Сборник задач по дисциплине «Алгоритмы и структуры данных»

Алексеевский П. И.
u@nyuu.ru

Екатеринбург, 2023

Оглавление

Задача №1. Лабиринт.

На прямоугольной сетке задан лабиринт, как показано на рисунке 1. Сетка однородна, шаг сетки принимается равным 1. Стены лабиринта лежат на линиях сетки. Также любая точка за пределами лабиринта считается стеной.

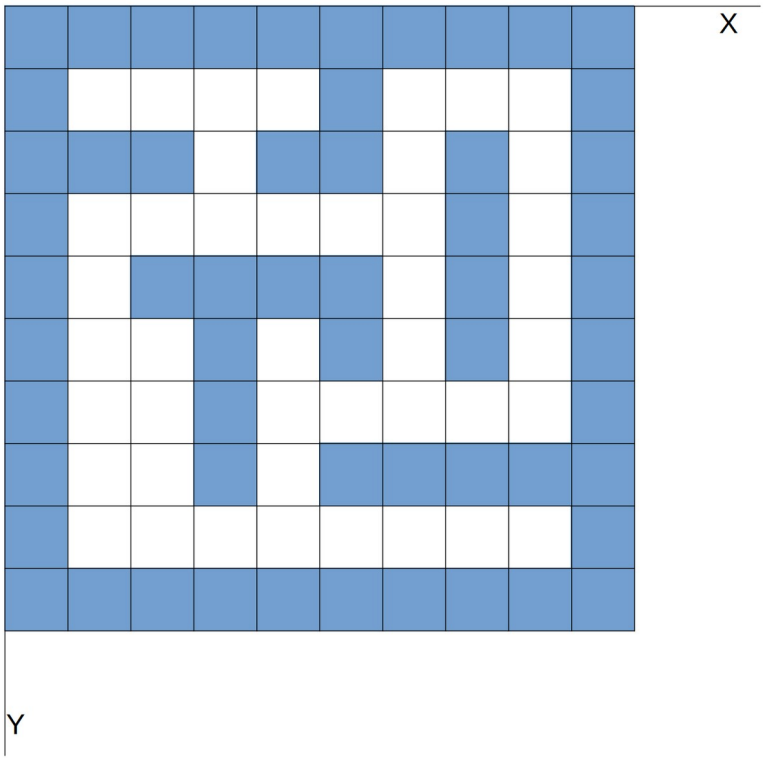

Рисунок 1 — Лабиринт, заданный на прямоугольной сетке

Также в лабиринте заданы две точки — начала и конца пути, при этом координаты точек являются действительными числами. Возможные расположения точек:

- внутри любых клеток лабиринта, не отмеченных как «стена»;
- на границе двух клеток (на горизонтальной или вертикальной линии сетки), при этом одна из клеток не должна быть отмечена как «стена», и предполагается, что точка принадлежит этой клетке;
- на пересечении линий сетки, при этом по крайней мере одна из ближайших клеток не должна быть отмечена как «стена»,

и предполагается, что точка может принадлежать любой из таких клеток.

Требуется реализовать алгоритм, позволяющий построить кратчайший маршрут из начальной точки в конечную и определить его длину. Маршрут может пролегать внутри клеток, а также вдоль их границ (при условии отсутствия «стены» в таких клетках). Примеры построенного маршрута приведены на рисунке 2.

