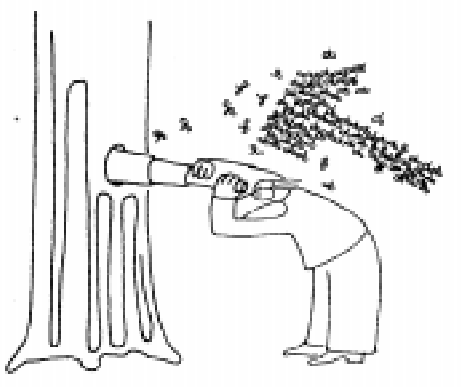
Елизаров Роман Анатольевич

РАЗБОР ЗАДАЧ ФИНАЛЬНОГО ТУРА 23-го ЕЖЕГОДНОГО МЕЖДУНАРОДНОГО СТУДЕНЧЕСКОГО ЧЕМПИОНАТА МИРА ПО ПРОГРАММИРОВАНИЮ АСМ



Задача: ***Разведение Пчел (Bee Breeding)***

Профессор Б. Хеиф проводит эксперименты с видом Южноамериканских пчел, которых он нашел во время экспедиции в Бразильские леса. Мед, производимый этими пчелами, превосходного качества по сравнению с медом Европейских и Североамериканских пчел. К несчастью, пчелы плохо размножаются в неволе. Профессор Хеиф думает, что причина в том, что расположение личинок (рабочих пчел, матки и так далее) внутри сот зависит от окружающей среды, которая различна в лаборатории и в лесу.

В качестве первого шага для проверки своей теории, профессор Хеиф хочет измерить разницу в положении личинок. Для этого он измеряет расстояние между ячейками сот, в которые помещены личинки. С этой целью профессор пометил ячейки, обозначив произвольную ячейку номером 1 и помечая оставшиеся ячейки по часовой стрелке, как показано на рисунке 1.

Например, две личинки в ячейках 19 и 30 находятся на расстоянии 5 ячеек друг от друга. Один из кратчайших путей, соединяющий эти две ячейки, проходит через ячейки 19 - 7 - 6 - 5 - 15 - 30, так что вы должны передвинуться в соседние ячейки пять раз, чтобы попасть из ячейки с номером 19 в ячейку с номером 30.

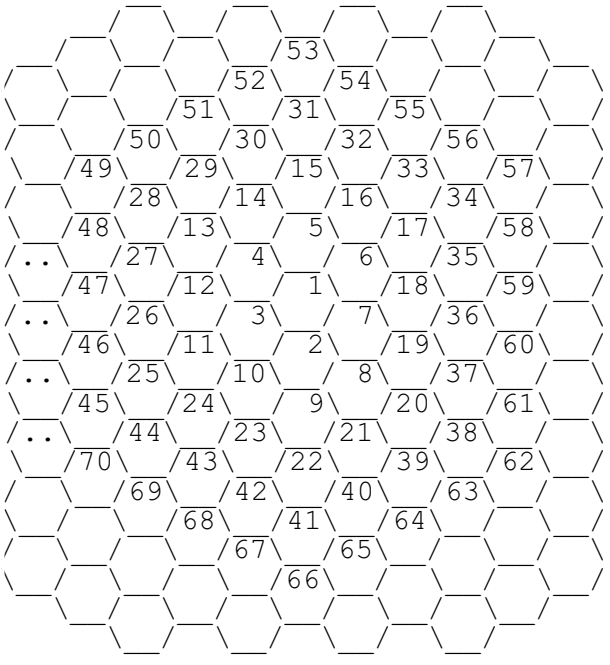


Рисунок 1.

Профессор Хеиф нуждается в вашей помощи в написании программы, которая посчитает расстояние, определенное как количество ячеек в кратчайшем пути для любой пары ячеек.

Входные данные

Входные данные состоят из нескольких строк, каждая из которых содержит два целых числа a и b ($a, b \le 10000$), обозначающих номера ячеек. Эти числа всегда положительны, за исключением последней строки, которая содержит $a=b=0$. Эта строка завершает ввод и не должна обрабатываться.

Выходные данные

Для каждой пары чисел (*a, b*) во входном файле выведите расстояние между ячейками с номерами *a* и *b* на отдельной строке. Расстояние - это минимальное число перемещений, необходимое для того, чтобы добраться из *a* в *b*.

