг от друга и послать каждый по одному человеку. Вряд ли хорошо повлияет на репутацию фирмы появление на встрече нескольких представителей. Могут подумать: «Ну и бардак у вас...».

Правильные способы действия те же, что и в распределённых системах. Нужно обеспечить появление на встрече ровно одного представителя.

Глава 4

Понятие жизненного цикла программного обеспечения и его модели

4.2. Тесты:

N	Вход			Выход
0	1	2	3	X = -2
1	10	20	30	X = -20
2	1	1	1	Ответа нет
3	1	-20	100	X = 10
4	-1	2	1	Ответа нет
Время работы программы 3 секунды				

4.23. В соревнованиях, где учитываются только пройденные тесты и не учитывается полное решение задачи, итеративный подход приносит неплохие результаты. Если не сосредотачиваться на полном решении, то можно порешать одну задачу, потом перейти ко второй и так далее по кругу. Если удаётся решить какую-нибудь задачу до конца — очень хорошо, а нет — тоже не страшно.

Подсчитаем, какова эффективность такого подхода. Пусть даны 4 задачи. Мы решили каждую в среднем на 50%. Получается 50% баллов — примерно как за 2 полных задачи. Но решить 4 задачи наполовину, обычно, легче, чем две полностью.

Глава 5

Выражения

5.7.

а) 14; б) 5202;
в) 12254; г) −19961;
д) 4293; е) −3878;
ж) 19812; з) −1031.

5.8.

N	Вход	Выход
0	Пупышев Вячеслав Викторович	Пупышев В.В.
1	Сидоров Сидор Сидорович	Сидоров С.С.
2	Пушкин Александр Сергеевич	Пушкин А.С.
3	И К Е	И К.Е.
Время работы программы 0 секунд		

5.29. Обозначим неизвестное целое количество лет x. Тогда количество свечей, потраченное на x лет равно $\frac{x(x+1)}{2}$ (сумма первых x элементов арифметической прогрессии с начальным членом 1 и разностью 1). Значит, искомое x– наибольшее натуральное число, которое удовлетворяет неравенству

$\frac{x(x+1)}{2} \leqslant N$. Решив его, находим x: $x = [\frac{-1+\sqrt{1+8N}}{2}]$, где квадратные скобки означают целую часть числа (наибольшее целое, не превосходящее данного числа).

5.32.

N	Вход		Выход
0	2	0	На ноль делить НЕЛЬЗЯ !!!
1	6	3	6 : 3 = 2
2	-10	0	На ноль делить НЕЛЬЗЯ !!!
3	0	0	На ноль делить НЕЛЬЗЯ !!!
4	0	100	0 : 100 = 0
5	543210	0	На ноль делить НЕЛЬЗЯ !!!
Время работы программы 0 секунд			

5.42.

N	Вход			
0	1	1	3	
1	2	2	100	
2	3	3	1000	
3	4	4	10000	
4	100	100	31997	
5	1000	1000	31991	
6	2000	2000	31991	
7	21990	10000	31991	
Время работы программы 10 секунд				

Глава 6

Разветвление вычислений

6.12.

N	Вход	Выход
0	9889	Счастливое
1	1212	Счастливое
2	9876	Не счастливое
3	5555	Счастливое
4	1000	Не счастливое
Время работы программы 0 секунд		

6.19.

Введите фамилию, отчество, имя и что-нибудь другое.

6.20. В тесте 5 выход может быть любым.

N	Вход	Выход
0	1	Очень плохо
1	3	Удовлетворительно
2	5	Отлично
3	4	Хорошо
4	2	Плохо или неудовлетворительно
5	6	
Время работы программы 0 секунд		

6.21.

N	Вход	Выход
0	-3	Не оценка
1	1	Очень плохо
2	3	Удовлетворительно
3	5	Отлично
4	4	Хорошо
5	2	Плохо или неудовлетворительно
6	-35	Не оценка
7	9999	Не оценка
Время работы программы 0 секунд		

6.34. Тесты для решения биквадратного уравнения:

N	Вход	Выход
0	1 1 0	0
1	1 -2 1	-1 1
2	1 -1 0	0 -1 1
3	1 -5 4	-1 1 -2 2
4	1 3 2	Нет корней
5	1 2 3	Нет корней

6.35. Программа должна проходить все тесты задачи 6.34 (см.выше) и, кроме этого, следующие тесты:

N	Вход	Выход
6	0 1 -4	-2 2
7	0 20 0	0
8	0 1 100	Нет корней
9	0 0 -1	Нет корней

6.46.

N	Вход	Выход
0	3 8 19 11	3 8 11 19
1	3 8 11 19	3 8 11 19
2	3 -8 19 11	-8 3 11 19
3	11 99 5 11	5 11 11 99
4	-2 -2 -2 -2	-2 -2 -2 -2
Время работы программы 0 секунд		

Глава 7

Циклические вычисления

7.1.
10 11 12 13 14 15 16 17 18 19 . . . 97 98 99 100

7.5.
2 4 6 8 10 12 14 16 18 20 22 24 . . . 96 98 100

7.35. Очевидно, что нужно выделить общие части в факториалах и каждый раз заново их не пересчитывать. Если для нахождения коэффициента бинома Ньютона прямо применять основную формулу (см. задачу 7.25), то количество умножений будет равно $2n - 2$. Если же применить оптимизацию, то оно не превзойдет $n-1$. Кроме этого, на единицу можно тоже не умножать.

7.42.
Входные и выходные данные описаны примерно.

N	Вход	Выход
0	«Текст задачи»	2
1	aaabbbbcddddffZZZZZh XXXXXYYYYYYYYsdvdfvdfvsdfv	8

2	$\underbrace{11111\ldots111}_{78}$ $\underbrace{aaaaa\ldots aaa}_{78}$ $\underbrace{22222\ldots222}_{78}$ $\underbrace{bbbbb\ldots bbb}_{78}$ $\underbrace{33333\ldots333}_{78}$ $\underbrace{ccccc\ldots ccc}_{78}$ $\underbrace{44444\ldots444}_{78}$ $\underbrace{ddddd\ldots ddd}_{78}$ $\underbrace{55555\ldots555}_{78}$ $\underbrace{eeeee\ldots eee}_{78}$ $\underbrace{66666\ldots666}_{78}$ $\underbrace{fffff\ldots fff}_{78}$ $\underbrace{77777\ldots777}_{78}$ $\underbrace{ggggg\ldots ggg}_{78}$ $\underbrace{88888\ldots888}_{78}$ $\underbrace{hhhhh\ldots hhh}_{78}$ $\underbrace{99999\ldots999}_{78}$ $\underbrace{iiiii\ldots iii}_{78}$ $\underbrace{00000\ldots000}_{78}$	78
3	Обычный текст, около 150 килобайт. Обычно пробел встречается чаще всего, но для уверенности добавьте строку из 38 пробелов.	38
4	Большой текст, около 500 килобайт. Вставьте в текст сроку из 32 одинаковых символов.	32
5	Все символы по одному.	1

6	0 00 000 0000 00000 000000 0000000 00000000 000000000 0000000000 00000000000 000000000000 0000000000000 00000000000000 000000000000000 0000000000000000 00000000000000000 000000000000000000 0000000000000000000 00000000000000000000 000000000000000000000 0000000000000000000000	23

7	0a 00b 000c 0000d 00000e 000000f 0000000g 00000000h 000000000i 0000000000j 00000000000k 000000000000l 0000000000000m 00000000000000n 000000000000000o 0000000000000000p 00000000000000000q 000000000000000000r 0000000000000000000s 00000000000000000000t 000000000000000000000u 0000000000000000000000v 00000000000000000000000w	23
8	$\underbrace{SSSSSSSSS\ldots SSS}_{1048576}$	1048576
9	Пустой файл.	0
Время работы программы 3 секунды		

7.53.