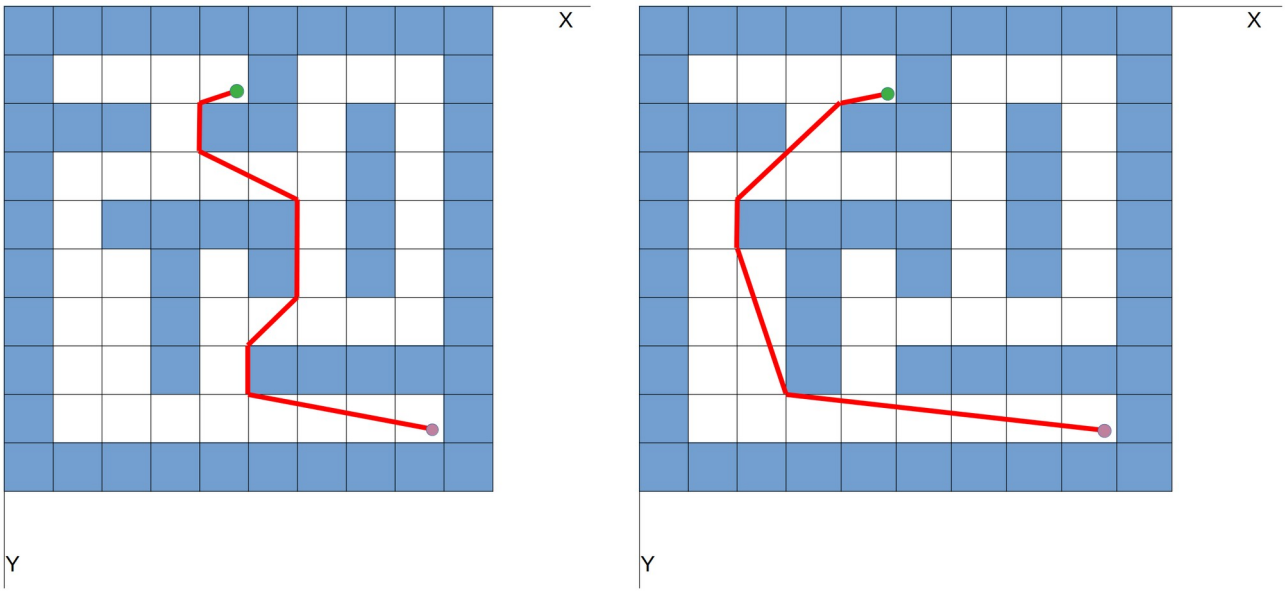

Рисунок 2 — Примеры маршрутов в лабиринте

На рисунке показаны два возможных варианта пути, однако правильным решением данной задачи является левый. Несмотря на то, что путь от начальной до конечной точки проходит через одинаковое количество клеток, маршрут, представленный слева, имеет меньшую длину.

Входные данные задаются в текстовой форме. Формат входных данных следующий:

- первая строка файла содержит шесть чисел, разделённых пробелами — ширина (W) и высота (H) лабиринта, координаты x_1 и y_1 начальной точки маршрута, координаты x_2 и y_2 конечной точки лабиринта;
- следующие H строк содержат информацию о стенах лабиринта — символом «#» обозначена стена, пробелом — её отсутствие;
- $3 \leqslant W \leqslant 1000$, $3 \leqslant H \leqslant 1000$, $0 < x_1 < W$, $0 < x_2 < W$, $0 < y_1 < H$, $0 < y_2 < H$;

Пример входных данных в соответствии с рисунком 2:

```
10 10 4.8 1.8 8.8 8.8
##########
#    #   #
### ## # #
#      # #
# #### # #
#  # # # #
#  #     #
#  # #####
#        #
##########
```

Результатом работы программы должна быть минимальная длина маршрута, вычисленная с точностью до третьего знака после десятичной точки. Например, для приведённого примера результатом будет 12.358. Если маршрута не существует (нет пути из начальной точки в конечную, либо одна из указанных точек находится внутри стены), в качестве ответа следует вывести значение -1.

Задача №2. Комнаты

На плоскости задано *N* точек. Некоторые точки соединены отрезками, образуя выпуклые многоугольники. Некоторые из отрезков будут общими для двух многоугольников. Один и тот же отрезок не может принадлежать более чем двум многоугольникам. Точки могут совпадать (иметь одинаковые значения координат), но такие совпадающие точки не могут быть в составе одного и того же многоугольника.

Введём следующие обозначения:

- Комната — выпуклый многоугольник, образованный отрезками, соединяющими заданные точки. Никакие две точки в составе комнаты не могут совпадать.
- Стена — отрезок, соединяющий две заданные точки и принадлежащий только одной комнате.
- Портал — отрезок, соединяющий две заданные точки и являющийся общим для двух соседних комнат.

Каждая точка помимо координат имеет порядковый номер, начиная с 0. Также порядковые номера есть у каждой комнаты.

