Рекурсии

1 Ханойские башни

Начнем обсуждение этой темы с задачи о Ханойских башнях. Головоломка “Ханойские башни” состоит из трёх стержней, на один из которых нанизано несколько колец. Все кольца имеют разный диаметр и расположены снизу вверх по убыванию диаметра.
За один ход разрешается снять верхнее кольцо с любого стержня и переместить его на другой стержень. При этом запрещается класть кольцо на кольцо меньшего диаметра.

Вот один из возможных алгоритмов. Пронумеруем кольца сверху вниз числами от 1 до n. Пусть нам нужно переложить n колец с первого стержня на третий. Будем рассуждать по индукции.

Если n=1, то решение задачи очевидно.

Пусть мы умеем перекладывать любое количество колец, меньшее n. Тогда n колец можно переложить следующим образом.

1. Перекладываем верхние n-1 колец с первого стержня на второй.
2. Перекладываем последнее (самое большое) кольцо с первого стержня на третий.
3. Перекладываем n-1 колец со второго стержня на третий.

Попробуем написать процедуру, печатающую последовательность перекладываний согласно описанному алгоритму.

```
procedure Hanoi (n, i, j, k : integer);      {переложить n колец
                                 с i-го стержня на j-й, использую k-й стержень
                                 в качестве вспомогательного}
begin
  if n=1 then writeln (i,'->',j)
  else begin
    Hanoi(n-1,i,k,j);       {Перекладываем n-1 колец
                                 с i-го стержня на k-й}
    writeln (i,'->',j);       {Перекладываем оставшееся кольцо
                                 с i-го стержня на j-й}
    Hanoi(n-1,k,j,i);       {Перекладываем n-1 колец
                                 с k-го стержня на j-й}
  end;
end;
```

Обратите внимание: наша процедура вызывает сама себя (но с другими параметрами). Так, Hanoi(2, 1,2,3) вызывает Hanoi(1, 1,3,2) и Hanoi(1, 3,2,1).

Определение. Процедуры, вызывающие себя в процессе работы называются *рекурсивными*.

Для того, чтобы программа не зацикливалась, рекурсивная процедура должна содержать *терминальное условие* – условие, при котором новых вызовов функции не происходит. В нашем случае таким условием служит условие n = 1. Уже при n = 3 последовательность вызовов функций будет довольно сложной, но мы пока не будем углубляться в механизмы работы рекурсии, а перейдем к следующей задаче.

2 Наибольший общий делитель

Всех вас в свое время на уроках математики учили находить наибольший общий делитель двух чисел. Для этого нужно было первым делом разложить их на простые множители. Сейчас мы рассмотрим другой алгоритм, который и реализовывать проще, да и работает он быстрее. Он основан на следующем свойстве НОД:

НОД(a,b)=НОД(a-b,b) при a≥b.

Это свойство несложно доказать. Если d – делитель чисел a и b, то на d делится и разность чисел a и b. Обратно, если d является делителем чисел a-bи b, то и их сумма (a-b)+b=a делится на d. Таким образом, всякий общий делитель чисел a и b является общим делителем чисел a-b и b, и наоборот. Поэтому и наибольшие делители у этих пар чисел совпадают.
Пользуясь этой формулой, можно легко написать рекурсивную функцию для вычисления НОД. Надо лишь не забывать про терминальное условие.

```
function NOD (a,b : integer) : integer;
begin
  if a<b then swap(a,b);       {переставляем большее число
                               на первое место}
  if  b=0 then NOD:=a          {терминальное условие: НОД(a,0)=a}
  else NOD:=NOD(a-b,b);
end;
```

Работу алгоритам можно еще ускорить, если воспользоваться формулой

НОД(a,b)=НОД(a mod b,b).

