

# Основы программирования

## Рекурсивные алгоритмы

# Пример: вычисление факториала

Эквивалентные рекуррентные соотношения для  $n!$ :

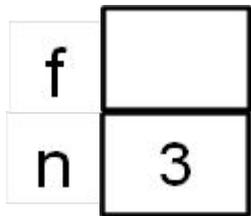
$$\begin{cases} F_0 = 1, \\ F_i = F_{i-1} * i, \quad i = 1, 2, \dots, n. \end{cases} \quad \begin{cases} 0! = 1, \\ n! = n \cdot (n - 1)! \quad n = 1, 2, \dots \end{cases}$$

можно использовать для построения не только рекуррентного (на основе цикла), но и рекурсивного алгоритма (вычислить  $n!$  через  $(n-1)!$  и т.д.).

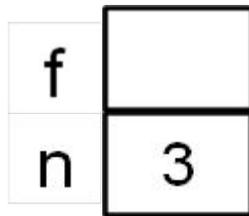
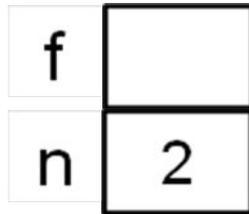
```
int fact(int n)
{
    if (n <= 0) return 1;
    else return fact(n-1) * n;
}
void main(...)
{
    cout << fact(3) ;
}
```

# Стек при рекурсивном спуске

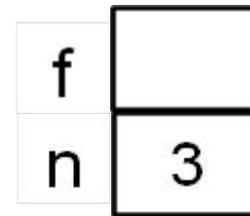
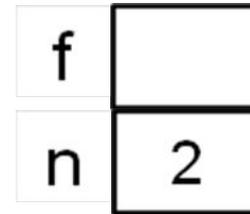
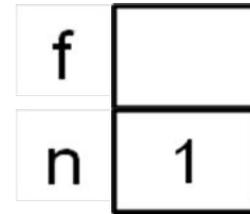
Последовательные вызовы рекурсивной функции с параметром **n** и возвращаемым значением **f** (условное имя)



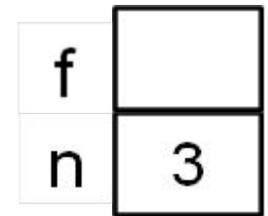
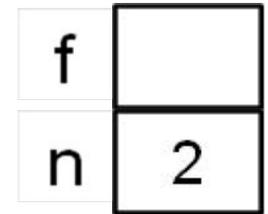
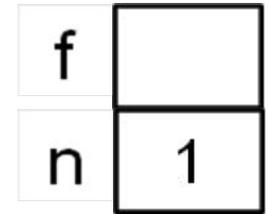
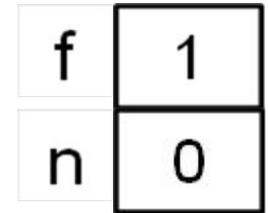
**fact (3)**



**fact (2)**



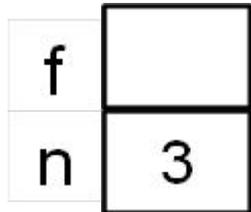
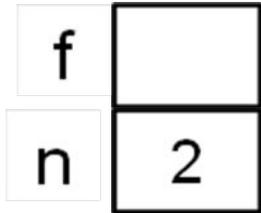
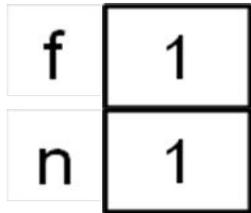
**fact (1)**



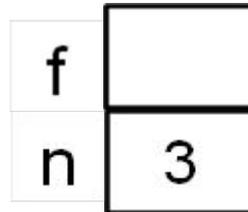
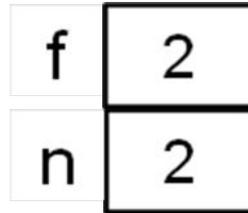
**fact (0)**

# Стек при рекурсивном подъеме

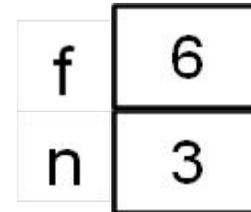
Состояние перед освобождением стека и возвратом в вызывающую функцию с передачей возвращаемого значения **f**



**fact (1)**



**fact (2)**



**fact (3)**

# Метод математической индукции

3 основных этапа метода математической индукции:

1) **базис**, 2) **предположение**, 3) **индуктивный вывод**.