Комнаты соединяются только с помощью порталов. Если две комнаты имеют одну общую точку, то такая точка не может считаться порталом. Также допустимы варианты комнат, не имеющие стен (все граничные отрезки являются порталами), либо не имеющие порталов (изолированные комнаты).

Пример исходных данных в графической форме приведён на рисунке 3.

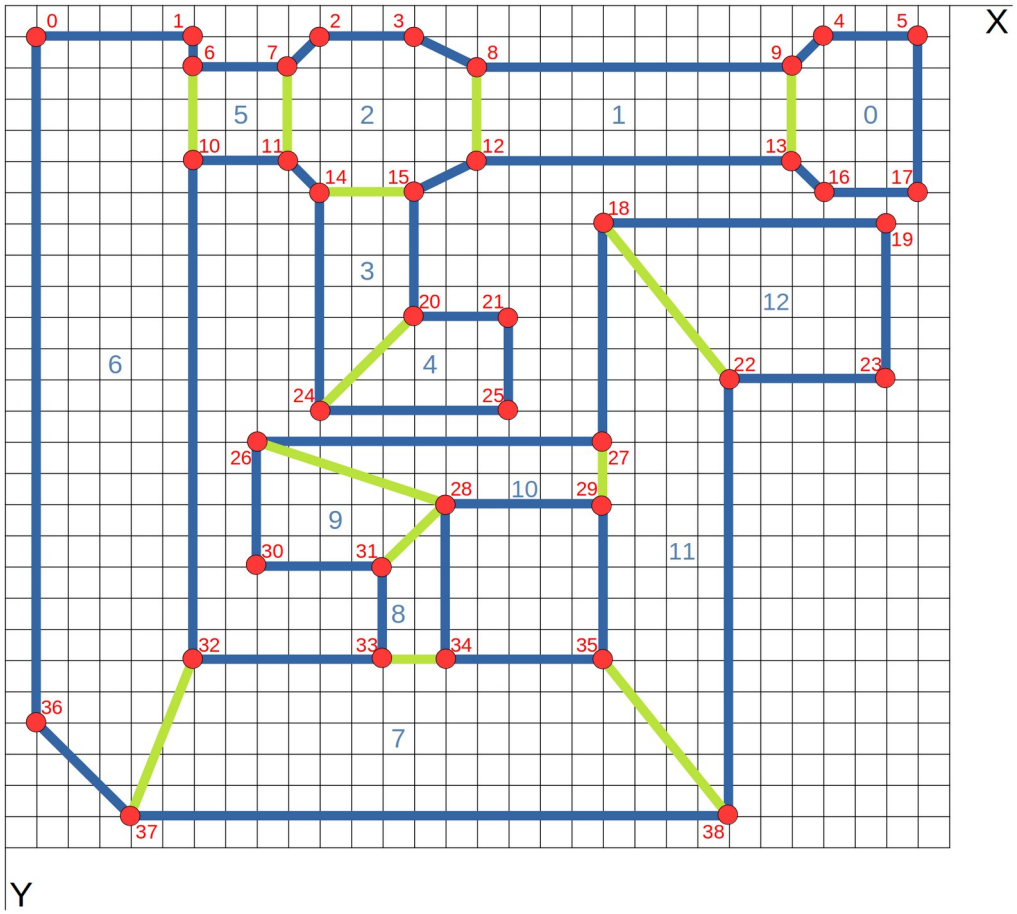

Рисунок 3 — Один из вариантов входных данных

Входные данные задаются в текстовой форме следующим образом:

- Числа разделяются любым количеством пробельных символов (пробел, табуляция, перевод строки и т. п.).
- Первое число в файле — количество точек N, $3 < N < 10000$.
- Следующие N пар действительных чисел — координаты x и y каждой из точек по порядку, начиная с точки номер 0.
- Далее указано количество комнат M, $1 < M < 3000$.
- Следом идут M групп целых чисел:
  - количество вершин K в комнате;
  - K порядковых номеров точек, составляющих комнату и указанных по порядку по часовой стрелке.