Пример входных данных	*Пример соответствующего выхода*
19 30	Расстояние между 19 и 30 равно 5
0 0	

Решение

Особая прелесть данной задачи заключается в том, что ее можно решать множеством различных способов. Проще всего ввести в гексагональной структуре ячеек систему координат (*x, y*) после чего задача разобьется на две более простые: преобразование номера ячейки в пару координат и нахождение расстояния между ячейками по паре координат.

Ввести систему координат можно, например, следующим образом:

a) Припишем ячейке под номером один координаты (0, 0).

b) Будем считать, что сверху от ячейки с координатами (*x, y*) находится ячейка с координатам (*x, y*+1), а снизу ячейка с координатами (*x, y*-1).

c) Будем считать, что справа снизу от ячейки с координатами (*x, y*) находится ячейка с координатами (*x*+1, *y*), а слева сверху ячейка с координатами (*x*-1, *y*).

Читатель может убедиться в том, что перечисленных выше правил достаточно, чтобы каждой ячейке однозначно присвоить пару целочисленных координат (*x, y*). Ячейка с номером 19, например, получит координаты (2, 0), а ячейка с номером 30 – (-1, 2).

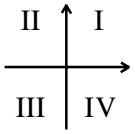
Следующим шагом нужно составить алгоритм нахождения координаты произвольной ячейки по ее номеру. Для этого необходимо обратить внимание на процесс последовательной нумерации ячеек. Будем обозначать переходы вверх и вниз буквами В и Н, соответственно, а переходы влево вверх, влево вниз, вправо вверх и вправо вниз как ЛВ, ЛН, ПВ и ПН, соответственно. Начнем с ячейки номер 2 и заметим, сколько раз и в каких направлениях происходят перемещения при последовательной нумерации ячеек: 1ЛВ, 1В, 1ПВ, 1ПН, 2Н, 1ЛН, 2ЛВ, 2В, 2ПВ, 2ПН, 3Н, 2ЛН, 3ЛВ, 3В, 3ПВ, 3ПН, 4Н, 3ЛН и так далее. В этой последовательности легко заметить закономерность, позволяющую написать алгоритм, который промоделирует процесс нумерации ячеек. Учитывая небольшие ограничения на номера ячеек (по условию задачи не более 10000), можно завести два массива целых чисел X и Y с целыми индексами от 1 до 10000, которые следует заполнить координатами ячеек, один раз промоделировав процесс нумерации в начале программы. Конечно, это не самый эффективный по времени и расходуемой памяти алгоритм. Любопытный читатель может вывести аналитическую формулу нахождения координаты ячейки по ее номеру, но вывод такой формулы достаточно сложен и подвержен ошибкам. На реальном соревновании лучше решить этот вопрос «в лоб» – программным путем, описанным выше.

В заключение необходимо лишь найти расстояние между ячейками, зная их координаты, что уже не сложно. Надо лишь помнить, что дело происходит не в обычной прямоугольной решетке, в которой наикратчайшее расстояние между двумя ячейками, измеряемое наименьшим количеством ячеек в цепочке, соединяющей их, выражалось бы формулой $|x2 - x1| + |y2 - y1|$, а в гексагональной решетке. Здесь у каждой ячейки с координатами (*x, y*), кроме «обычных» соседей с координатами (*x*+1, *y*), (*x*-1, *y*), (*x*, *y*+1) и (*x, y*-1), есть еще два соседа (общее число соседей ячейки равно шести) с координатами (*x*+1, *y*+1) и (*x*-1, *y*-1). Мы выведем эту формулу пользуясь формулой для прямоугольной решетки как аналогией и следующими соображениями:

a) Для расстояния важны только относительные координаты $\Delta x = x2 - x1$ и $\Delta y = y2 - y1$.

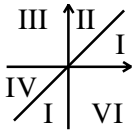
b) Для «прямоугольной» формулы $|\Delta x| + |\Delta y|$, если снять модули, получим четыре варианта вычислений (по одному для каждой четверти плоскости):

I) $\Delta x + \Delta y$, если $\Delta x > 0$ и $\Delta y > 0$,
II) $-\Delta x + \Delta y$, если $\Delta x < 0$ и $\Delta y > 0$,
III) $-\Delta x - \Delta y$, если $\Delta x < 0$ и $\Delta y < 0$,
IV) $\Delta x - \Delta y$, если $\Delta x > 0$ и $\Delta y < 0$.



c) Для «гексагональной» формулы будет существовать шесть вариантов расчета, для различных частей плоскости. Эти формулы легко представить «в уме» и выписать:

I) Δx, если $\Delta x > 0$, $\Delta y > 0$ и $\Delta x > \Delta y$,
II) Δy, если $\Delta x > 0$, $\Delta y > 0$ и $\Delta x < \Delta y$,
III) $-\Delta x + \Delta y$, если $\Delta x < 0$ и $\Delta y > 0$,
IV) $-\Delta x$, если $\Delta x < 0$, $\Delta y < 0$ и $\Delta x < \Delta y$,
V) $-\Delta y$, если $\Delta x < 0$, $\Delta y < 0$ и $\Delta x > \Delta y$,
VI) $\Delta x - \Delta y$, если $\Delta x > 0$ и $\Delta y < 0$.



d) Эти формулы можно записать в программе прямо в таком виде, а можно объединить в формулу с тремя модулями, до которой нетрудно догадаться, посмотрев на приведенные выше случаи. Итак, искомое расстояние будет равно $(|\Delta x| + |\Delta y| + |\Delta x - \Delta y|)/2$.

Текст программы

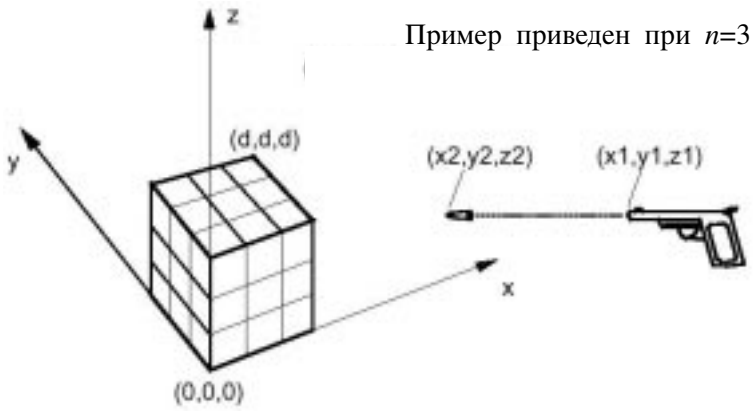
```
{ Решение задачи Разведение Пчел (Bee Breeding) }
{ Компилятор: Borland Pascal 7.0                 }
{ (C) Роман Елизаров, 1999                       }
program WATER;
const
     MAX = 10000; { Максимальное значение для a и b из условия задачи }

     { Законы нумерации }
     { ns - количество перемещений (относительно базового числа) }
     { nx - смещение по x }
     { by - смещение по y }
     ns: array[0..5] of integer = ( 0,  0,  0,  0,  1,  0);
     nx: array[0..5] of integer = (-1,  0,  1,  1,  0, -1);
     ny: array[0..5] of integer = ( 0,  1,  1,  0, -1, -1);

var
     a, b: integer;      { Значение a и b }
     n: integer;         { Номер клетки }
     s: integer;         { Количество шагов при нумерации }
     dx, dy: integer;    { Текущие координаты }
     i, j: integer;      { Временные переменные }
     x, y: array [1..MAX] of integer; { Массив координат }

begin
     { Подготаваливаем координаты }
     x[1]:= 0; y[1]:= 0; { Координаты первой клетки }
     n:= 2;              { Начнем с клетки 2 }
     dx:= 0; dy:= -1;    { ... с координатами (0, -1) }
     s:= 1;              { Начальные шаг равен 1 }
     while n <= MAX do begin
          for i:= 0 to 5 do
               for j:= 1 to s + ns[i] do begin
                    if n > MAX then break;
                    x[n]:= dx;
                    y[n]:= dy;
                    inc(n);
                    inc(dx, nx[i]);
                    inc(dy, ny[i]);
               end;
          { Увеличиваем шаг }
          inc(s);
     end;
```

```
    { Главный цикл по всем данным }
    while not SeekEof do begin
        { Читаем входные данные (a и b) }
        read(a, b);
        if (a = 0) or (b = 0) then exit;
        { Считаем и выводим ответ }
        dx:= x[b] - x[a];
        dy:= y[b] - y[a];
        writeln('Расстояние между ', a, ' и ', b, ' равно ',
                (abs(dx) + abs(dy) + abs(dx-dy)) div 2);
    end;
end.
```

Задача: ***Пулевое отверстие (Bullet Hole)***

Куб подвешен в пространстве (рисунок 2). Декартова система координат определена так, что ее начало находится в одном из нижних углов куба, как показано на рисунке. Длина ребра куба – d, таким образом, его противоположные углы имеют координаты (0, 0, 0) и (d, d, d). Положительное направление оси z координатной системы направлено «вверх» по отношению к гравитации.