Алгоритм вычисления НОД по этой формуле называют *алгоритмом Евклида.*
Заметим, что эта формула верна для всех a и b, но если a<b, то a mod b = a и формула вырождается в тривиальную. Чтобы обойтись без процедурыswap, можно пользоваться формулой НОД(a,b)=НОД(b, a mod b).

```
function NOD (a,b : integer) : integer;
begin
  if  b=0 then NOD:=a          {терминальное условие: НОД(a,0)=a}
  else NOD:=NOD(b, a mod b);
end;
```

3 Возведение в степень

Формулировка задачи очень проста: требуется возвести число a в степень n. Проще всего, конечно, умножить число a на себя нужное число раз. Этот алгоритм потребует n-1 умножений. Можно ли существенно ускорить, этот алгоритм? Оказывается, да. Заметим, что если мы хотим вычислить, например,a^{10}, то, вычислив на некотором шаге a^5, можно умножить его на себя и, тем самым, сэкономить несколько умножений. Такой трюк можно использовать для любой четной степени, что позволяет написать формулы, аналогичные формуле из предыдущей задачи:

Эта формула позволяет написать рекурсивную функцию вычисления степени (не забывая о терминальном условии n=1).

Итак, подведём некоторые итоги.

- *Рекурсия* – это прием программирования, при котором функция (процедура) вызывает сама себя.
- Для того, чтобы этот процесс вызовов не зацикливался, в функции обязательно должно быть *терминальное условие* – при выполнении такого условия процесс рекурсивных вызовов прерывается.
- Любой рекурсивный алгоритм можно реализовать и не прибегая к рекурсии, но рекурсия позволяет записывать многие алгоритмы в более простой форме. При этом рекурсивные алгоритмы обычно более требовательны к ресурсам (памяти), чем нерекурсивные.

Задача №199. НОД

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

Ограничение по времени, *сек*	1
Ограничение по памяти,*мегабайт*	64

Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	PHP	Python 2.7	Mono C#	Ruby	Python 3.1	Haskell
Min время, *сек*	0.002	0.002	0.002	0.004	0.119	0.018	0.024	0.043	0.04	0.048	0.005
Среднее время,*сек*	0.01	0.009	0.014	0.011	0.412	0.018	0.043	0.067	0.04	0.063	0.005
Верных решений	284	9	482	48	39	1	3	18	1	30	2

Даны два числа. Найти их наибольший общий делитель.

Входные данные

Вводятся два натуральных числа, не превышающих 10^9.

Выходные данные

Выведите НОД введенных чисел.

Примеры

входные данные

```
9 12
```

выходные данные

```
3
```

Задача №200. Площадь комнаты

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

		Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Mono C#	Python 3.1	Haskell
Ограничение по времени, *сек*	1	**Min время, *сек***	0.001	0.004	0.002	0.004	0.158	0.044	0.049	0.004
Ограничение по памяти, *мегабайт*	64	**Среднее время, *сек***	0.006	0.006	0.013	0.011	0.441	0.078	0.081	0.022
		Верных решений	108	5	188	21	5	2	22	3

Требуется вычислить площадь комнаты в квадратном лабиринте.

Входные данные

В первой строке вводится число *N* – размер лабиринта (3 <= *N* <= 10). В следующих *N* строках задан лабиринт (‘.’ – пустая клетка, ‘*’ – стенка). И наконец, последняя строка содержит два числа – номер строки и столбца клетки, находящейся в комнате, площадь которой необходимо вычислить. Гарантируется, что эта клетка пустая и что лабиринт окружен стенками со всех сторон.

Выходные данные

Требуется вывести единственное число – количество пустых клеток в данной комнате.

Примеры

входные данные

```
5
*****
**..*
*.*.*
*..**
*****
2 4
```

выходные данные

```
3
```

Задача №1414. Фишки

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

		Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Perl	Python 3.1	Haskell
Ограничение по времени, *сек*	1									

Ограничение по памяти, *мегабайт*	64

Min время, *сек*	0.002	0.004	0.002	0.003	0.119	0.01	0.052	0.005
Среднее время, *сек*	0.004	0.007	0.013	0.01	0.216	0.039	0.099	0.006
Верных решений	39	3	107	22	3	2	7	2

Дана полоска из клеток, пронумерованных от 1 до *N*. На каждом ходе разрешено поставить фишку на клетку (если её там еще нет) или снять фишку с клетки (если она там есть). При этом, можно выбрать не любую клетку, а только клетку под номером 1 или клетку, следующую за самой первой фишкой.
Изначально полоска пуста. Требуется занять все клетки.