Для фигуры, представленной на рисунке 3, входные данные могут быть заданы следующим образом:

```
39
1 1 6 1 10 1 13 1 26 1 29 1
6 2 9 2 15 2 25 2
6 5 9 5 15 5 25 5
10 6 13 6 26 6 29 6
19 7 28 7
13 10 16 10
23 12 28 12
10 13 16 13
8 14 19 14
14 16 19 16
8 18 12 18
6 21 12 21 14 21 19 21
1 23
4 26 23 26
13
6 4 5 17 16 13 9
4 8 9 13 12
8 2 3 8 12 15 14 11 7
4 14 15 20 24
4 20 21 25 24
4 6 7 11 10
7 0 1 6 10 32 37 36
6 32 33 34 35 38 37
4 28 34 33 31
4 26 28 31 30
4 26 27 29 28
6 18 22 38 35 29 27
4 18 19 23 22
```

Выберем произвольным образом точку A, находящуюся внутри одной из комнат, и проведём из этой точки в произвольном направлении луч. В данных условиях этот луч обязательно пересечёт по крайней мере одну стену. Отметим на луче точку B — ближайшее к точке A пересечение со стеной (не с порталом). Выберем на луче произвольную точку C. Если расстояние от точки A до точки C меньше либо равно расстоянию от точки A до точки B, то такую точку C будем называть видимой из точки A.

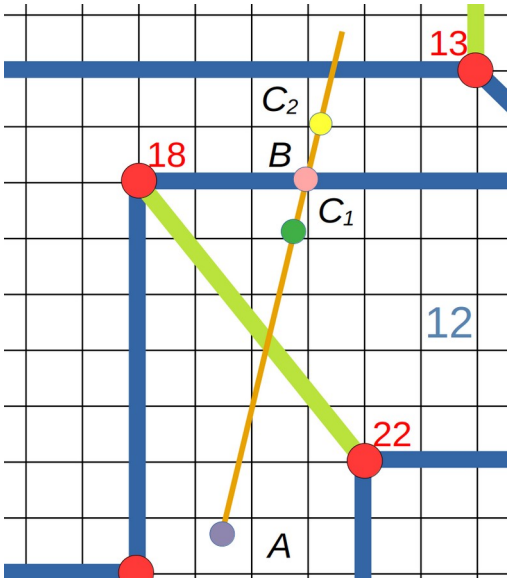

Рисунок 4 — Расположение точек на луче

На рисунке 4 точка C_1 является видимой из точки A, а точка C_2 — невидимой. Факт пересечения луча с порталом не является помехой.

Предположим, что точка A находится внутри комнаты P, а точка C видима и находится в комнате Q. В этом случае будем считать, что комната Q видима из комнаты P. Другими словами, комната Q считается видимой из комнаты P, если существуют такие две точки $A \in P$ и $C \in Q$, что отрезок AC не пересекает ни одной стены. При этом точки, соединяющие стену и портал, считаются принадлежащими стене, а следовательно, препятствием, и должны рассматриваться как точки пересечения со стеной. Так, на рисунке 5 точка C считается невидимой из точки A.

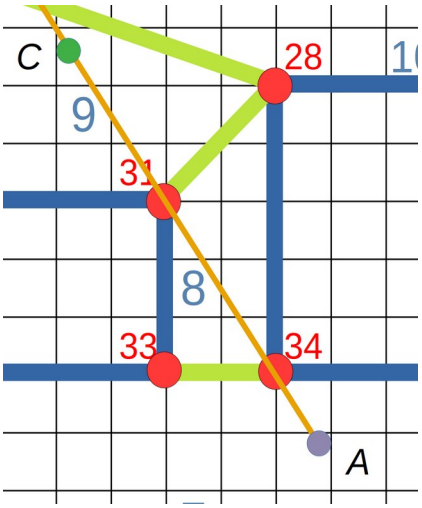

Рисунок 5 — Луч, проходящий через границы порталов

Требуется реализовать алгоритм, позволяющий для каждой комнаты определить, какие комнаты видимы из неё.

Результат должен быть представлен в текстовой форме. Вывод должен содержать *M* групп чисел, имеющих следующий формат:

- Первое число в группе — номер текущей комнаты.
- Второе число в группе — количество комнат *K*, видимых из текущей.
- Следующие K чисел — номера видимых комнат в любом порядке.