Внутренность куба содержит отделения с одинаковым разбиением в каждом направлении таким образом, что куб разбит на n^3 мини-кубиков одинакового размера. Отделения тонкие и водонепроницаемые, и каждый мини-куб наполнен водой. Общий объем воды во всех мини-кубах равен d^3.



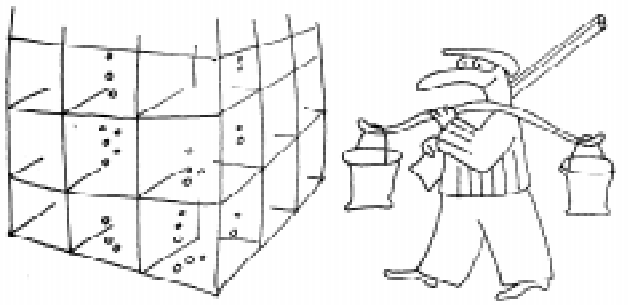
Пример приведен при n=3

Рисунок 2.

Из пистолета вылетает пуля, которая может попасть в куб. Дуло пистолета находится в точке (x_1, y_1, z_1). Точка (x_2, y_2, z_2) на пути полета пули задает направление. Пуля не разбивает куб, но там, где проходит, она делает маленькую дырку. Дырки могут быть сделаны на гранях, ребрах или в вершинах мини-кубов. Вода под воздействием гравитации вытекает через эти маленькие дырки. Вся вода, которая вытекает из большого куба, собирается, и измеряется ее объем.

Ваша программа должна посчитать общий объем воды, которая вытекает из большого куба.

Входные данные



Входные данные содержат несколько испытаний. Каждое испытание задается восьмью целыми числами. Первое число n ($n \le 50$) описано выше. Второе число d ($d \le 100$). Оставшиеся шесть чисел – x_1, y_1, z_1, x_2, y_2, z_2 – представляют начало и точку на пути пули ($-100 \le x_1, y_1, z_1, x_2, y_2, z_2 \le 100$). Начало и точка на пути не совпадают. После последнего испытания, целое число 0 завершает набор входных данных.

Выходные данные

Для каждого испытания, напечатайте номер испытания и общее число вытекшей воды с точностью до двух знаков после запятой. Выведите пустую строку между испытаниями.

Замечание

В данной задаче два вещественных числа считаются равными, если их разность меньше 10^{-6}.

Пример входных данных	*Пример соответствующего выхода*
5 25 5 15 0 5 15 100	Испытание 1, Объем = 2500.00
3 30 0 -35 0 3 -25 3	Испытание 2, Объем = 1950.00
10 16 8 17 11 12 19 6	Испытание 3, Объем = 0.00
0	

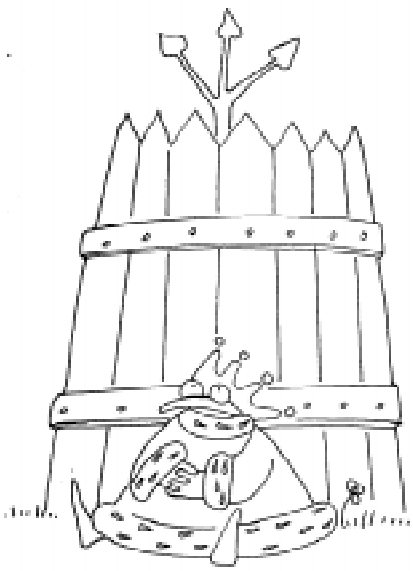
Решение

Читателю предлагается самому решить эту задачу.

Задача: ***Укрепленный лес (The Fortified Forest)***

В далекой стране жил-был король. У короля была маленькая коллекция редких и ценных деревьев, которые были собраны его предшественниками во время их путешествий. Чтобы защитить деревья от воров, король приказал построить вокруг них высокий забор. Дело было поручено волшебнику.

Увы! Волшебник очень быстро обнаружил, что единственный доступный и подходящий для постройки забора материал - это древесина самих деревьев. Другими словами, необходимо срубить некоторые деревья, чтобы построить забор вокруг остальных. Конечно, чтобы предотвратить срубание своей собственной головы, волшебник хотел минимизировать ценность деревьев, которые должны быть срублены. Волшебник ушел в свою башню и пробыл там до тех пор, пока не нашел лучшее решение данной задачи. Забор был построен, и с тех пор все жили счастливо.