N	Вход	Выход
0	aaa ёёёBBB	aaaёёёBBB
1	aaa ёёё ввв	aaaёёёввв
2	A	A
3	ааабббёёё	ааабббёёё
4	ЪЪЪбббввв	ЪЪЪбббввв
Время работы программы 1 секунда		

7.55.

N	Вход	Выход
0	3 2 -1 8 4	4 8 -1 2 3
1	1 3 5 7 9 11	11 9 7 5 3 1
2	11 12 11 12	12 11 12 11
3	1234567890	1234567890
4	-1 -1 -1	-1 -1 -1
5	3 3 3 3 3 3 3 3 3 3	3 3 3 3 3 3 3 3 3 3
Время работы программы 1 секунда		

7.57.

N	Вход	Выход
0	121000012343	Плавное
1	32101	Плавное
2	$\underbrace{7777777777777\ldots7}_{79}$	Плавное
3	987654321098	Нет
4	98765432101234567 89	Плавное
5	131313131313	Нет
6	1	Плавное
7	90	Нет
Время работы программы 1 секунда		

7.61.

В тесте 3 на вход подаётся пустая строка.

N	Вход	Выход
0	Мама мыла раму !	`В строке "Мама мыла раму !" букв "ы" - 1`
1	Мыла мама раму ! МЫЛА !	`В строке "Мыла мама раму ! МЫЛА !" букв "ы" - 1`
2	ЫыЫыЫыЫыЫы	`В строке "ЫыЫыЫыЫы"букв "ы" - 4`
3		`В строке "" букв "ы" - 0`
Время работы программы 1 секунда		

7.76. Задачу можно решить разными способами, в частности, методом решета.

7.82.

N	Вход	Выход
0	N = 12	(2,4), (11,12)
1	N = -111	Решений нет
2	N = 0	Решений нет
3	N = 1	(1,0), (0,1) и все пары (X,X), где X — целое неотрицательное число
4	N = 6	(0,3), (1,3), (5,6)
5	N = 9	(8,9)
6	N = 5765760	(10,16), (5765759,5765760)
7	N = 720	(0,6), (1,6), (7,10), (719,720)
Время работы программы 4 секунды		

Глава 8

Подпрограммы

8.6.

N	Вход	Выход
0	1 2 3 0	1) 1 2 3 2) 1 3 3 3) -1 1 1 4) 0 1 1
1	-1 1 -2 2	1) 1 1 2 2 2) 1 1 2 2 3) -1 -1 0 0 4) 0 0 0 0
2	-10	1) 10 2) 10 3) 8 4) 8
3	9 8 7 6 5 4 3	1) 9 8 7 6 5 4 3 2) 9 9 7 9 5 9 3 3) 7 7 5 7 3 7 1 4) 7 7 5 7 3 7 1
Время работы программы 1 секунда		

8.7.

N	Вход и выход
0	Вход: 12345678901234567890 Выход: +1 = 12345678901234567891 ∗1 = 12345678901234567890 ∗10 = 123456789012345678900 ∗2 = 24691357802469135780
1	Вход: 11111111111111111111 Выход: +1 = 11111111111111111112 ∗1 = 11111111111111111111 ∗10 = 111111111111111111110 ∗2 = 22222222222222222222
2	Вход: 99999999999999999999 Выход: +1 = 100000000000000000000 ∗1 = 99999999999999999999 ∗10 = 999999999999999999990 ∗2 = 199999999999999999998
3	Вход: 555555555555555555555 Выход: +1 = 55555555555555555556 ∗1 = 55555555555555555555 ∗10 = 555555555555555555550 ∗2 = 11111111111111111110
Время работы программы 1 секунда	

8.8.

N	Вход	Выход
0	x=12	8
1	x=99	36
2	x=1	5
3	x=9000	18
4	x=12345	396
5	x=46000	50
6	x=0	0
7	x=-101	16
Время работы программы 0 секунд		

8.17.

N	Вход	Выход
0	$a = A \;\; b = B \;\; c = C \;\; d = D$	x = C
1	$a = 1 \;\; b = 20 \;\; c = 3 \;\; d = 40$	x = 30
2	$a = 12345 \;\; b = 56789 \;\; c = 90ABC \;\; d = CDEF$	НЕТ РЕШЕНИЙ
3	$a = -A \;\; b = -A \;\; c = -A \;\; d = -A$	НЕТ РЕШЕНИЙ
4	$a = 1F \;\; b = F0 \;\; c = 1F \;\; d = -F0$	x-ЛЮБОЙ
5	$a = 12F \;\; b = -401E \;\; c = 56D \;\; d = 7C$	x = -F
6	$a = 12E \;\; b = -5CE \;\; c = F2 \;\; d = -40A$	x = 2A
Время работы программы 1 секунда		

8.18.

N	Вход		Выход
0	x=-2	y=3	1
1	x=-3	y=2	-1
2	x=0	y=0	0
3	x=10	y=20	30
4	x=-100	y=100	0
5	x=34	y=34	34
Время работы программы 0 секунд			

8.21. Механизм передачи параметров «по умолчанию» здесь, как нам кажется, неприменим, так как функция не сможет определить, какое именно количество параметров ей было передано (если, конечно, не применить что-нибудь хакерское). Лучше воспользоваться другими способами, например, определением перегруженной функции (статический полиморфизм).

8.23. Вероятно, для этого нужно привлечь механизм процедурных типов или указателей на подпрограммы.

8.37.

N	Вход	Выход
0	01902021	$190 + 20 = 210$
1	1212989911111	$1212 + 9899 = 11111$
2	$\underbrace{9\ldots9}_{79}$	Нет
3	29	Нет
4	101	$1 + 0 = 1$ или $0 + 1 = 1$
5	$\underbrace{1\ldots1}_{20}0\underbrace{9\ldots9}_{10}$	$\underbrace{9\ldots9}_{10} + \underbrace{1\ldots1}_{10} = \underbrace{1\ldots1}_{10}0$
6	911009	$9 + 91 = 100$ или $99 + 1 = 100$
Время работы программы 1 секунда		

Глава 9

Структуры данных

9.1.

N	Вход		Выход
0	Маша Коля Саша Валя	120 050 150 0130	`Саша`

1	Вася
	Оля
	Сима
	Вера
	Елена
		110
		120
		120
		110
		100
			`Сима`
			или
			`Оля`

2	Петя Петя Слава	100 120 110	`Петя`
3	Вова	200	`Вова`
4	Вова Павел Евгений	200 210 190	`Павел`
Время работы программы 1 секунда			

9.7.

N	Вход	Выход
0	123456789123456789	ДА
1	1982	ДА
2	9121288	НЕТ
3	5555555555555555555	ДА
4	9999999999999999999	НЕТ
5	268194	НЕТ
6	3	НЕТ
Время работы программы 1 секунда		

9.12.

N	Вход	Выход
0	1	1
1	2	Нет решений
2	6	2
3	10	Нет решений
4	1000	Нет решений
5	500500000	1000
6	2000000000	Нет решений
7	2146828125	1625
Время работы программы 1 секунда		

9.38. Заметим, что не все приведенные равенства являются тождествами.