Например, для приведённого выше примера входных данных результат может иметь следующий вид:

```
0  5 1 2 3 5 6
1  5 0 2 3 5 6
2  6 0 1 3 4 5 6
3  6 0 1 2 4 5 6
4  3 2 3 5
5  7 0 1 2 3 4 6 7
6  8 0 1 2 3 5 7 8 11
7  7 5 6 8 9 10 11 12
8  5 6 7 9 10 11
9  4 7 8 10 11
10 5 7 8 9 11 12
11 6 6 7 8 9 10 12
12 3 7 10 11
```

Задача №3. Музыка

Одним из способов записи музыки на компьютере является язык MML (Music Macro Language). Существует множество вариантов этого языка, но в данной задаче используется один из самых простых.

Выражение на языке MML представляет собой строку, содержащую символьные команды. Каждая команда обозначается одним символом, после которого, в зависимости от типа команды, могут следовать различные параметры и флаги. Список основных команд приведены в таблице 1.

Таблица 1 — Команды языка MML

№ п/п.	Команда	Формат	Описание
1	T	`T <nn>`	Установка темпа в значение nn (от 32 до 255, по умолчанию — 120).
2	V	`V <nn>`	Установка громкости в значение nn (от 0 до 15, по умолчанию 12)
3	O	`O <nn>`	Выбор октавы номер nn (от 0 до 7, по умолчанию 4 — первая октава)
4	L	`L <nn>`	Установка длительности по умолчанию в значение nn (целые положительные делители числа 96, по умолчанию 4).
5	<	`<`	Уменьшить текущий номер октавы на 1. Если текущий номер октавы равен 0, то команда не имеет эффекта.
6	>	`>`	Увеличить текущий номер октавы на 1. Если текущий номер октавы равен 7, то команда не имеет эффекта.
7	R	`R [nn] [.]`	Сделать паузу. Если указано значение nn, то использовать его в качестве длительности, иначе использовать длительность по умолчанию. Если есть точка, то длительность увеличивается наполовину.
8	A…G	`<cmd> [+|-] [nn] [.] [&]`	Воспроизвести звук, соответствующий указанной в команде ноте (C — нота «до») в текущей октаве. Знак «+» или «-», если он присутствует, обозначает соответственно диез (увеличение высоты ноты на полтона) или бемоль (уменьшение высоты ноты на полтона). Если указано значение nn, то использовать его в

			качестве длительности, иначе использовать длительность по умолчанию. Если есть точка, то длительность увеличивается наполовину. Если есть знак «&» и следующая нота имеет ту же высоту, то длительности складываются.
9	N	N <nn> [.] [&]	Воспроизвести звук, соответствующий указанной числом nn клавише в соответствии со стандартом MIDI (от 21 до 108), используя

Обозначения:

nn	— целое неотрицательное число
<...>	— обязательный параметр
[...]	— необязательный параметр
a | b	— альтернативные варианты

Команды в выражении опционально могут быть разделены любым количеством пробельных символов (пробелы, табуляции, переводы строки). Регистр букв не имеет значения. В качестве единицы измерения темпа используется количество четвертных нот (♩) в минуту.

В данной задаче не предполагается какой-либо вывод звука. Требуется разработать алгоритм, определяющий длительность всей композиции в микросекундах, а также самую низкую и самую высокую ноту (используя номера клавиш в соответствии со стандартом MIDI, т. е. нота «до» первой октавы имеет номер 60). Результатом будут 3 целых числа, если выражение на языке MML корректно, либо -1, если оно содержит ошибки.

Пример входных данных:

```
T120 L8 O5
a4a4e16f16g4efda4e16d16c4c
d4d4<a16b16>c4c<bab>c<a4a4>
d4d4<a16b16>c4c<bab>c<a4a4>
L16
a4a8gfefg4fed<a>dfa8gfedc8.<ab>c
d4d8c<bab>c4c8<b>c<bag+eg+b ab>ceagfe
d4d8c<bab>c4c8<b>c<bag+eg+b a4>a8r8
```

Пример выходных данных:

```
24000 64 96
```

Задача №4. Приёмник

Через некоторый канал связи передаётся текстовая строка в коде ASCII. Передача осуществляется по дифференциальной паре, в которой возникают синфазные помехи. Формат передачи данных известен, но неизвестна скорость. При этом формат позволяет принимающей стороне автоматически определить скорость передачи данных. Требуется реализовать алгоритм, позволяющий декодировать передаваемую строку.

