



# ОСНОВЫ программирования

---

Учитель информатики и ИКТ  
ГОО г.Москвы СОШ №310  
«У Чистых прудов»  
Цыбикова Т.Р.



Тема 10.

# РЕКУРСИЯ





# СОДЕРЖАНИЕ

- Рекурсивные объекты
- Рекурсивное определение
- Рекурсия
- Рекурсивный алгоритм
- Пример 1. Определение факториала (слайды 8-11)
- Пример 2. Вычисление степени с натуральным показателем (слайд 12)
- Пример 3. Вычисление чисел Фибоначчи (слайды 13-15)
- Пример 4. Решение задачи о Ханойских башнях (слайды 16-20)
- Вопросы и задания
- Источники



# Рекурсивные объекты

- Если поставить два зеркала напротив друг друга и между ними поместить предмет, то получится бесконечное множество изображений, причем каждое из них содержит свое собственное.
- Любое из этих изображений можно рассматривать как рекурсивный объект, который частично состоит или определяется с помощью самого себя.
- **Рекурсивные объекты обладают несколькими свойствами:**
  - *простотой построения;*
  - *несхожестью конечного результата с начальными данными;*
  - *внутренним самоподобием.*



# Рекурсивное определение

- В математике встречаются рекурсивные определения, позволяющие описать объекты через самих себя.
- К таким определениям относится, *например*, **определение натурального числа**:
  - 1) единица есть натуральное число;
  - 2) число, следующее за натуральным (т.е. больше его на единицу), есть натуральное число.
- Определение, которое задает некоторый объект в терминах более простого случая этого же объекта, называется **рекурсивным определением**.



# Рекурсия

- **Мощность рекурсивного определения** заключается в том, что оно позволяет с помощью конечного высказывания определить бесконечное множество объектов.
- Как и цикл, рекурсивное определение содержит повторения, но каждый раз при этом используются новые данные, т. е. повторения не являются явными.
- **Рекурсия — это способ описания функций или процессов через самих себя.**



# Рекурсивный алгоритм

- Процесс может быть описан некоторым **алгоритмом**, называемым в данном случае **рекурсивным**.
- В таких алгоритмах выделяется два этапа выполнения:
  - 1) **«погружение»** алгоритма в себя, т. е. применение определения **«в обратную сторону»**, пока не будет найдено начальное определение, не являющееся рекурсивным;
  - 2) последовательное построение **от начального определения до определения с введенным в алгоритм значением**.
- Рассмотрим примеры рекурсивных алгоритмов, часто оформляемых в виде процедур и функций.



# Пример 1. Определение факториала

- Наиболее распространенным рекурсивным определением является **определение факториала** (нерекурсивное вычисление факториала приведено в примере E9):

$$(a) \ 1! = 1,$$

$$(b) \ n > 1, n := n * (n - 1)!$$

- На основе этого определения можно записать программу вычисления факториала, использующую **рекурсивную функцию**.





```
program E25;  
var n, y: integer;  
function F (x: integer): integer; {описание рекурсивной  
функции}  
begin  
    if x = 1  
        then F := 1 {вызов для начального определения}  
        else F := x * F (x - 1) {вызов для предыдущего  
        определения}  
end; {конец описания функции}  
begin {начало главной программы}  
    readln (n);  
    Y := F (n); {вызов функции в главной программе}  
    write (n, '! = ', Y)  
end. {конец главной программы}
```



## Выполним программу E25 для $n=4$ .

- Выполним программу **E25** для  $n=4$ .
- Рекурсивная функция будет работать следующим образом (при вызове функции значение  $n$  присваивается переменной  $x$ ).
- Сначала осуществляется «погружение», работает оператор ветви **else** условного оператора:

1-й шаг:  $x = 4, x - 1 = 3,$

выполняется промежуточное вычисление  $4! = 4 * 3!$

2-й шаг:  $x = 3, x - 1 = 2,$

выполняется промежуточное вычисление  $3! = 3 * 2!$

3-й шаг:  $x = 2, x - 1 = 1,$

выполняется промежуточное вычисление  $2! = 2 * 1!$

4-й шаг (последний):  $1! = 1$  по начальному определению, работает оператор **F: = 1** ветви **then** условного оператора.



## Следующий этап выполнения рекурсивного алгоритма

- Следующий этап выполнения рекурсивного алгоритма — **построение «прямого» определения**, от начального до получения результата с исходными для алгоритма данными (числом 4). При этом осуществляется подстановка предыдущих вычислений (более поздних шагов) в более ранние:
  - 5-й шаг:  $2! = 2 * 1 = 2$
  - 6-й шаг:  $3! = 3 * 2 = 6$
  - 7-й шаг:  $4! = 4 * 6 = 24$  — получен результат, он возвращается в плавную программу и присваивается переменной Y.