9.43. Всю работу по представлению матрицы в виде «виртуального» линейного массива, который можно сортировать любым обычным методом сортировки, можно возложить на специальную функцию, вычисляющую координаты элемента в матрице по его «номеру» в змейке. Например, для случая расположения элементов, изображенного на рис. 9.1ж), можно вывести следующие формулы для координат k-го элемента змейки матрицы размера $N \times N$ (считаем, что строки и столбцы матрицы имеют индексы от 0 до $N-1$):

$$i = [k/n]; j = \begin{cases} k - i * N, & \text{если i – четное;} \\ (i+1) * N - k - 1, & \text{если i – нечетное.} \end{cases}$$

(Здесь квадратные скобки означают целую часть числа. Считается, что элементы змейки нумеруются подряд от нуля до $N^2 - 1$.)

Для других случаев расположения элементов матрицы формулы будут несколько сложнее – их вывод сам по себе представляет интересную

математическую задачу. Этот пример показывает, что предварительная математическая проработка решения задачи может упростить алгоритм (в смысле временной сложности вычисления) на несколько порядков.

Глава 10

Автоматное программирование

10.20. Вероятно, лучше создать настраиваемый препроцессор, состоящий из двух программных модулей: один из них — прототип интерпретатора, включающий исходный код, одинаковый для любого автомата. Другой модуль — переменная часть, зависящая от таблицы переходов (то есть определения конкретного конечного автомата).

10.26. В структуре, которая хранит азбуку Морзе, должно быть учтено, что несколько разных кодов букв могут начинаться одинаково. Поэтому, для устранения повторных просмотров (в том числе, скрытых в функциях сравнения строк) можно создать структуру типа графа или дерева.

10.51.

N	Вход	Выход
0	03112002	300
1	20022003	20
2	1	612579512
3	04	612579511
4	0	564151952
5	1000000000	1
6	123	6990001
7	2003	5999970
8	11111	46000
9	100000000	1
Время работы программы 3 секунды		

10.62. Вероятно, оптимальной структурой с точки зрения указанной рекомендации будет дерево (как в задаче про азбуку Морзе 10.26).

10.63. Для примера приведем цепочку превращений из «мухи» в «сло-

на»: муха - мура - тура - тара - кара - каре - кафе - кафр - каюр - каюк - крюк - урюк - урок - срок - сток - стон - слон.

10.64. Муха - хула - луна - лунь - ноль - слон. Вообще, посмотреть различные игры такого рода можно на http://www.xword.ru.

10.79.

N	Вход	Выход
0	`7abcd20ab2003cd1cdEFGH abcdc`	67
1	`100a100b100c b`	101
2	`123456a123456b123456c bc`	246912
3	`987654ab2cd100e abcde`	1975307
4	`999999qwertyuiopasdfghjkl2zz zzz`	18999982
5	`2zz999999qwertyuiopasdfghjkl zzz`	1
6	`999999zzZzz2z999999zzZzz ZZ`	0
7	`999999zzZzz2z999999zzZzz zzzzz`	4999994
Время работы программы 3 секунды		

Глава 11

Методы, основанные на рекурсии

11.17.

makkarti(n) = max(n - 10, 91).

11.18.

cadiou(x, 0, 1) = x!.

11.24. Для сокращения перебора цепочек было бы неплохо пользоваться уже проведенными ранее вычислениями, то есть использовать ранее построенные цепочки.

11.26.

N	Вход		Выход
0	X= 3	Y= 4	16
1	X= 2	Y= 2	8
2	X= 1	Y= 1	4
3	X= 1	Y= 49	129
4	X= 1000	Y= 17	1048
5	X= 100	Y= 20	152
Время работы программы 50 секунд			

11.56.

n	Количество
2	1
3	2
4	24
5	1344
6	1128960

11.58. Попытки доказать или опровергнуть эту гипотезу предпринимались в течение более чем 200 лет и профессионалами, и любителями математики. Однако все затраченные усилия были напрасными. И только лишь в 1967 году с помощью компьютера американскими учеными Л.Лендером и Т.Паркином гипотеза Эйлера была опровергнута. Составленная ими программа выдала следующий результат:

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5.$$

11.69.

Ниже приведены тесты в виде пар <лабиринт, количество убираемых стен>.

```
# ########## 10   # ########## 1   # ##########   2
#############      #  ####### #     #    #     #
# # # # # ##      ##  ##### ##     #    #     #
## # # # # #      ###  ### ###     #    #     #
# # # # # ##      #### #  ####     #    #     #
## # # # # #      ####### ####     #    #     #
# # # # # ##      ####### ####     #############
## # # # # #      #### #  ####     #    #     #
# # # # # ##      ### ###  ###     #    #     #
## # # # # #      ## #####  ##     #    #     #
#############      # #######  #     #    #     #
########## #      ########## #     ########## #


# ##########  6   # ########## 19  # ##########  0
#  ##  #  ##      #############     #          #
# ##  #  ###      #############     #          #
###  #  # ##      #############     #          #
##  #  #  ##      #############     #          #
#  #  #  ###      #############     #          #
# #  #  # ##      #############     #          #
##  #  #  ##      #############     #          #
#  #  #  ###      #############     #          #
# #  #  #  #      #############     #          #
#########  #      #############     #          #
########## #      ########## #     ########## #
```

```
# ##########  0   # ##########  5  # ##########  4
#    #     #      #          #     #    ##    #
###  #  #  #      ############     # ######## #
#    #  #  #      # # # # #  #     # #  #   # #
#  #### ####      ############     # # #### # #
#         #      #  #  #  # #     ### #  #####
### #### ###      ############     #####  # ###
#    #     #      #  #    #  #     # # #### # #
#  # #  #  #      ############     # #   #  # #
#  # #######      #  # #######     # ######## #
#  #       #      #  #      ##     #    ##    #
########## #      ########## #     ########## #
```

```
# ##########  3
# # ### #  #
#  #### ## #
# # ### ## #
##  ### ## #
# # ### ## #
# # ### ## #
# # ### ## #
##  ### ## #
# # ### ## #
#    #  ## #
########## #
```

11.76.

Наверное, Вы заметили, что программа должна выдавать «Да» или «Нет» не как обычно, а наоборот.

N	Вход		Выход
0	`[a[b]]c`	`ab`	ДА
1	`stststs`	`stststs`	НЕТ
2	`+++++++++`	`+++++++-+`	ДА
3	`[1][1][1]`	`11`	НЕТ
4	`[[9][0]0]`	`9000`	ДА
5	`[[[[1][2]][3]][4]][5]`	`25`	НЕТ
6	`[[[[1][2]][3]][4]][5]`	`1243`	ДА
7	`[[[[1][2]][3]][4]][5]`	`1`	НЕТ
8	`[3[3]3]`	`3`	ДА
9	`0[0[0[0[0[0[00]]]]]]`	`0000000`	ДА
Время работы программы 3 секунды			

11.85. Такой пример привести невозможно. Всегда вместо рекурсии можно использовать цикл и стек.

Глава 12

Объектно-ориентированный подход

12.31. Вот некоторые из возможных средств: перегрузка (переопределение) функции; шаблон функции; шаблон класса; функция с переменным числом параметров.

