Реку́рсия —

в определении, описании, изображении какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация**, когда объект является частью самого себя**.
Термин «рекурсия» используется в различных специальных областях знаний — от лингвистики до логики, но наиболее широкое применение находит
в математике и информатике.

В математике рекурсия

имеет отношение к методу **определения** функций и числовых рядов: ***рекурсивно заданная функция*** определяет своё значение через обращение к себе самой с другими аргументами.

Конечная рекурсивная функция. $$n! = \begin{cases} n(n-1)!, n > 0 \\ 1, n = 0 \end{cases}$$

Здесь каждое следующее рекурсивное обращение делается с аргументом, меньшим на единицу. Поскольку **n**, по определению, целое неотрицательное число, через **n** рекурсивных обращений вычисление функции гарантированно придёт к частному случаю , на котором рекурсия прекратится.

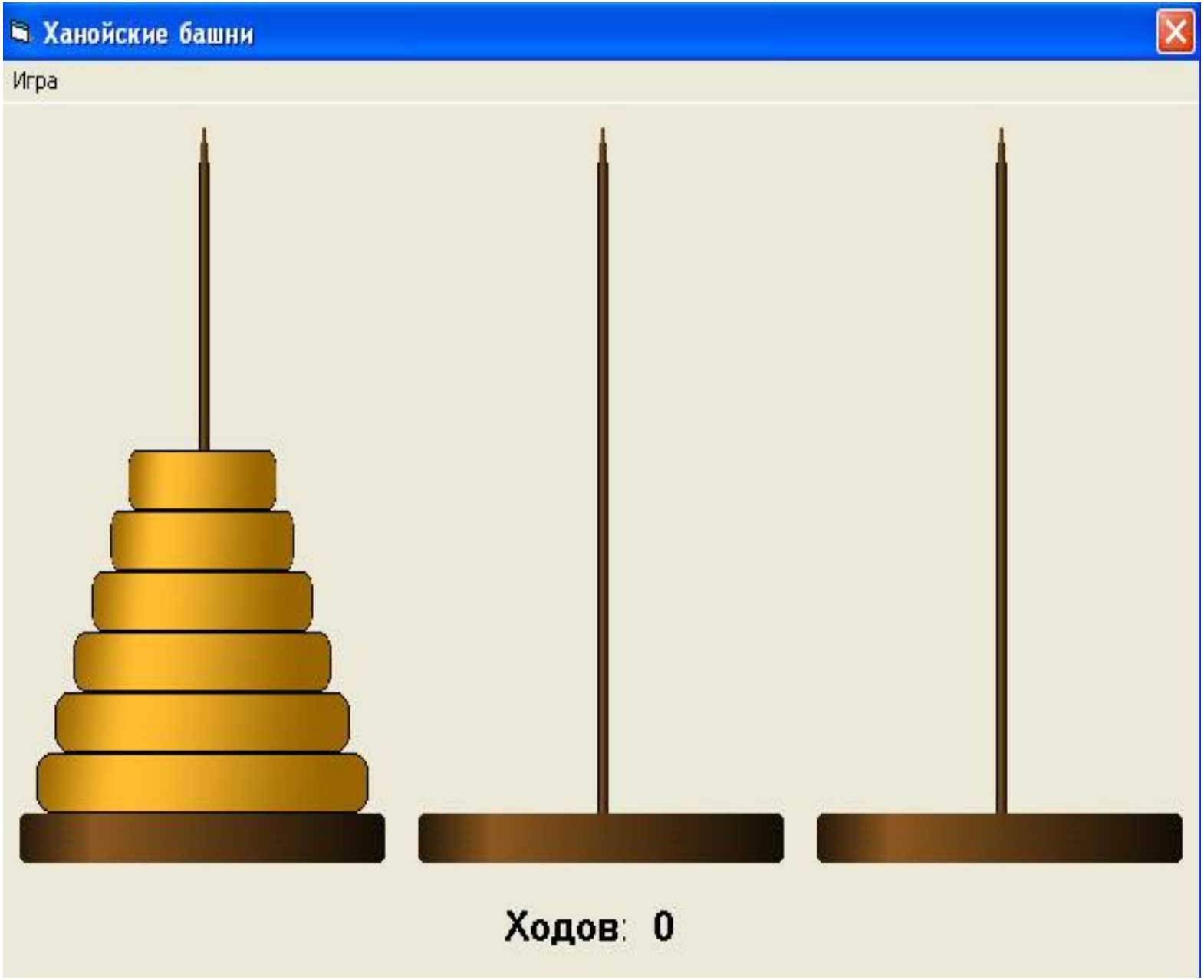
В математике рекурсия

Бесконечная рекурсивная функция.

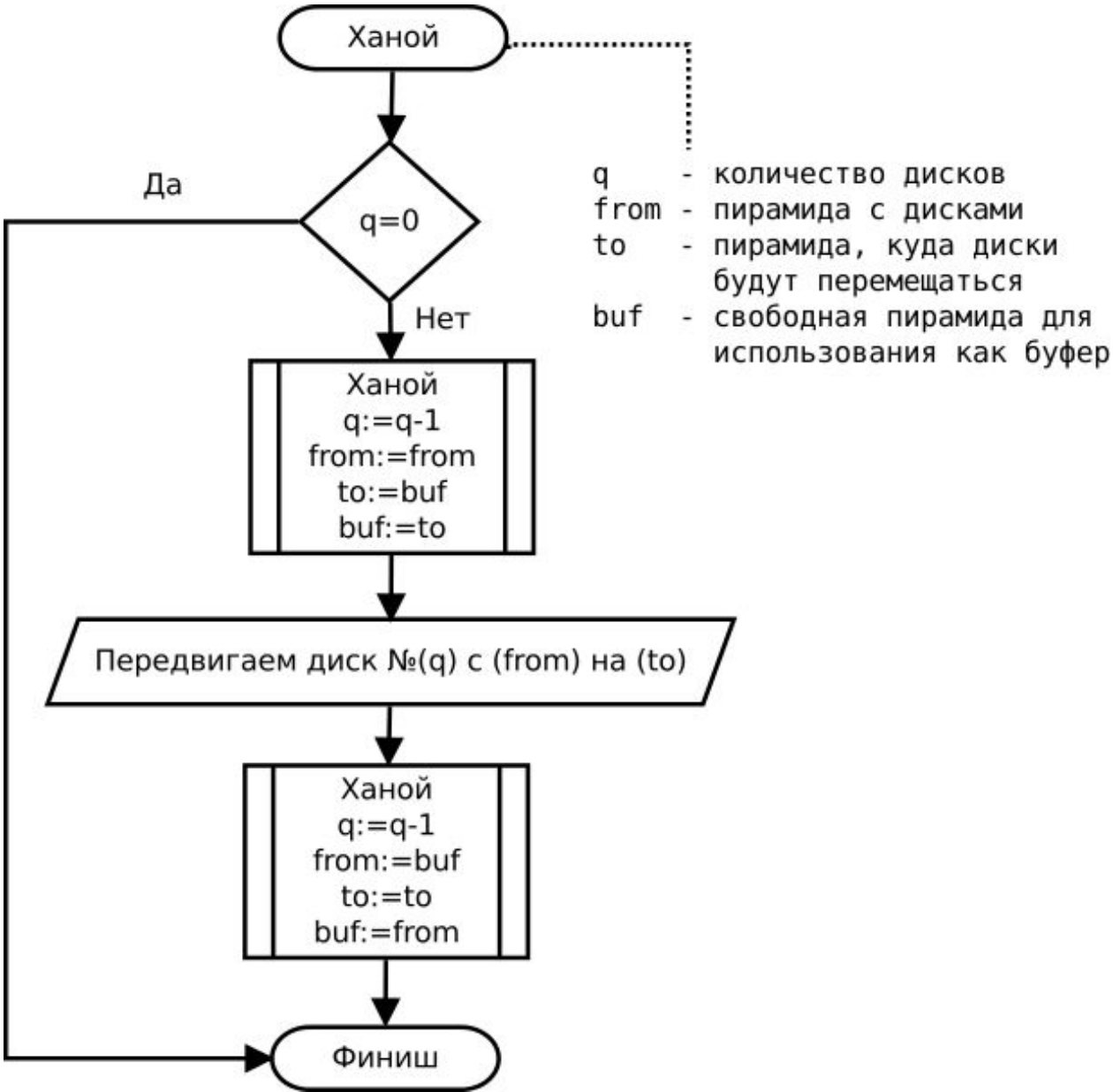
Примером может служить один из вариантов разложения **числа Эйлера:**