Входные данные представлены последовательностью пар действительных чисел, соответствующих значениям уровня сигнала на прямом и инверсном входах. Например, для битовой последовательности «11011010011001» входные данные можно наглядно представить в виде графика на рисунке 6.

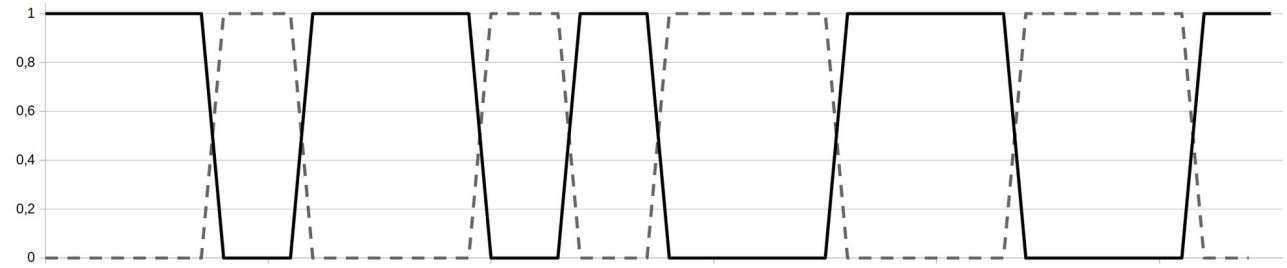

Рисунок 6 — Пример идеального дифференциального сигнала

На рисунке сплошной линией показаны значение на прямом входе, а штриховой линией — на инверсном входе. Показан идеальный вариант, без помех. На практике же линия передачи данных может быть подвержена воздействию внешних помех, что приведёт к искажению данных, как показано на рисунке 7.

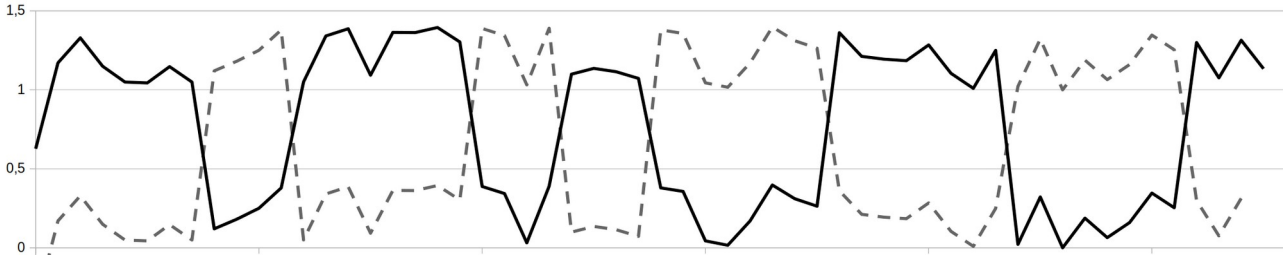

Рисунок 7 — Пример зашумлённого дифференциального сигнала

Тем не менее, несмотря на наличие шумов, дифференциальная пара позволяет в рамках данной задачи однозначно определить, какие логические уровни (высокий или низкий) передаются через канал связи.

Данные передаются с помощью прямоугольных импульсов разной ширины. При передаче импульса длительностью *t*, половину этого времени на прямом входе будет находиться низкий уровень, на инверсном — высокий, а другую половину времени на прямом входе будет находиться высокий уровень, а на инверсном — низкий.