Глава 13

Сентенциальные методы

13.24. Маленькие тесты:

Тест 0: *3* *Фирма "Рога и копыта" покупает акции фирмы "Копыта и рога"* *Фирма "Рога и копыта" не покупает акции фирмы "Копыта и рога"* *Фирма не "Копыта и рога" покупает акции фирмы "Копыта и рога"* *1* Ответ: `Фирма не "Копыта и рога" покупает акции фирмы "Копыта и рога"`
Тест 1: *3* *Фирма "МАРК" не покупает акции фирмы "Копыта и рога"* *Фирма "Рога и копыта" не покупает акции фирмы "Копыта и рога"* *Фирма "Копыта и рога" не покупает акции фирмы "Копыта и рога"* *1* Ответ: `Ничего не ясно`
Тест 2: *12* *Фирма не "1" не покупает акции фирмы "1"* *Фирма "2" не покупает акции фирмы "2"* *Фирма "1" не покупает акции фирмы "3"*

Фирма "1" не покупает акции фирмы не "1"
Фирма "1" не покупает акции фирмы не "3"
Фирма "2" не покупает акции фирмы не "2"
Фирма "2" покупает акции фирмы не "3"
Фирма "3" покупает акции фирмы "3"
Фирма не "4" покупает акции фирмы "4"
Фирма "2" не покупает акции фирмы "1"
Фирма не "2" покупает акции фирмы "4"
Фирма "3" не покупает акции фирмы "3"
7
Ответ:
Фирма "2" покупает акции фирмы не "3"

Тест 3:
11
Фирма "A" покупает акции фирмы "A"
Фирма "A" покупает акции фирмы "B"
Фирма "A" покупает акции фирмы "C"
Фирма "A" покупает акции фирмы "D"
Фирма "C" покупает акции фирмы "C"
Фирма "A" не покупает акции фирмы "A"
Фирма "B" не покупает акции фирмы "A"
Фирма "C" не покупает акции фирмы "A"
Фирма "D" не покупает акции фирмы "A"
Фирма "A" не покупает акции фирмы не "E"
Фирма не "E" покупает акции фирмы "A"
2
Ответ:
Фирма "C" покупает акции фирмы "C"

Тест 4:
8
фирма не "E" покупает акции фирмы "A"
фирма "D" не покупает акции фирмы не "E"
фирма "D" покупает акции фирмы "C"
фирма "D" покупает акции фирмы "B"
фирма "D" покупает акции фирмы "D"
фирма "C" не покупает акции фирмы "A"
фирма "B" не покупает акции фирмы "A"
фирма "A" не покупает акции фирмы "A"
1
Ответ:

```
Ничего не ясно
```

Тест 5:

5
Фирма "Рога" не покупает акции фирмы "Копыт"
Фирма "Рога" покупает акции фирмы "Копыт"
Фирма "Укроп" не покупает акции фирмы "Копыт"
Фирма "Укроп" покупает акции фирмы "Копыт"
Фирма не "Рогатый укроп" не покупает акции фирмы не "Укропный Копыт"
2

Ответ:

```
Фирма не "Рогатый укроп" не покупает акции фирмы не
"Укропный Копыт"
```

Тест 6:

6
Фирма "A" покупает акции фирмы не "E"
Фирма "A" не покупает акции фирмы "A"
Фирма "A" не покупает акции фирмы "C"
Фирма "A" не покупает акции фирмы "D"
Фирма "A" не покупает акции фирмы "B"
Фирма "A" покупает акции фирмы "B"
1

Ответ:

```
Фирма "A" покупает акции фирмы не "E"
Фирма "A" не покупает акции фирмы "A"
Фирма "A" не покупает акции фирмы "C"
Фирма "A" не покупает акции фирмы "D"
Фирма "A" покупает акции фирмы "B"
```

Время работы программы 50 секунд

13.37.

N	Входные данные	Верный ответ
0	00900963	5
1	11243733	5
2	12351298	8
3	43647251	7
4	57363817	6
5	00898009	4
6	91990008	3
7	44422556	0
8	01919190	?
9	22254442	?
Время работы программы 3 секунды		

13.45. ЧаВО по задаче 13.45:

Что представляет из себя проверка и как проверяющая программа ищет этот обман?

Доигрывает до конца. В конце число становится известно, а затем проверяет каждый ход.

Вы писали: "...отгадыватель записывает девятизначное число (если нужно дополнив незначащими нулями)". Что означают незначащие нули? Учитываются ли они загадывателем?

Незначащие нули — нули записанные слева в числе. Например, 000012345. Это тоже число, что и 12345, только девятизначное. Конечно, загадыватель должен учитывать все девять цифр.

Можно ли использовать временные файлы?

Нет. Вся информация извлекается из numbers.txt

13.49. ЧаВО по задаче 13.49:

Могут ли встретиться некорректные входные данные? Например оба игрока получают по два очка, или сумма не совпадает.

Да. Тогда нужно написать в correct.txt единственное число 0.

Однозначно ли можно восстановить таблицу результатов в случае корректного ввода?

Не всегда. И тогда правильным ответом будет любой из верных вариантов.

Тесты к задаче

0.

```
3
01 02 03 Всего
 X  ?  ?  ?
 1  X  ?  1
 ?  ?  X  2
```

1.

```
3
01 02 03 Всего
 X  ?  ?  ?
 1  X  ?  1
 ?  ?  X  2
```

2.

```
4
01 02 03 04 Всего
 ?  ?  ?  ?  6
 ?  ?  ?  ?  4
 ?  ?  ?  ?  2
 ?  ?  ?  ?  0
```

3.

```
4
 1  2  3  4  Всего
 ?  ?  ?  ?  ?
 ?  ?  ?  ?  ?
 ?  ?  ?  ?  ?
 ?  ?  ?  ?  ?
```

4.

```
2
 1  2  Всего
 X  ?  1
 ?  X  2
```

5.

```
5
 1  2  3  4  5  Всего
 X  ?  ?  ?  ?    3
 ?  X  ?  ?  ?    1
 ?  ?  X  ?  ?    0
 ?  ?  ?  X  ?    1
 ?  ?  ?  ?  X    3
```

6. Поле размером 50 из одних '?'.

7. Поле размером 49 полностью целое.

8. Поле размером 30 с с большим количеством '?', которое невозможно правильно восстановить.

9. Поле размером 48 с с большим количеством '?', которое имеет единственный вариант восстановления.

Глава 14

Функциональное программирование

14.7.

N	Вход	Выход
0	$X * ab + (c * X) = aabcabcac$	$X = ac$
1	$X * a + b = cabab$	$X = cb$
2	$X * (X * X) = zzzaazaazzaaza$	$X = za$
3	$(X + (X + X)) = xyxyxy$	$X = xy$
4	$X = abc$	$X = abc$
5	$b * X + X = bca$	Нет решений
6	$a * X * z = azzzaz$	$X = za$
7	$ax * w + X = awxwwwwww$	$X = wwwww$
8	$a * (X + X) * a = aababa$	$X = b$
Время работы программы 3 секунды		

14.14. ЧаВО по задаче 14.14:

Пример выходного файла для данного случая. Swintus Fingalae ((SKKs(SKIw)(IIIi)(KnK)tu)S) В исходном файле 1-й и последней скобочек не было.

Извините, ошибка. Поставьте их во входном файле.

И, как обычно, какова производительность процессора на проверяющем компьютере, ограничения по времени и памяти?

Время на тест 10 сек. Производительность приблизительно на PentiumIII-500

Имеется ли в виду, что непосредственно после преобразования (1) не может быть произведено преобразование (2) (и наоборот)? Если нет, то считается ли тем же самым фрагментом измененная

по другому правилу (S, K, или I) последовательность кодонов?

Как только применено преобразование (2), взятое в скобки выражение (выражение X), должно быть преобразовано по одному из правил: I или K или S.

Что такое «печальный исход»?

Печальный исход — когда преобразование приводит к слишком длинному коду или может продолжаться более 100 шагов.

Чем отличаются квалификация «Опасный» и «Фатальный»?

Возможностью получить печальный исход. Опасный эмбрион может иногда «закончить плохо». Фатальный эмбрион всегда «закончит плохо».