Вы должны написать программу, решающую задачу, с которой столкнулся волшебник.

Входные данные

Входные данные состоят из нескольких заданий, каждое из которых описывает гипотетический лес. Каждое задание начинается со строки содержащей единственное число n ($2 \le n \le 15$), количество деревьев в лесу. Деревья идентифицируются последовательными числами от 1 до n. Каждая из последующих n строк содержит четыре целых числа x_i, y_i, v_i, l_i, которые описывают одно дерево. Пара (x_i, y_i) задает положение дерева на плоскости, v_i это его ценность, а l_i - это длина забора, который может быть построен из древесины данного дерева; v_i и l_i лежат в диапазоне от 0 до 10000.

Входные данные заканчиваются пустым заданием (n = 0).

Выходные данные

Для каждого задания вычислите подмножество деревьев таких, что, используя древесину из этого подмножества, оставшиеся деревья могут быть окружены одним общим забором. Найдите подмножество с минимальной суммарной ценностью деревь-

ев. Если существует более одного множества с минимальной ценностью, то выберете то, которое содержит наименьшее число деревьев. Для простоты считайте, что деревья имеют нулевой диаметр.

Выведите, как показано ниже, номер задания (1, 2, ...), идентификатор каждого дерева, которое нужно срубить, и лишнюю длину забора (с точностью до двух знаков после запятой).

Выведите пустую строку между заданиями.

Пример входных данных	*Пример соответствующего выхода*
6	Лес 1
0 0 8 3	Срубить деревья: 2 4 5
1 4 3 2	Лишняя древесина: 3.16
2 1 7 1	
4 1 2 3	Лес 2
3 5 4 6	Срубить деревья: 2
2 3 9 8	Лишняя древесина: 15.00
3	
3 0 10 2	
5 5 20 25	
7 -3 30 32	
0	

Решение

Для решения данной задачи не требуется ничего, кроме знания двух простых алгоритмов: генерация всех подмножеств данного множества и поиск выпуклой оболочки множества точек на плоскости.

Ограничение $n = 15$ позволяет решить эту задачу простым перебором всех нетривиальных подмножеств множества чисел от 1 до 15 (которых в худшем случае будет лишь 2^{15}-2=32766) . Для каждого подмножества необходимо любым известным способом найти выпуклую оболочку оставшихся точек и посчитать ее длину для определения допустимости данного решения. Из всех допустимых решений надо отобрать наилучшее с точки зрения условия задачи.

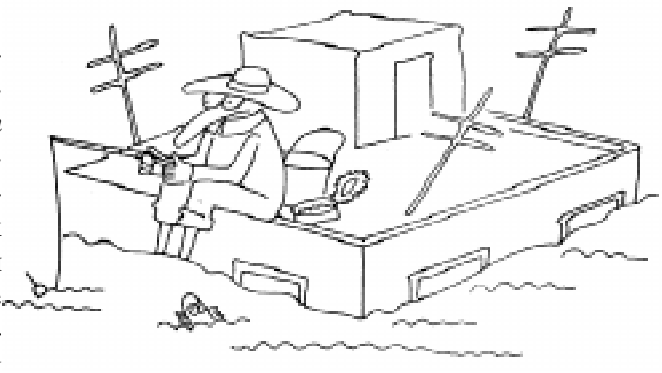
На компьютерном туре в силу заданных временных ограничений и быстродействия компьютеров жюри требовалось ввести в перебор некоторые отсечения и реализовать поиск выпуклой оболочки со сложностью не большей квадратичной. Только в таком случае программа могла пройти все тесты за требуемое время.

Задача: ***Затопленный! (Flooded!)***

Чтобы позволить покупателям домов определить стоимость страховки дома, фирма, которая продает недвижимость, предоставляет клиентам замеры высоты каждого квадрата 10 на 10 метров в регионах, где могут быть куплены дома. Вода от дождя, тающего снега и прорвавшихся водопроводов сперва собирается в областях с наименьшими высотами, так как вода будет стекать с более высоких областей. Для простоты мы предполагаем, что ливневые коллекторы позволят воде из высоко находящихся долин (полностью окруженных еще более высокими областями) стекать в более низкие квадраты и что вода не поглощается землей.