## Пример 2. Вычисление степени с натуральным показателем

- Вычисление степени с натуральным показателем можно определить рекурсивно:

(а)  $x^0 = 1$

(б)  $k > 0: x^k = x * x^{k-1}$

- Этому определению соответствует рекурсивная функция **power(k,x)**. Программа имеет вид:

```
program E26;
var y: real; n: integer;
function power (k: integer; x: real): real; {описание рекурсив-
ной функции}
begin
  if k = 0
  then power := 1 {начальное определение}
  else power := x * power (k - 1, x) {рекурсивное определение}
end; {конец описания функции}
begin {начало главной программы}
  write (' основание степени x = ');
  readln (y);
  write ('показатель степени k = ');
  readln (n);
  write ('x в степени k ', power(n, y)) {вызов функции и печать
результата}
end. {конец главной программы}
```



## Пример 3. Вычисление чисел Фибоначчи

Вычисление чисел Фибоначчи.

- Итальянский математик Фибоначчи придумал последовательность натуральных чисел: 1, 1, 2, 3, 5, 8, 13, ...  
**. Первые два члена последовательности равны единице, а каждый, начиная с третьего, равен сумме двух предыдущих. Для чисел Фибоначчи верно соотношение:**

$$F_k = F_{k-1} + F_{k-2}$$

- Рекурсивная функция получения значения n-го числа Фибоначчи имеет вид:

```
function Fib (n: integer): integer;  
begin  
  if k < 3  
    then Fib: = 1  
    else Fib: = Fib (n - 1) + Fib (n - 2)  
end;
```



# Для чисел Фибоначчи используется следующее рекурсивное определение

- Для чисел Фибоначчи используется следующее рекурсивное определение:
  - (a)  $n = 1, n = 2: \text{fib}(n) = 1$
  - (b)  $n > 2: \text{fib}(n) = \text{fib}(n - 2) + \text{fib}(n - 1)$
- Для того чтобы определить **fib(6)**, применяя данное рекурсивное определение, надо вычислить:

$$\begin{aligned}\text{fib}(6) &= \text{fib}(4) + \text{fib}(5) = \text{fib}(2) + \text{fib}(3) + \text{fib}(5) = \\ &= 1 + \text{fib}(3) + \text{fib}(5) = \\ &= 1 + \text{fib}(1) + \text{fib}(2) + \text{fib}(5) = \\ &= 1 + 1 + 1 + \text{fib}(5) = \\ &= 3 + \text{fib}(3) + \text{fib}(4) = \\ &= 3 + \text{fib}(1) + \text{fib}(2) + \text{fib}(4) = \\ &= 3 + 1 + 1 + \text{fib}(4) = \\ &= 5 + \text{fib}(2) + \text{fib}(3) = \\ &= 5 + 1 + \text{fib}(1) + \text{fib}(2) = 6 + 1 + 1 = 8\end{aligned}$$



- Количество действий в данных вычислениях с использованием рекурсивного определения чисел Фибоначчи резко возрастает, потому что это определение ссылается само на себя дважды.
- При вычислении факториала количество действий при выполнении программы с рекурсивной функцией и примера E9 одинаково.



## Пример 4. Решение задачи о Ханойских башнях

- Рекурсивные алгоритмы могут быть оформлены и в виде процедур.
- Примером такой процедуры является решение задачи о Ханойских башнях.
- Эта задача связана с легендой о том, что **в одном из восточных храмов находится бронзовая плита с тремя алмазными стержнями. На один из них при сотворении мира нанизали 64 диска из чистого золота так, как показано на рисунке 36. Жрецы должны переносить диски с одного стержня на другой, следуя следующим законам:**
  - диски можно перемещать только по одному;
  - нельзя класть больший диск на меньший.
- **Согласно легенде, когда все диски будут перенесены с одного стержня на другой, наступит конец света.**