Некоторые тесты:

embrion.txt 0:
((SKKs(SKIw)(IIIi)(KnK)tu)S)
species.txt 0:
```
Swintus Fingalae
(swintuS)
Canis Fingali
((((ca)n)I)s)
Chimera Baldina
(SSSSS)
```

embrion.txt 1:
(S(KS)K(SI)Kfi)
species.txt 1:
```
Scuns Levini
(fi)
Cripta Archana
(if)
Transformer Olgin
((SII)(SII))
Mono A
(a)
Di A
(aa)
Tri A
(aaa)
Qart A
(aaaa)
Sept A
(aaaaaaa)
```

embrion.txt 2:

(S(S(S(S(S(SII)II)II)II)II)IIa)

species.txt 2:

```
Scuns Levini
(fi)
Cripta Archana
(if)
Mono A
(a)
Di A
(aa)
Tri A
(aaa)
Qart A
(aaaa)
Sept A
(aaaaaaa)
```

embrion.txt 3:

(K(S(S(S(S(S(SII)I)I)I)I)Ia)(SII(SII(SII(SII(SII(SII(SIIa)
))))))

species.txt 3:

```
Scuns Levini
(fi)
Cripta Archana
(if)
Mono A
(a)
Di A
(aa)
Tri A
(aaa)
Qart A
(aaaa)
Sept A
(aaaaaaa)
```

embrion.txt 4:

(SII(SII(SII(SII(SII(SIIa))))))

species.txt 4:

```
Scuns Levini
(fi)
Cripta Archana
```

(if) Mono A (a) Di A (aa) Tri A (aaa) Qart A (aaaa) Sept A (aaaaaaa)
embrion.txt 5: *((abc)def)* **species.txt 5:** Gen1 (ab(c)de) Gen2 ((((((a)b)c)d)e)f)
Время работы программы 5 секунд

14.17. ЧаВО по задаче 14.17:

Какая максимальная длина строки может быть у входных и выходных данных (тестов)?

Входные, как и выходные, данные тестов будут не длиннее 100 символов.

Может ли во входных данных (тестах) встречаться символ '-', например "sf-r32-e"?

Может, как и другие текстовые символы. Текстовые символы — символы, которые изображены на основной клавиатуре, на символьных клавишах.

14.23. ЧаВО по задаче 14.23:

Что значит, что «... '0' означает, что в момент времени i контактная пара номер j разомкнута[то есть эти точки соединены], '1' означает, что она замкнута». Ведь «разомкнуть» — означает разъединить, а не соединить.

Просим прощения за опечатки, следует читать: «разъединены».

Что значит в примере "Z2=2, Z2=4"и "R1=0, R1=1".

Следует читать: "Z2=2, U2=4" и "R1=0, R2=1".

14.24. ЧаВО по задаче 14.24:

Соседние пролеты всегда перпендикулярны?

Нет. Могут быть и под другими углами.

Могут ли яблони стоять на заборе или в узлах забора?

Нет, «все яблони должны быть внутри области огороженной забором».

Глава 15

Моделирование

15.29. Тесты:

N	Вход					Площадь
0	10	10	-10	-10	100	3711.99
1	-10	10	10	-10	100	3711.99
2	1	1	-1	-1	10	37.12
3	1	1	-1	-1	100	7412.83
4	0	0	0	0	10000	78539816.34
5	-1000	5	1000	0	10000	48669255.39
6	1	2	4	8	16	46.92
7	-1	-1	-1	-2	2	0.00
8	1	2	1	2	0	0.00
9	10	20	30	40	50	Короткая верёвка
Время работы программы 3 секунды						

Глава 16

Задачи, решаемые различными методами

16.11.

N	Вход	Выход
0	программа гамма	ДА
1	программа рога	ДА
2	программа мама	НЕТ
3	123456789012345678901 15	ДА
4	12345612345123412312 1 555	НЕТ
Время работы программы 1 секунда		

16.13.

N	Вход	Выход
0	6 3	ВИЗУЛЬНО
1	3 6	ВИЗУЛЬНО
2	4 4	ВИЗУЛЬНО
3	1 1	ВИЗУЛЬНО
4	1 26	ВИЗУЛЬНО
5	11 1	ВИЗУЛЬНО
6	10 25	ВИЗУЛЬНО
Время работы программы 1 секунда		

16.15. Тесты:

N	Вход			Один из возможных ответов
0	тоска	астра	астра	атлет
1	астра	астра	атлет	тоска
2	атлет	тоска	астра	астра
3	олово	олово	олово	олово
4	тоска	аванс	синяк	кадет
5	спорт	тоска	атлет	тезис
6	спорт	тоска	аванс	сеанс
7	опрос	синяк	кулак	кайло
8	казус	спорт	тоска	ацтек
9	опрос	спорт	тоска	авизо
Время работы программы 1 секунда				

16.33. Ответ задачи весьма прост и его теоретическое обоснование можно найти, например, в [4].

16.34. Если вы решили предыдущую задачу, то вывести закономерность для этой не составит особого труда.

16.47. Можно воспользоваться интересным свойством чисел Фибоначчи: $\forall f_i \forall f_j : f_i \in \Phi \wedge f_j \in \Phi \Rightarrow \text{НОД}(f_i, f_j) \in \Phi$, где буква Ф обозначает множество всех чисел Фибоначчи.

16.53. Очевидно, что для чисел с таким количеством цифр обычные алгоритмы перевода из системы счисления в другую на ЭВМ практически не применимы. Программировать операции с длинными числами тоже утомительно. Гораздо проще увидеть, что для систем счисления с основаниями R и Q, где $R = Q^n$, n – натуральное число, алгоритмы перевода основаны на вполне определенных соотношениях между цифрами разных систем.

16.69. Интересное решение этой задачи предложено М. С. Густокашиным на сайте http://algolist.manual.ru.

16.87.

N	Вход	Комментарий к тесту	Выход
0	x)a(;-)(x()(	Текст примера	10
1	(f(fgr()) ((x)((fg)(x)(	Недостаток скобок невелик	81
2	)x()xxx) ((()(c)e)0123456789)	Недостаток открывающих скобок слева	8
3	*)*)(x(x)x)*)*)*) (0(1)2(3(4)5(6)7)8)9 (+(+((x(x)x)+(+(+	Недостаток скобок и слева и справа достаточно велик	22127616
4	(((x)+(y)+((z)*5)-4)) (1.1*(6.55+(1-2)))+ (sqrt(p*(p-a)*(p-b)*(p-c)))	Большое сбалансированное выражение	0
5	(((((*)+(*))+ (((((((*))))))) +(((*)+(*)+(*))))	Большое почти сбалансированное выражение (недостаток двух закр. скобок)	1127
6	Пятнадцать человек на сундук мертвеца, Йо-хо-хо, И бутылка рома!	Длинный текст без скобок	0
7	**))))))))))-((((((((((	Недлинный текст с большим недостатком открывающих и закрывающих скобок	394044256
8		Пустой текст	0
9	1234567890 1234567890 1234567890 123456789) (1234567890 1234567890 1234567890 123456789	Длинный текст с недостатком одной откр. и одной закр. скобки	1600

16.88.

N	Вход	Выход
1	-1.2(34)	-1 116/495
2	123456789	123456789
3	10.333333333	10 333333333/1000000000
4	0.098(001122)	371216/3787875
5	-333333333.789(999999)	-333333333 79/100
6	999999999.99999999(9)	1000000000
7	987654321.87654321(0)	987654321 87654321/100000000
8	-000000000.0000(00000)	0

16.96. Для решения задачи нужно проделать некоторую предварительную работу, пользуясь методами математического анализа.

16.99.