Входные данные

С клавиатуры вводится натуральное число N ($1 \le N \le 10$).

Выходные данные

Требуется вывести последовательность номеров клеток, с которыми совершается действие. Если фишка снимается, то номер клетки должен выводиться со знаком минус. Количество действий не должно превышать 10^4.
Если существует несколько возможных решений задачи, то разрешается вывести любое.

Примеры

Ввод	Вывод
3	1 2 -1 3 1

Примеры

входные данные

```
3
```

выходные данные

```

```

Задача №1470. Спиралька

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.
:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

Ограничение по времени, *сек*	1

Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Python 2.7	Mono C#	Python 3.1
Ограничение по памяти, *мегабайт*	64	Min время, *сек*	0.002	0.002	0.002	0.002	0.118	0.025	0.045	0.047

Среднее время, *сек*	0.006	0.004	0.01	0.013	0.233	0.025	0.099	0.166
Верных решений	139	4	217	48	16	1	5	22

Выведите двумерный массив, размерами $N \times N$, заполненный числами от единицы до N^2 по спирали. Числовая спираль начинается в левом верхнем углу и закручивается по часовой стрелке.

Входные данные

Входной файл содержит единственное число $1 \le N \le 10$.

Выходные данные

Выведите N^2 чисел – заполненный по спирали массив.

Примеры

входные данные

```
1
```

выходные данные

```
1
```

входные данные

```
2
```

выходные данные

```
1 2
4 3
```

входные данные

```
3
```

выходные данные

```
1 2 3
8 9 4
7 6 5
```

Задача №153. N-е число Фибоначчи

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Perl	Mono C#	Python 3.1	Haskell
Min время, *сек*	0.001	0.002	0.001	0.002	0.117	0.685	0.044	0.046	0.021
Среднее время, *сек*	0.017	0.014	0.014	0.022	0.296	0.713	0.075	0.454	0.077
Верных решений	441	10	583	287	20	4	8	13	3

Максимальное время работы на одном тесте: 1 секунда

Последовательностью Фибоначчи называется последовательность чисел a_0, a_1, ..., a_n, ..., где $a_0 = 0$, $a_1 = 1$, $a_k = a_{k-1} + a_{k-2}$ ($k > 1$).

Требуется найти N-е число Фибоначчи.

Примечание. В программе запрещается использовать циклы.

Входные данные

На вход программы поступает целое неотрицательное число N ($N \le 30$).

Выходные данные

Требуется вывести N-е число Фибоначчи.

Примеры

входные данные

```
3
```

выходные данные

```
2
```

Задача №154. НОД (рекурсивный вариант)

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Python 2.7	Mono C#	Python 3.1
Min время, *сек*	0.001	0.001	0.001	0.002	0.117	0.008	0.044	0.048
Среднее время, *сек*	0.014	0.003	0.015	0.014	0.292	0.035	0.056	0.091
Верных решений	311	8	634	287	59	4	7	12

Максимальное время работы на одном тесте: 1 секунда

Даны два натуральных числа A и B. Требуется найти их наибольший общий делитель.

***Примечание*.** В программе запрещается использовать циклы.

Входные данные

Вводятся два натуральных числа A и B (A, $B \le 10^9$).

Выходные данные

Требуется вывести НОД A и B.

Примеры

входные данные

```
12 42
```

выходные данные

```
6
```

Задача №155. Генератор

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.
:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Python 2.7	Python 3.1
Min время, *сек*	0.009	0.039	0.036	0.034	0.269	0.26	0.184
Среднее время, *сек*	0.054	0.048	0.146	1.044	0.852	0.26	0.426
Верных решений	155	7	244	161	11	1	13

Максимальное время работы на одном тесте: 2 секунды

Даны два натуральных числа N и K. Требуется вывести все цепочки x_1, x_2, ..., x_N такие, что x_i - натуральное и $1 \le x_i \le K$.

Входные данные

Вводятся два натуральных числа N и K (N, $K \le 6$).

Выходные данные

Выведите все требуемые цепочки в произвольном порядке – по одной на строке. Никакая цепочка не должна встречаться более одного раза.