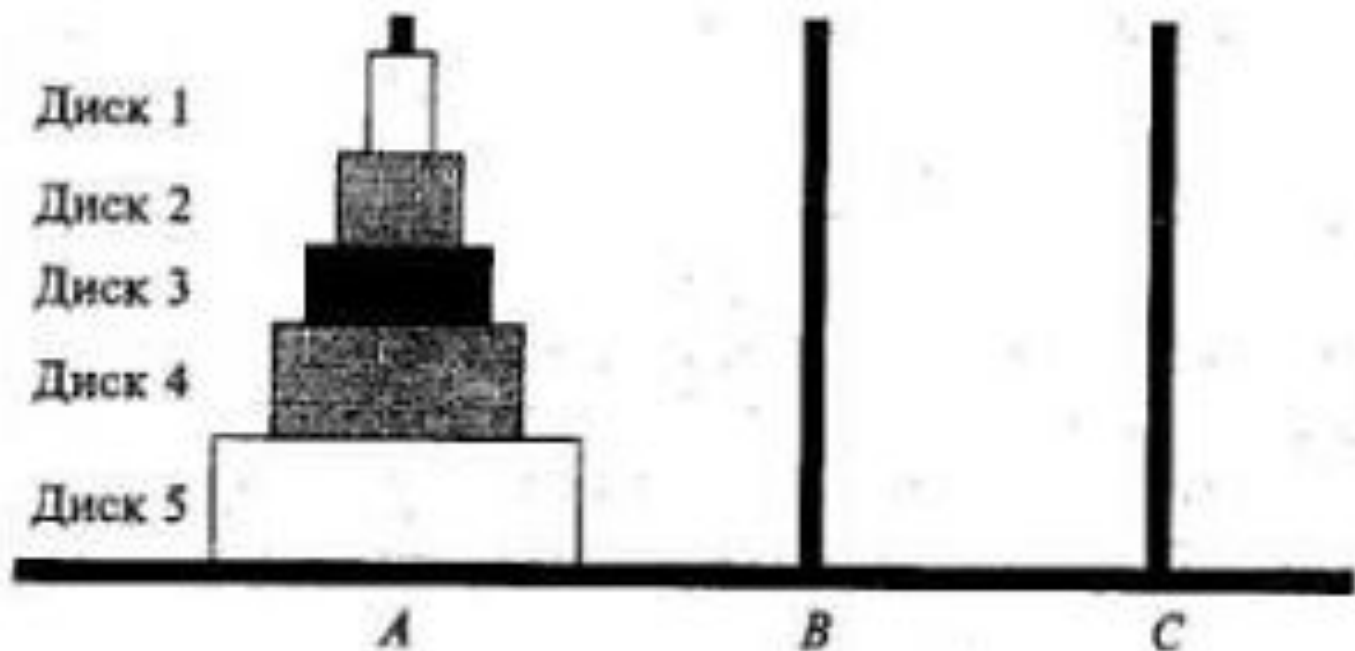


Рис. 36. Ханойские башни



# Решение этой задачи реализовано в виде рекурсивного алгоритма

- Решение этой задачи реализовано в виде рекурсивного алгоритма, который представляет собой инструкцию по перемещению дисков.
- Сформулируем задачу, присвоив имена стержням (**A, B, C**) и номера дискам (**от 1 до n**).
- Надо перенести диски со стержня **A** на стержень **C**, используя **B** как вспомогательный и следуя приведенным выше правилам переноса дисков.
- **Алгоритм на естественном языке имеет вид:**
  - 1) если  $n = 0$ , остановиться;
  - 2) переместить верхние  $n - 1$  дисков со стержня **A** на стержень **B**, используя стержень **C** как вспомогательный;
  - 3) переместить оставшийся диск со стержня **A** на стержень **C**;
  - 4) переместить  $n - 1$  дисков со стержня **B** на стержень **C**, используя стержень **A** как вспомогательный.
- В процедуре появляется новый тип данных — **char**, значение этого типа — **один символ**, заключенный в апострофы.



## Программа имеет вид:

```
program E27;  
var k: integer;  
procedure Hanoi (n: integer; One, Two, Three: char);  
begin  
  if n > 0 then  
    begin  
      Hanoi (n - 1, One, Three, Two);  
      write (' переместить диск', n, ' со стержня ', One,  
        'на стержень ', Two);  
      Hanoi (n - 1, Two, One, Three)  
    end;  
  end;  
begin  
  write ('введите количество дисков');  
  readln (k);  
  Hanoi (k, 'A', 'B', 'C')  
end.
```



# Результат работы программы для $n=3$

- Результат работы программы для  $n=3$  — это инструкция из 7 пунктов ( $n=4$  — инструкция из 15 пунктов):
  - переместить диск 1 со стержня А на стержень С
  - переместить диск 2 со стержня А на стержень В
  - переместить диск 1 со стержня С на стержень В
  - переместить диск 3 со стержня А на стержень С
  - переместить диск 1 со стержня В на стержень А
  - переместить диск 2 со стержня В на стержень С
  - переместить диск 1 со стержня А на стержень С



# Вопросы и задания

1. Что такое рекурсивный объект и каковы его свойства?
2. Приведите примеры рекурсивного определения в математике.
3. Что такое рекурсия?
4. Как выполняется рекурсивный алгоритм?
5. Поясните выполнения рекурсивной функции вычисления степени с натуральным показателем.
6. Напишите главную программу для вычисления  $n$ -го числа Фибоначчи.
7. Почему использовать рекурсивный алгоритм вычисления  $n$ -го числа Фибоначчи невыгодно?
8. Определите рекурсивно умножение как сложение и деление как вычитание и оформите алгоритмы в виде рекурсивных функций с вызовом из главных программ.



# Литература

- **А.А.Кузнецов, Н.В.Ипатова**  
«Основы информатики», 8-9 кл.:
  - Раздел 3. ОСНОВЫ ПРОГРАММИРОВАНИЯ,  
С.130-135