N	Вход		Выход
0	A=160	B=112	$X = 8$
1	A=11111111	B=255	$X = 2$
2	A=123	B=123	$X = 10$
3	A=7	B=7	$X > 7$
4	A=100	B=101	Нет решений
5	A=21	B=1999999999	$X = 999999999$
6	A=192	B=38	Нет решений
Время работы программы 2 секунды			

16.113. Тесты:

N	Входные данные	Верный ответ
0	abc␣abc␣abc	2
1	1121231234	4
2	1234567890abcde	15
3	12345678900987654321	10
4	123123123123123123	3
5	$\underbrace{1\sqcup 2\sqcup \ldots 1\sqcup 2\sqcup}_{80}$	2
6	20.12.2001-64,56,6445	5
7	$\underbrace{+++\ldots++}_{80}$	1
8	это␣секретный␣тест	?
9	Это␣тест␣достаточно␣сложный␣тест	?
Время работы программы 3 секунды		

16.119. ЧаВО по задаче 16.119:

Что можно делать во втором режиме, когда на поле стоит знак '?'?

В этом режиме нужно (не только нужно) заменить знак '?' на «лису» или число. А вот число может и не соответствовать количеству видимых лис. Диагностика ошибки в этом режиме невозможна, поле-то Ваше.

Будет ли удаляться temp-файл, создаваемый моей программой во время работы, по ее окончании или же после окончания работы арбитра?

Временный файл гарантировано храниться до окончания работы арбитра. После этого временный файл может быть удалён.

Всегда ли программа начинает играть с «чистого» поля или могут быть тесты, в которых заранее подготовлена некоторая стартовая конфигурация?

Всегда с «чистого».

16.120. Тесты и рекорды:

N	Вход		Рекорд соревнования
0	M=4	N=5	4
1	M=3	N=3	3
2	M=9	N=5	6
3	M=10	N=10	7
4	M=99	N=100	37
5	M=2	N=100	4
Время работы программы 30 секунд			

16.122. ЧаВО по задаче 16.122:

Наверное, в примере <Количество стоимостей> должно быть 8, а не 5?

Конечно должно быть 8, это опечатка. В тексте уже исправлено.

Что будет оценивать жюри в решении данной задачи, будут ли учитываться такие вещи как проверка ошибок в cost.txt и т.д. и т.п. А также за что будут снимать баллы, т.е. чего в программе не должно быть?

Главное, найти путь как можно дешевле. Проверку ошибок мы учитывать не будем. Снимать баллы будем за несоблюдение технических условий.

Если не ко всем города можно будет проехать, что делать?

В условии задачи ничего про это не сказано. Поэтому форма сообщения об этой невозможности оставлена на Ваше усмотрение.

Верно ли, что любые два города соединены не более чем одной дорогой?

Нет. Об этом не было сказано в условии. Значит нужно поступать как в реальной жизни, т.е. дорог может быть несколько.

При выводе городов их обязательно указывать по тому порядку, по которому ехал коммивояжер?

Да. Это подразумевается в условии задачи. Иначе это не путь.

16.124. ЧаВО по задаче 16.124:

Существуют ли ограничения на размер входных данных (количество отрезков)?

Ограничение на размер платы можно считать равным 100x100.

Что такое нижний конец у горизонтального отрезка?

Слово «нижний» — опечатка, должно быть «левый».

Каково максимальное время на один тест?

Можете считать, что 30 секунд.

При совпадении плат расстояния между контактными площадками не учитываются (как и в примере)?

Расстояния не учитываются, важен факт соединения.

Если на второй плате окажется лишней (недостающей) одна или несколько площадок или компонентов — считать ли это за несовпадение? И еще не все ли равно как сравнивать платы — первую со второй, или наоборот вторую с первой?

а) по условию вторая плата проверяется на предмет возможности замены ею первой,

б) по техническим условиям количество групп площадок всегда ОДНО,

в) наличие «лишней» площадки в группе, имеющей соединения с другими площадками где-либо, по условию означает, что вывод элемента, соединяемый с этой площадкой, получает соединение, которого быть не должно,

г) если «лишняя» площадка не имеет соединений с другими площадками, то по условию ее можно не считать: она не дает соединений,

д) таким образом, можно считать что каждый электронный элемент имеет ровно четыре вывода, которые соединяются с четырьмя площадками, некоторые из этих площадок могут не иметь соединений с другими площадками вообще, в этом случае площадка может быть никак не обозначена в данных: к ней может не подходить проводник, какие выводы соединяются с какими площадками — не важно, главное — соединить каждый вывод с отдельной площадкой в группе (содержащей 4 площадки и «расположенной» в одной «точке» с целочисленными координатами).

16.126. Тесты:

N	Вход		Верный ответ
0	2	10	160 005
1	2	11	160 006
2	20	10	1 599 591
3	1999	1999	303 711 985
4	1	10	10
5	40000	40000	79 999
6	30000	20000	1 200 049 999
7	20001	20000	1 600 000 000
8	32100	12345	1 014 428 154
Время работы программы 50 секунд			

16.127.

```
############
# * * * * *#
#* * * * * #
# * * * * *#
#* * * * * #
# * # * * *#   4
#* * * * * #
# * * * * *#
#* D * * * #
#S* * * * *#
############
############

############
#**********#
#**********#
#**********#
#**********#
#***DS*****#
#**********#   10
#**********#
#**********#
#**********#
#**********#
############
```

```
############
#D        *#
# ######## #
# #S#* #*# #
# # #    # #
# # * #  # #  4
# #####  # #
# #   #  # #
# #*#   ## #
# #######* #
#*        *#
############

############
#D      ***#
# ########*#
# #***   #*#
# # * ***# #
# # * ** # #   4
# #** ***# #
# #*   * # #
#*#*** * # #
#*######### #
#***      S#
############

############
#D         #
# ###  *** #
#  #   *   #
#  #   **  #
#      *   #
# ###  *** #   0
# #   ###  #
# #    #   #
# ###  # ###
#        #S#
############
```

16.128.

```
3
3 5 7
1 1 0
2 4 0

4
2 5 11 13
1 3 7 11
0 1 5 6

10
32003 31873 31883 31891 31847 31849 31859 31793 31799 31817
30000 30000 30000 30000 30000 30000 30000 30000 30000 30000
30000 30000 30000 30000 30000 30000 30000 30000 30000 30000

10
32003 31873 31883 31891 31847 31849 31859 31793 31799 31817
31003 30873 30883 30891 30847 30849 30859 30793 30799 30817
31002 30872 30882 30890 30846 30848 30858 30792 30798 30816

10
26881 26891 26893 26903 26921 26927 26947 26951 26953 26959
20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
25000 25001 25002 25003 25004 25005 25006 25007 25008 25009


10
2 26891 26893 26903 26921 26927 26947 26751 26953 26959
0 20000 19999 20000 20000 20000 20000 20000 19998 20000
0 25001 25002 25003 25004 25005 25006 25000 25008 5000
```

16.129. ЧаВО по задаче 16.129:

Что подается на вход программе?

Программа должна читать файл filelist «...полученный командой dir>filelist...». И ни к каким другим файлам и каталогам (кроме лично созданных программой) программа обращаться не должна.

Что должна сделать программа, если существует файл размером больше 1300Кб?

Нужно диагностировать ошибку.

Могут ли быть в списке вложенные каталоги?

Команда dir>filelist будет выполняться в каталоге без подкаталогов. Значит в filelist каталогов не будет, кроме стандартных.

Каковы ограничения на имена файлов?

Это не принципиально. Будем считать, что длина имени до 8 символов, расширения — до 3, символы в названии файла будут только латинскими буквами и арабскими цифрами.

Команда copy не копирует файлы с размером 0. Что с ними делать?