$$e = 2 + \cfrac{2}{2 + \cfrac{3}{3 + \cfrac{4}{4 + \ldots}}}$$

Подобным образом могут задаваться бесконечные ряды, бесконечные непрерывные дроби и так далее.


Ханойские башни
Игра
Ходов: 0

Ханойская башня



Ханойская башня

```
//n – количество дисков
//a, b, c – номера штырьков. Перекладывание производится со
    штырька a,
//на штырек b при вспомогательном штырьке c.
procedure Hanoi(n, a, b, c: integer);
begin
  if n > 1 then
  begin
    Hanoi(n-1, a, c, b);
    writeln(a, ' -> ', b);
    Hanoi(n-1, c, b, a);
  end else
    writeln(a, ' -> ', b);
  end;
```

В программировании рекурсия —

вызов функции (процедуры) из неё же самой, непосредственно (*простая рекурсия*) или через другие функции (*сложная* или *косвенная рекурсия*), например, функция A вызывает функцию B, а функция A — функцию B .

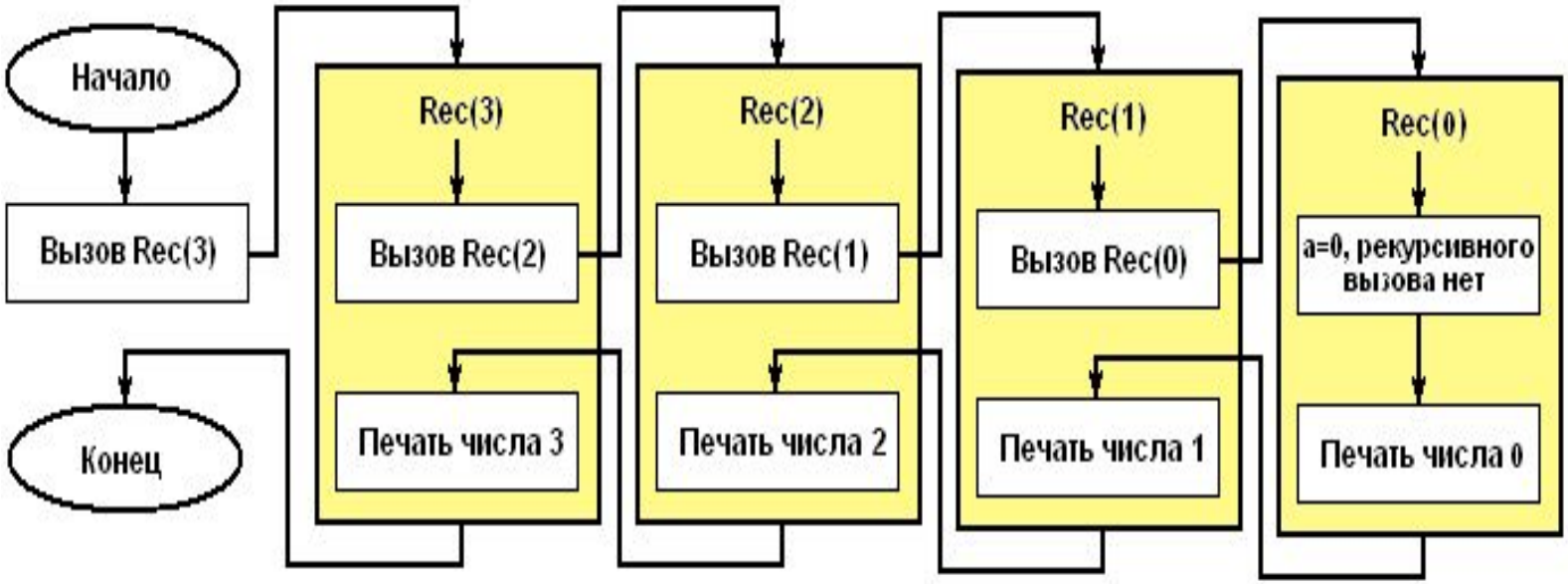
Количество вложенных вызовов функции или процедуры называется глубиной рекурсии. Рекурсивная программа позволяет описать повторяющееся или даже потенциально бесконечное вычисление, причём без явных повторений частей программы и использования циклов.