Протокол передачи данных следующий:

- В обычном состоянии, до передачи данных, на прямом входе неопределённо долго присутствует высокий уровень.
- В начале блока располагается сигнал синхронизации — неопределённое количество импульсов длительностью *6x*. Гарантируется, что количество этих импульсов — не менее 4. Значение параметра *x* заранее неизвестно и должно быть определено автоматически.
- За сигналом синхронизации следует сигнал начала блока данных — один импульс длительностью *2x*.
- Далее идёт *k* символов — наборов из 8 импульсов, определяющих значения битов передаваемого кода символа, начиная с самого младшего. Нулевое значение бита кодируется импульсом длительностью *2x*, значение 1 кодируется импульсом длительностью *4x*. Количество символов *k* заранее неизвестно, символы должны приниматься до тех пор, пока не будет обнаружен признак конца блока.
- После всех символов передаётся импульс длительностью *6x* — признак конца блока.
- После признака конца блока на прямой вход подаётся высокий уровень на неопределённое время.
- При необходимости процесс повторяется.

Рассмотрим пример. Пусть передаётся слово «Hello». Коды букв в различных системах счисления представлены в таблице 2.

Таблица 2 — Коды символов, используемых в примере

Символ	Десятичный код	Двоичный код
H	72	01001000_2
e	101	01100101_2
l	108	01101100_2
o	111	01101111_2