Таких файлов не будет.

16.130. ЧаВО по задаче 16.130:

Обозначьте приоритеты.
Что лучше: больше уплотнить или сделать меньше перемещений?

В порядке убывания (как записано в формулировке задачи):

а) не делать ошибок, в том числе, на больших количествах архивов;
б) использовать как можно меньше носителей (это единственный смысл "плотности хранения", который можно найти в формулировке задачи);
в) сделать как можно меньше перемещений (перемещением считается вся процедура переноса одного архива из начального положения в конечное, никакие промежуточные операции не считаются).

16.131. Описания и тесты.

Описания:

N	Описание сути теста
0	пример из условия
1	обязательный простой общий пример
2	"спираль"
3	один треугольник
4	пусто
5	предельная вложенность
6	предельный "паркет"
7	предельные "зигзаги"
8	"двуугольник" — ошибка
9	"одноугольник" — ошибка
Время работы программы 20 секунд	

Тесты (многоточия восстановите сами):

N	0	1	2	3	4	5	6	7	8	9
	3	6	3	1	0	50	100	4	2	2
	4	3	4	3		4	4	98	3	3
	0 0	2 2	3 3	0 0		0 0	0 0	0 0	0 0	0 0
	0 5	2 3	3 4	0 1		0 99	0 1	1 92	0 6	0 6
	5 5	3 2	4 4	1 0		99 99	49 1	2 0	6 0	6 0
	5 0	5	4 3			99 0	49 0	3 92	2	1
	4	0 2	4			4	4	4 0	1 1	1 1
	1 1	4 0	6 3			1 1	0 2	5 92	2 2	
	1 4	8 2	6 4			1 98	0 3	6 0		
	4 4	8 8	7 4			98 98	49 3	7 92		
	4 1	3 8	7 3			98 1	49 2	8 0		
	4	7	16			4	4	9 92		
	2 2	7 7	0 0			2 2	0 4	10 0		
	2 3	7 3	0 8			2 97	0 5	11 92		
	3 3	5 3	11 8			97 97	49 5	12 0		
	3 2	6 5	11 0			97 2	49 4	13 92		
		5 6	10 0			4	4	14 0		
		3 5	10 7			3 3	0 6	15 92		
		3 7	1 7			3 96	0 7	16 0		
		3	1 1			96 96	49 7	17 92		
		4 4	8 1			96 3	49 6	18 0		
		3 4	8 5			4	4	19 92		
		4 2	5 5			4 4	0 8	20 0		
		4	5 2			4 95	0 9	21 92		
		5 5	2 2			95 95	49 9	22 0		
		4 1	2 6			95 4	49 8	23 92		
		1 2	9 6			4	4	24 0		
		2 4	9 0			5 5	0 10	25 92		
		6				...	...	...		
		10 10				4	4	11 98		
		10 2				48 48	50 96	10 6		
		6 0				48 51	50 97	9 98		
		9 2				51 51	99 97	8 6		
		9 9				51 48	99 96	7 98		
		0 9				4	4	6 6		
						49 49	50 98	5 98		
						49 50	50 99	4 6		
						50 50	99 99	3 98		
						50 49	99 98	2 6		

16.132. ЧаВО по задаче 16.132:

У меня возник вопрос по поводу фрмулировки задачи: «Начальное положение ящиков может выбираться жюри» и «Начальное положение ящиков на поле будет всегда такое, как показано в примере, т.е. играть будут четыре игрока и их ящики будут стоять по углам»
Как же будут стоять ящики?

Вступление описывает общий вариант игры. Можно играть разным количеством и с разными начальными положениями. А техническое задание описывает ту ситуацию, для которой нужно писать программу, т.е. «играть будут четыре игрока и их ящики будут стоять по углам».

Нас интересует, можно ли для решения создавать дополнительные файлы?

Можно. Но суммарным объёмом не более 1Мб. Файлы нужно располагать в текущем каталоге. Они будут хранится в течение всей игры. Для других игр сохранность файлов не гарантируется.

16.133. ЧаВО по задаче 16.133:

Какое максимальное количество правил может содержать файл rules.txt?

Значение не ограничено (это предмет состязания, см. условие задачи).

Какое максимальное значение может принимать N?

Значение не ограничено (это предмет состязания, см. условие задачи).

На какой объем памяти я могу рассчитывать при использовании BP70 в Real Mode и Protected Mode?

Решение будет тестироваться на машине с объёмом оперативной памяти 64M, виртуальной — 256M.

Есть ли ограничения на внешние файлы?

Объём файлов должен позволять их запись и обработку за 10 секунд.

16.135. Тесты:

N	Вход		Рекорд кантований
0	1	3	3
1	3	1	3
2	15	19	8
3	1	1	2
4	1	9	3
5	39	9	11
6	9	9	4
7	7	5	6
8	25	1	3
Время работы программы 30 секунд			

16.136. ЧаВО по задаче 16.136:

Что должна делать программа, если расположение тюков таково, что расставить ОКО так, чтобы всё свободное пространство склада находилось под контролем, невозможно?

Если нельзя взять под охрану всё свободное место, то нужно охранять хотя бы максимально возможное.

Какие решения будут оцениваться выше?

Если возможно охранять всё свободное место, то выше оцениваются решения с меньшим количеством ОКО. И только если охранять всё невозможно, то выше оцениваются программы, охраняющие большую площадь, а среди них, использующие меньше ОКО.

16.137. ЧаВО по задаче 16.137:

Какими ресурсами располагает программа (в т.ч. временнЫми)?

1Мбайт, 50 секунд.

Какова максимальная координата пера по осям x и y?

Как и координаты абсолютного перемещения — 4 десятичные цифры.

Сколько всего может быть команд в файле input.gpc?

Сколько сможете обработать.

В каком положении изначально находится перо?

Можно считать, что поднято.

Что считать правильным образом отрезка прямой?

Правильным образом отрезка прямой считается кратчайшая неразрывная последовательность пикселов (содержащая меньше всего пикселов), центры которых находятся на минимальных расстояниях до этого отрезка прямой.

Последнее можно уточнить так:
если пикселы расположить в порядке уменьшения (неувеличения) рас-

стояния до данного отрезка, то

1) первый пиксел в этой последовательности (самый удалённый) должен быть расположен на наименьшем из возможных расстояний до этого отрезка,
2) следующий пиксел должен быть также расположен на наименьшем из возможных для него расстояний до отрезка, и т. д.

Неразрывность понимается как смежность любых двух последовательных пикселов (по горизонтали, по вертикали или по диагонали).

Пикселы начала и конца отрезка должны входить в последовательность.

Примеры ошибок в отрезке `PA0,0;PD;PR+4,+2;` (заголовок, кавычки, запятые и т. п. для краткости опущены):

а)
```
abbbb
bbabb
bbbba
```
— есть разрывы,

б)
```
abbbb
babbb
bbaaa
```
— один из пикселов слишком далеко.

в)
```
aabbb
baaab
bbbaa
```
— слишком много пикселов.

Примеры правильных решений:

```
aabbb
bbabb
bbbaa,

abbbb
baaab
bbbba,

aabbb
bbaab
bbbba,

abbbb
baabb
bbbaa.
```

Верно ли я понимаю, что при абсолютном передвижении и опущенном пере рисуется отрезок?

Да.

Останется ли точка после таких команд: ...PD; PU;...?

Да.

Должна ли программа обрабатывать такие команды: "PD ;", "PA 2,3;", "PR2,-3;", "PA00045,0;", "PD PR-2,-4;" и им подобные?

`"PD ;"`: ошибка — пробел.