Их архивов мы знаем типичный объем воды, который собирается в регионе. Как будущие покупатели мы хотим узнать уровень воды после того, как она соберется в низко лежащих областях, и процентное отношение площади, которая полностью находится под водой (те есть процентное отношение 10-метровых квадратов, высота которых строго меньше уровня воды). Вы должны написать программу, которая посчитает эти величины.

Входные данные



Вход состоит из последовательности описания регионов. Каждый регион начинается с пары целых чисел *m* и *n*, каждое из которых меньше 30, задающих размеры региона в 10-метровых единицах. Далее следуют *m* строк из *n* целых чисел, задающих высоты квадратов. Высоты даны в метрах, где положительные и отрицательные значения представляют значения выше и ниже уровня моря, соответственно. Последнее значение в каждом регионе указывает количество кубических метров воды, которая соберется в регионе. После описания последнего региона следует пара нулей.

Выходные данные

Для каждого региона выведите его номер (1, 2, ...), уровень воды (в метрах, выше или ниже уровня моря) и процент площади региона, который находится под водой, на отдельной строке. Уровень воды и процент затопления должны быть выведены с точностью до двух знаков после запятой. Завершайте вывод каждого региона пустой строкой.

Пример входных данных	*Пример соответствующего выхода*
3 3	Регион 1
25 37 45	Уровень воды 46.67 метров.
51 12 34	66.67% региона под водой.
94 83 27	
10000	
0 0	

Решение

Для решения данной задачи необходимо отсортировать все регионы в порядке возрастания их высоты и смоделировать процесс затопления снизу вверх. При этом надо помнить о том, что квадраты имеют размер 10 на 10 метров, и каждый затопленный метр одного квадрата содержит 100 кубометров воды.

Сначала происходит затопление только одного квадрата до уровня следующего (по возрастанию высоты) квадрата. Потом затапливаются уже два квадрата до высоты третьего и так далее, пока не кончится вода.

Текст программы

```
{ Решение задачи Затопленный! (Flooded!) }
{ Компилятор: Borland Pascal 7.0         }
{ (C) Роман Елизаров, 1999               }
program WATER;
const
     MAX = 29; { Максимальные m и n из условия задачи }

var
     region: integer;   { Номер региона }
     m, n: integer;     { Значение m и n }
     v: longint;        { Общий объем воды }
     w: longint;        { Временная переменная }
     cnt: integer;      { Общее количество высот }
     i, j, t: integer;  { Временные переменные }
```

```
     h: array [0..MAX*MAX] of integer; { Массив высот }

begin
     region:= 1; { Инициализируем счетчик регионов }
     { Главный цикл по всем регионам }
     while not SeekEof do begin
          { Читаем входные данные (m и n) }
          read(m, n);
          if (m = 0) or (n = 0) then exit;
          { Читаем все высоты }
          cnt:= 0;
          for i:= 1 to m do
               for j:= 1 to n do begin
                    read(h[cnt]);
                    inc(cnt);
               end;
          { Читаем объем воды }
          read(v);
          { Сортируем высоты. При данных ограничениях, }
          { можем воспользоваться самым простым методом сортировки.}
          for i:= 0 to cnt - 2 do
               for j:= i + 1 to cnt - 1 do
                    if h[i] > h[j] then begin
                         t:= h[i];
                         h[i]:= h[j];
                         h[j]:= t;
                    end;
          { Установим ограничитель в массив высот }
          h[cnt]:= maxint;
          { Начинам процесс затопления }
          i:= 1; { Количество затапливаемых квадратов }
          { Затапливаем пока можем }
          while true do begin
               { Посчитаем сколько можно затопить }
               w:= longint(100) * i * (h[i] - h[i-1]);

               if v > w then
                    { Можно затопить все и еще останется вода }
                    dec(v, w)
               else
                    { Можно затопить только частично - конец! }
                    break;
               inc(i);
          end;
          { На выходе цикла: искомый уровень воды = h[i-1] + v/i/100}
          { Количество затапливаемых квадратов = i }
          { Выводим ответ }
          writeln("Регион ", region);
          writeln("Уровень воды ", (h[i-1] + v/i/100):0:2,"метров.");
          writeln(i/cnt*100:0:2, "% региона под водой.");
          inc(region);
     end;
end.
```

Елизаров Роман Анатольевич,
председатель жюри Северо-Восточного Европейского Региона Командного Чемпионата Мира по Программированию ACM.

НАШИ АВТОРЫ