Примеры

входные данные

```
2 3
```

выходные данные

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

Задача №156. Без массивов

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Mono C#	Python 3.1	Haskell
Min время, *сек*	0.001	0.002	0.001	0.003	0.125	0.041	0.047	0.006
Среднее время, *сек*	0.005	0.003	0.006	0.016	0.289	0.042	0.074	0.006
Верных решений	333	12	539	203	24	2	19	1

Максимальное время работы на одном тесте: 1 секунда

Дано натуральное число N и последовательность из N элементов. Требуется вывести эту последовательность в обратном порядке.

Примечание. В программе запрещается объявлять массивы и использовать циклы (даже для ввода и вывода).

Входные данные

В первой строке входных данных содержится натуральное число N ($N \le 10^3$). Во второй строке через пробел идут N целых чисел, по модулю не превосходящих 1000, – элементы последовательности.

Выходные данные

Требуется вывести заданную последовательность в обратном порядке.

Примеры

входные данные

```
2
3 4
```

выходные данные

```
4 3
```

Задача №157. Монетки

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Python 3.1	Haskell
Min время, *сек*	0.18	0.02	0.004	0.128	0.327	1.034	0.261
Среднее время, *сек*	0.478	0.075	0.29	0.481	0.51	1.034	0.37
Верных решений	27	2	118	28	4	1	7

Максимальное время работы на одном тесте: 1 секунда

В Волшебной стране используются монетки достоинством A_1, A_2,..., A_M. Волшебный человечек пришел в магазин и обнаружил, что у него есть ровно по две монетки каждого достоинства. Ему нужно заплатить сумму N. Напишите программу, определяющую, сможет ли он расплатиться без сдачи.

Входные данные

На вход программы сначала поступает число N ($1 <= N <= 10^9$), затем - число M ($1 <= M <= 15$) и далее M попарно различных чисел A_1, A_2,..., A_M ($1 <= A_i <= 10^9$).

Выходные данные

Сначала выведите K - количество монет, которое придется отдать Волшебному человечку, если он сможет заплатить указанную сумму без сдачи. Далее выведите K чисел, задающих достоинства монет. Если решений несколько, выведите вариант, в котором Волшебный человек отдаст наименьшее возможное количество монет. Если таких вариантов несколько, выведите любой из них.

Если без сдачи не обойтись, то выведите одно число 0. Если же у Волшебного человечка не хватит денег, чтобы заплатить указанную сумму, выведите одно число -1 (минус один).

Примеры

входные данные

```
100 6
11 20 30 40 11 99
```

выходные данные

```
3
40 30 30
```

Задача №158. Сумма кубов

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Python 3.1	Haskell
Min время, *сек*	0.001	0.008	0.002	0.005	0.272	0.06	0.035
Среднее время, *сек*	0.246	0.205	0.261	0.349	0.474	0.06	0.22
Верных решений	38	5	94	31	5	1	7

Известно, что любое натуральное число можно представить в виде суммы не более чем четырех квадратов натуральных чисел. Вася решил придумать аналогичное утверждение для кубов - он хочет узнать, сколько кубов достаточно для представления любого числа. Его первая рабочая гипотеза - восемь.

Выяснилось, что почти все числа, которые Вася смог придумать, представляются в виде суммы не более чем восьми кубов. Однако число 239, например, не допускает такого представления. Теперь Вася хочет найти какие-либо другие такие числа, а также, возможно, какую-либо закономерность в представлениях всех остальных чисел, чтобы выдвинуть гипотезу относительно вида всех чисел, которые не представляются в виде суммы восьми кубов.

Помогите Васе написать программу, которая проверяла бы, возможно ли представить данное натуральное число в виде суммы не более чем восьми кубов натуральных чисел, и если это возможно, то находила бы какое-либо такое представление.

Входные данные

Вводится натуральное число $N <= 2*10^9$.

Выходные данные

Требуется вывести не более восьми натуральных чисел, кубы которых в сумме дают *N*. Если искомого представления не существует, то в выходной файл необходимо вывести слово `IMPOSSIBLE`.

Примеры

входные данные

```
239
```

выходные данные

```
IMPOSSIBLE
```

входные данные

```
17
```

выходные данные

```
2 2 1
```