Структурно рекурсивная функция на верхнем уровне всегда представляет собой команду ветвления (выбор одной из двух или более альтернатив в зависимости от условия (условий), которое в данном случае уместно назвать «условием прекращения рекурсии»), имеющей две или более альтернативные ветви, из которых хотя бы одна является *рекурсивной* и хотя бы одна — *терминальной*. Рекурсивная ветвь выполняется, когда условие прекращения рекурсии ложно, и содержит хотя бы один рекурсивный вызов — прямой или опосредованный вызов функцией самой себя.

Пример рекурсивной процедуры:

```
procedure Rec(a: integer);
begin
  if a>0 then
    Rec(a-1);
  writeln(a);
end;
```

Если в основной программе поставить вызов, например, вида Rec(3). Ниже представлена блок-схема, показывающая последовательность выполнения операторов.



- Процедура Rec вызывается с параметром a = 3. В ней содержится вызов процедуры Rec с параметром a = 2. Предыдущий вызов еще не завершился, поэтому можете представить себе, что создается еще одна процедура и до окончания ее работы первая свою работу не заканчивает. Процесс вызова заканчивается, когда параметр a = 0. В этот момент одновременно выполняются 4 экземпляра процедуры. Количество одновременно выполняемых процедур называют ***глубиной рекурсии.***

- Четвертая вызванная процедура (Rec(0)) напечатает число 0 и закончит свою работу. После этого управление возвращается к процедуре, которая ее вызвала (Rec(1)) и печатается число 1. И так далее пока не завершатся все процедуры. Результатом исходного вызова будет печать четырех чисел: 0, 1, 2, 3.

- В качестве самостоятельного упражнения подумайте, что получится при вызове Rec(4). Также подумайте, что получится при вызове описанной ниже процедуры Rec2(4), где операторы поменялись местами.

```
procedure Rec2(a: integer);
begin
  writeln(a);
  if a>0 then
    Rec2(a-1);
end;
```

- Обратите внимание, что в приведенных примерах рекурсивный вызов стоит **внутри условного оператора.** Это необходимое условие для того, чтобы рекурсия когда-нибудь закончилась. Также обратите внимание, что сама себя процедура вызывает с другим параметром, не таким, с каким была вызвана она сама. Если в процедуре **не используются глобальные переменные**, то это также необходимо, чтобы рекурсия не продолжалась до бесконечности.

Перевод числа в двоичную систему.

```
while x>0 do
begin
  c:=x mod 2;
  x:=x div 2;
  write(c);
end;
```

```
procedure BinaryRepresentation(x: integer);
var
  c, x: integer;
begin
  {Первый блок. Выполняется в порядке
    вызова процедур}
  c := x mod 2;
  x := x div 2;
  {Рекурсивный вызов}
  if x>0 then
    BinaryRepresentation(x);
  {Второй блок. Выполняется в обратном
    порядке}
  write(c);
end;
```