Таким образом, передаваться будет битовая последовательность «0001 0010 1010 0110 0011 0110 0011 0110 1111 0110». Если принять параметр *x* равным 2, то, в соответствии с описанным выше протоколом, значения, подаваемые на вход, могут быть следующими:

```
 0.71 -0.29  0.95 -0.05  0.99 -0.01  1.23  0.23
 0.77 -0.23  1.28  0.28 -0.21  0.79  0.01  1.01
-0.19  0.81  0.21  1.21 -0.13  0.87 -0.19  0.81
 1.20  0.20  1.25  0.25  0.96 -0.04  0.73 -0.27
 0.78 -0.22  0.72 -0.28  0.29  1.29  0.22  1.22
 0.12  1.12 -0.26  0.74 -0.19  0.81 -0.09  0.91
 0.98 -0.02  0.97 -0.03  0.99 -0.01  0.93 -0.07
 0.92 -0.08  0.89 -0.11 -0.06  0.94  0.15  1.15
-0.25  0.75 -0.27  0.73  0.19  1.19  0.19  1.19
 0.87 -0.13  0.73 -0.27  1.18  0.18  1.09  0.09
 0.85 -0.15  0.86 -0.14 -0.06  0.94 -0.10  0.90
 0.00  1.00  0.11  1.11  0.24  1.24  0.07  1.07
 1.16  0.16  0.87 -0.13  0.95 -0.05  1.25  0.25
 1.25  0.25  1.18  0.18  0.22  1.22 -0.12  0.88
 0.83 -0.17  1.29  0.29  0.15  1.15 -0.03  0.97
 0.75 -0.25  0.71 -0.29 -0.29  0.71 -0.12  0.88
 1.20  0.20  1.18  0.18 -0.02  0.98  0.02  1.02
 0.71 -0.29  1.22  0.22 -0.12  0.88  0.02  1.02
 0.02  1.02 -0.20  0.80  1.19  0.19  0.86 -0.14
 1.09  0.09  1.25  0.25 -0.25  0.75 -0.23  0.77
 0.99 -0.01  0.84 -0.16  0.01  1.01 -0.08  0.92
 1.24  0.24  1.07  0.07  0.02  1.02  0.27  1.27
-0.19  0.81  0.25  1.25  0.85 -0.15  1.22  0.22
 0.76 -0.24  1.00  0.00  0.15  1.15  0.16  1.16
 1.29  0.29  0.93 -0.07 -0.07  0.93  0.04  1.04
 0.29  1.29  0.26  1.26  0.79 -0.21  1.25  0.25
 0.77 -0.23  0.95 -0.05 -0.17  0.83  0.23  1.23
 0.89 -0.11  0.76 -0.24 -0.22  0.78 -0.14  0.86
-0.24  0.76  0.22  1.22  0.93 -0.07  1.11  0.11
 0.76 -0.24  1.14  0.14  0.30  1.30  0.01  1.01
 0.82 -0.18  1.24  0.24 -0.22  0.78 -0.28  0.72
 1.24  0.24  0.85 -0.15 -0.18  0.82 -0.10  0.90
-0.27  0.73 -0.06  0.94  0.81 -0.19  1.22  0.22
 0.97 -0.03  1.01  0.01  0.05  1.05 -0.04  0.96
```

```
 0.26  1.26 -0.24  0.76  1.02  0.02  0.96 -0.04
 1.29  0.29  1.16  0.16  0.17  1.17  0.26  1.26
 1.03  0.03  1.00  0.00  0.26  1.26 -0.01  0.99
 1.03  0.03  0.79 -0.21  0.22  1.22 -0.29  0.71
 0.84 -0.16  1.10  0.10  0.23  1.23 -0.18  0.82
-0.04  0.96  0.05  1.05  1.00  0.00  1.01  0.01
 0.87 -0.13  1.12  0.12  0.14  1.14 -0.03  0.97
-0.01  0.99  0.02  1.02  0.85 -0.15  1.00  0.00
 0.76 -0.24  0.89 -0.11  0.25  1.25 -0.28  0.72
 0.79 -0.21  0.91 -0.09  0.02  1.02 -0.12  0.88
-0.23  0.77  0.04  1.04  0.77 -0.23  1.19  0.19
 1.13  0.13  0.93 -0.07  0.18  1.18  0.10  1.10
 0.05  1.05  0.14  1.14  0.71 -0.29  1.05  0.05
 1.16  0.16  1.17  0.17  0.25  1.25  0.10  1.10
 1.14  0.14  0.93 -0.07  0.27  1.27  0.25  1.25
 1.24  0.24  1.08  0.08 -0.15  0.85  0.14  1.14
 1.16  0.16  1.06  0.06 -0.14  0.86 -0.19  0.81
-0.24  0.76 -0.28  0.72  0.70 -0.30  0.96 -0.04
 1.06  0.06  0.91 -0.09  0.12  1.12 -0.11  0.89
-0.16  0.84 -0.17  0.83  0.80 -0.20  0.86 -0.14
 1.04  0.04  1.30  0.30  0.28  1.28  0.24  1.24
 1.03  0.03  0.88 -0.12 -0.18  0.82  0.09  1.09
 0.29  1.29 -0.07  0.93  0.86 -0.14  1.06  0.06
 0.70 -0.30  0.81 -0.19 -0.28  0.72 -0.08  0.92
 0.16  1.16 -0.02  0.98  0.70 -0.30  1.20  0.20
 0.77 -0.23  1.29  0.29 -0.17  0.83 -0.02  0.98
 0.98 -0.02  0.82 -0.18  0.18  1.18  0.09  1.09
 0.28  1.28 -0.28  0.72  1.11  0.11  0.84 -0.16
 0.82 -0.18  1.14  0.14  0.20  1.20 -0.07  0.93
-0.10  0.90 -0.30  0.70  0.90 -0.10  1.13  0.13
 1.08  0.08  0.92 -0.08  0.28  1.28 -0.13  0.87
 0.29  1.29 -0.04  0.96  0.88 -0.12  0.78 -0.22
 1.13  0.13  1.26  0.26 -0.26  0.74 -0.22  0.78
-0.24  0.76  0.27  1.27  1.20  0.20  1.27  0.27
 1.08  0.08  0.76 -0.24  0.15  1.15 -0.09  0.91
 1.28  0.28  1.03  0.03  0.16  1.16  0.06  1.06
 0.19  1.19 -0.04  0.96  1.17  0.17  0.70 -0.30
 0.73 -0.27  1.28  0.28  0.20  1.20 -0.27  0.73
 0.12  1.12 -0.29  0.71  1.01  0.01  1.10  0.10
 0.99 -0.01  0.83 -0.17  0.04  1.04 -0.26  0.74
 1.18  0.18  1.05  0.05 -0.29  0.71 -0.26  0.74
 0.10  1.10 -0.03  0.97 -0.26  0.74  0.17  1.17
 1.17  0.17  0.84 -0.16  1.09  0.09  0.78 -0.22
 0.96 -0.04  1.07  0.07  0.82 -0.18  0.72 -0.28
 1.23  0.23  0.96 -0.04  1.09  0.09  1.19  0.19
 1.16  0.16  0.78 -0.22  0.89 -0.11  0.89 -0.11
```

Важное ограничение: в решении данной задачи не следует использовать массивы и другие аналогичные структуры данных. Получаемые на вход данные хранить нет необходимости.