`"PA 2,3;"`: ошибка — пробел.

`"PR2,-3;"`: ошибка — нет "+".

`"PA00045,0;"`: ошибка — в числе больше 4 цифр.

`"PD PR-2,-4;"`: ошибка — нет ";" после PD.

Сообщения об ошибках могут оцениваться только в последнюю очередь и только при прочих равных условиях.

Можно ли использовать графические библиотеки? В частности, Graph в TP или gdi32.dll в Windows т.е. в Delphi...

Никакие готовые программы графических преобразований использовать, разумеется, нельзя.

16.149.

Следующая таблица содержит описания тестов, которые надо записать в файл.

N	Вход	Выход
0	5 20 13 23 12 18 6 14 3 17 11 21 2 1 4 25 7 15 5 16 10 22 8 19 9 24	23
1	5 1 2 3 4 5 16 17 18 19 6 15 24 25 20 7 14 23 22 21 8 13 12 11 10 9	46
2	3 1 2 1 2 1 2 1 2 1	3

3	$\begin{matrix} 1 \\ 10 \end{matrix}$	10
4	$\begin{matrix} 9 \\ 1\ 6\ 6\ 6\ 5\ 7\ 2\ 7\ 1 \\ 7\ 7\ 7\ 7\ 6\ 6\ 1\ 6\ 6 \\ 6\ 6\ 6\ 6\ 5\ 7\ 2\ 9\ 1 \\ 7\ 7\ 7\ 7\ 4\ 2\ 1\ 6\ 6 \\ 6\ 6\ 6\ 5\ 9\ 3\ 7\ 7\ 7 \\ 7\ 7\ 7\ 7\ 1\ 6\ 6\ 6\ 6 \\ 6\ 6\ 6\ 6\ 1\ 7\ 7\ 7\ 7 \\ 7\ 7\ 7\ 7\ 2\ 6\ 6\ 6\ 6 \\ 1\ 9\ 9\ 9\ 9\ 9\ 9\ 9\ 1 \end{matrix}$	20
5	$\begin{matrix} 99 \\ 9\ 9\ \ldots\ 9\ 9\ 9\ 9\ 9\ 9 \\ 1\ 1\ \ldots\ 1\ 1\ 1\ 1\ 1\ 1 \\ 9\ 9\ \ldots\ 9\ 1\ 9\ 9\ 9\ 9 \\ 39 \left\{ \begin{matrix} \vdots\ \ \vdots\ \ \ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots \end{matrix} \right. \\ 9\ 9\ \ldots\ 9\ 1\ 9\ 9\ 9\ 9 \\ 9\ 9\ \ldots\ 9\ 1\ 1\ 1\ 1\ 1 \\ 9\ 9\ \ldots\ 9\ 1\ 9\ 9\ 9\ 9 \\ \vdots\ \ \vdots\ \ \ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots \\ 9\ 9\ \ldots\ 9\ 1\ 9\ 9\ 9\ 9 \end{matrix}$	402
6	$\begin{matrix} 51 \\ 101\ \ 101\ \ \ldots\ \ 101 \\ \vdots\ \ \ \ \vdots\ \ \ \ \ \ \vdots \\ 101\ \ 101\ \ \ldots\ \ 101 \end{matrix}$	2626
Время работы программы 50 секунд		

16.183.

N	Вход	Выход
0	2	1
1	7	6
2	74	40
3	127	126
4	9999	9126
5	500500001	500500000
6	2000000000	1999743786
7	2146828126	2146828125
Время работы программы 3 секунды		

16.184.

N	Вход	Выход
0	9112004	9117004
1	198273645	698273645
2	1234567890123456789	1234567890123456789
3	1	11
4	988888888877777777	088888888877777777
5	987654888888	
6	$\underbrace{99\ldots9}_{19}0$	$\underbrace{99\ldots9}_{20}$
7	999555	Нет возможности
8	22222455	Нет возможности
9	7777766666	7777266666
Время работы программы 3 секунды		

16.185.

N	Входные данные	Верный ответ
0	10	1111
1	100	8031
2	2004	9410
3	12345	911
4	98765	9910
5	1	220
6	99999	10
Время работы программы 3 секунды		

16.186.

N	Входные данные	Верный ответ
0	строчка	a
1	1234567890	9
2	X	X
3	QWERTY	E
4	12345678901234567890123456789O	9
5	qaz	q
6	Пример	и
Время работы программы 3 секунды		

Литература

[1] Абрамов С. А., Гнездилова Г. Г., Капустина Е. Н., Селюн М. И. Задачи по программированию. — М.: Наука, 1988. — 224 с.

[2] Абрамов С. А., Зима Е. В. Начала информатики. — М.: Наука, 1989. — 256 с.

[3] Брукс Ф. П. Мифический человеко-месяц, или как создаются программные системы. СПб: Символ-Плюс, 1999.

[4] Воробьев Н. Н. Числа Фибоначчи. М.: Наука, 1978., 144 с.

[5] Непейвода Н. Н., Скопин И. Н. Основания программирования. Москва-Ижевск, РХД, 2003, 880 с.

[6] Непейвода Н. Н. Стили и методы программирования. М.: ИНТУИТ.РУ «Интернет-Университет Информационных Технологий», 2005 — 320 с.

[7] Пильщиков В. Н. Сборник задач и упражнений по языку Паскаль: Учебное пособие для вузов. — М.: Наука, 1989. — 160 с.

[8] Котарова И. Н., Зубов В. С., Архипов О. Г. и др. Сборник задач по базовой компьютерной подготовке. М.: Изд-во МЭИ, 1998. — 178 с.

[9] Пупышев В. В. 50 задач по началам программирования. — Ижевск: Изд. дом «Удмуртский университет», 1999. — 60 с.

[10] Купчинаус С. Ю., Анисимов А. Е. Программирование. Введение в алгоритмизацию задач и программирование на языке Паскаль: Учеб. пособие. – Ижевск: Изд-во Удм. ун-та/ВКУПП, 1994/1997. – 80 с.

[11] Скопин И. Н. Основы менеджмента программных проектов. Курс лекций. Учебное пособие. — М.: ИНТУИТ.РУ «Интернет-Университет Информационных Технологий», 2004 — 336 с.

[12] Кушниренко А. Г., Лебедев Г. В. Программирование для математиков: Учеб. пособие для вузов. — М.: Наука, 1988. – 384 с.

[13] Райли Д. Абстракция и структуры данных: Вводный курс: Пер. с англ. — М.: Мир, 1993. – 752 с.

[14] Павловская Т. А. C/C++. Программирование на языке высокого уровня. — СПб.: Питер, 2001. — 464 с.

[15] Нивергельт Ю., Фаррар Дж., Рейнголд Э. Машинный подход к решению математических задач. — М.: Мир, 1977. — 322 с.

[16] Косневски Ч. Занимательная математика и персональный компьютер. — Пер. с англ. — М.: Мир, 1987. — 192 с.
[17] Уилсон Р. Введение в теорию графов. — М.: Мир, 1977. — 208 с.
[18] Уэзерелл Ч. Этюды для программистов. Пер с англ. — М.: Мир, 1982. — 228 с.
[19] Программирование на языке Паскаль: задачник / под ред. Усковой О. Ф. — Спб.: Питер, 2002. — 336 с.: ил.
[20] Юркин А. Г. Задачник по программированию. — СПб.: Питер, 2002. — 192 с.
[21] Петров Ю. А. Логические проблемы абстракций бесконечности и осуществимости. — М.: Наука, 1967 — 164 с.

Анисимов Андрей Евгеньевич
Пупышев Вячеслав Викторович

Сборник заданий по основаниям программирования