На этапе **базиса** проверяют, что доказываемое утверждение  $P(n)$  относительно параметра индукции  $n$  справедливо при  $n = n_0$ .

На втором этапе делается **предположение**, что утверждение  $P(n)$  справедливо для всех значений  $n$ , **не бóльших**, чем некоторое  $k$ ,  $k \geq n \geq n_0$ .

На этапе **индуктивного вывода** доказываемое утверждение  $P(n)$  будет справедливо при  $n = k + 1 > n_0$ . Из этого следует, что утверждение  $P(n)$  справедливо для **любого**  $n \geq n_0$ .

# Задача «Ханойские башни»

Имеется три колышка: **a**, **b**, **c**. На колышке **a** расположено **n** дисков в виде башни.

**Задача:** Переложить всю башню на колышек **c**, перекладывая по одному диску так, чтобы любой диск большего размера **не лежал** на меньшем диске.

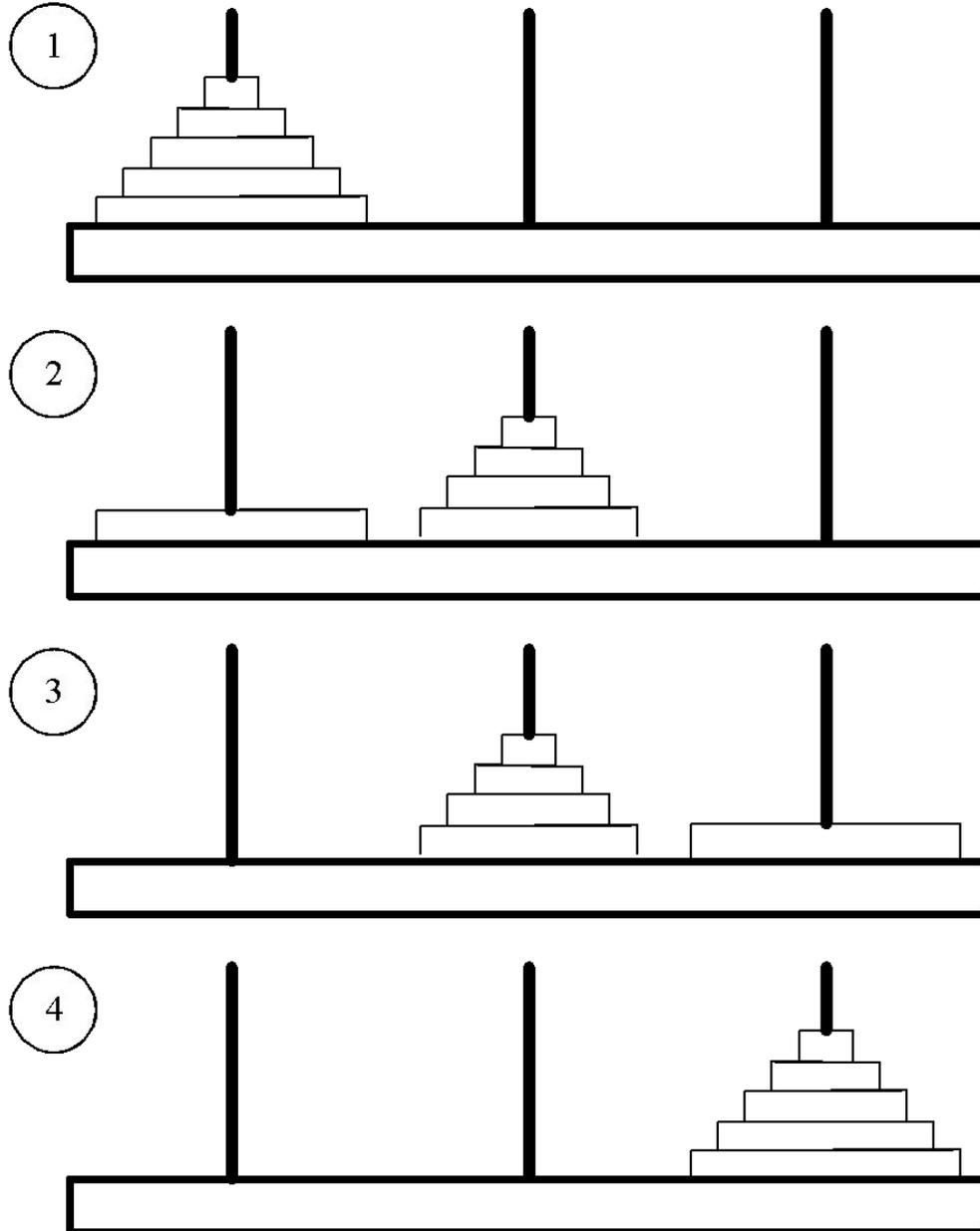
Решение на основе математической индукции:

**Базис.** Число дисков  $n = 1$ . Переложить диск с колышка **a** на колышек **c**.

**Предположение.** Пусть алгоритм умеет перекладывать башни из  $n = k \geq 1$ .

**Индукция.** Пусть  $n = k+1 \geq 2$ . Перекладываем башню из **k** дисков с колышка **a** на колышек **b**, затем один диск с колышка **a** на колышек **c** и, наконец, башню из **k** дисков с колышка **b** на колышек **c**.

# Иллюстрация для шага индукции



# Рекурсивная функция

```
void hanoi(int n, int a, int b, int c)
{
    if (n == 1)
        cout << a << "->" << c << endl;
    else
    {
        hanoi(n-1, a, c, b);
        cout << a << "->" << c << endl;
        hanoi(n-1, b, a, c);
    }
}

void main(...)
{
    hanoi(6, 1, 2, 3); // 6 дисков с 1 на 3
}
```

# Рекуррентное соотношение для трудоемкости

$$H(n) = \begin{cases} 1, & n = 1, \\ 2H(n-1) + 1, & n = 2, 3, \dots, \end{cases}$$

из которого получаем:

$$H(n) = 2H(n-1) + 1 = 2^2H(n-2) + 2 + 1 = \\ 2^{n-1} + 2^{n-2} + \dots + 2 + 1 = 2^n - 1.$$

Глубина рекурсии:  $n$ .

Если на перекладывание одного диска тратить 1 секунду, то при  $n = 64$  всю работу можно завершить за  $2^{64}$  сек  $\approx 580$  млрд. лет.

# Рекурсивное вычисление чисел Фибоначчи

```
int fib(int n)
{
    if (!n) return 0;
    else if (n == 1) return 1;
    else return fib(n-1) + fib(n-2);
}
```

Количество рекурсивных вызовов  $R_n$  и их связь с числами Фибоначчи  $F_n$ :

$$\begin{cases} R_0 = 1, R_1 = 1, \\ R_n = R_{n-1} + R_{n-2} + 1, \quad n = 2, 3, \dots \end{cases}$$

$$R_n : 1, 1, 3, 5, 9, 15, 25, 41, 67, \dots$$

$$F_n : 0, 1, 1, 2, 3, 5, 8, 13, 21, \dots$$

$$R_n = 2F_{n+1} - 1$$

# Рекурсивный алгоритм с массивом

```
int F[50];  
int fib(int n)  
{  
    if (!n) return F[0] = 0;  
    else if (n == 1) return F[1] = 1;  
    else if (F[n] > 0) return F[n];  
    else return F[n] = fib(n-1) + fib(n-2);  
}  
void main(...)  
{  
    int i, n; cin >> n;  
    for (i = 0; i < n; i++) F[i] = 0;  
    cout << fib(n);  
}
```