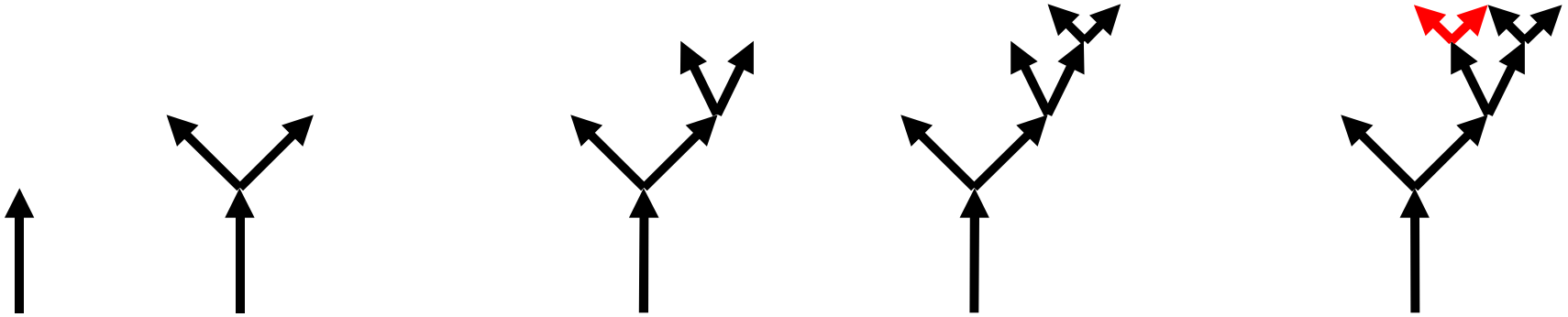
ТЕМА 2.

Рекурсивные алгоритмы

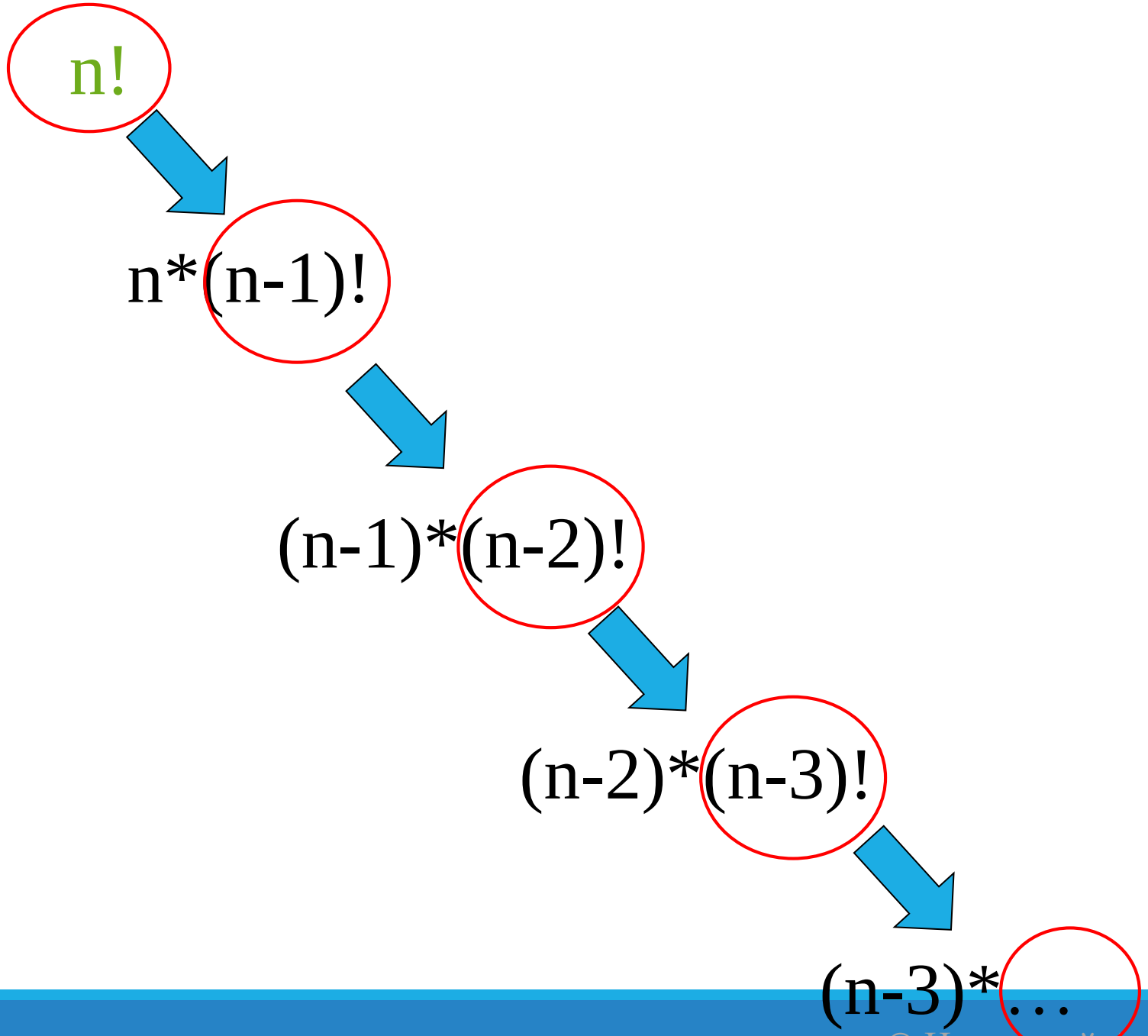
2.1. Понятие рекурсии



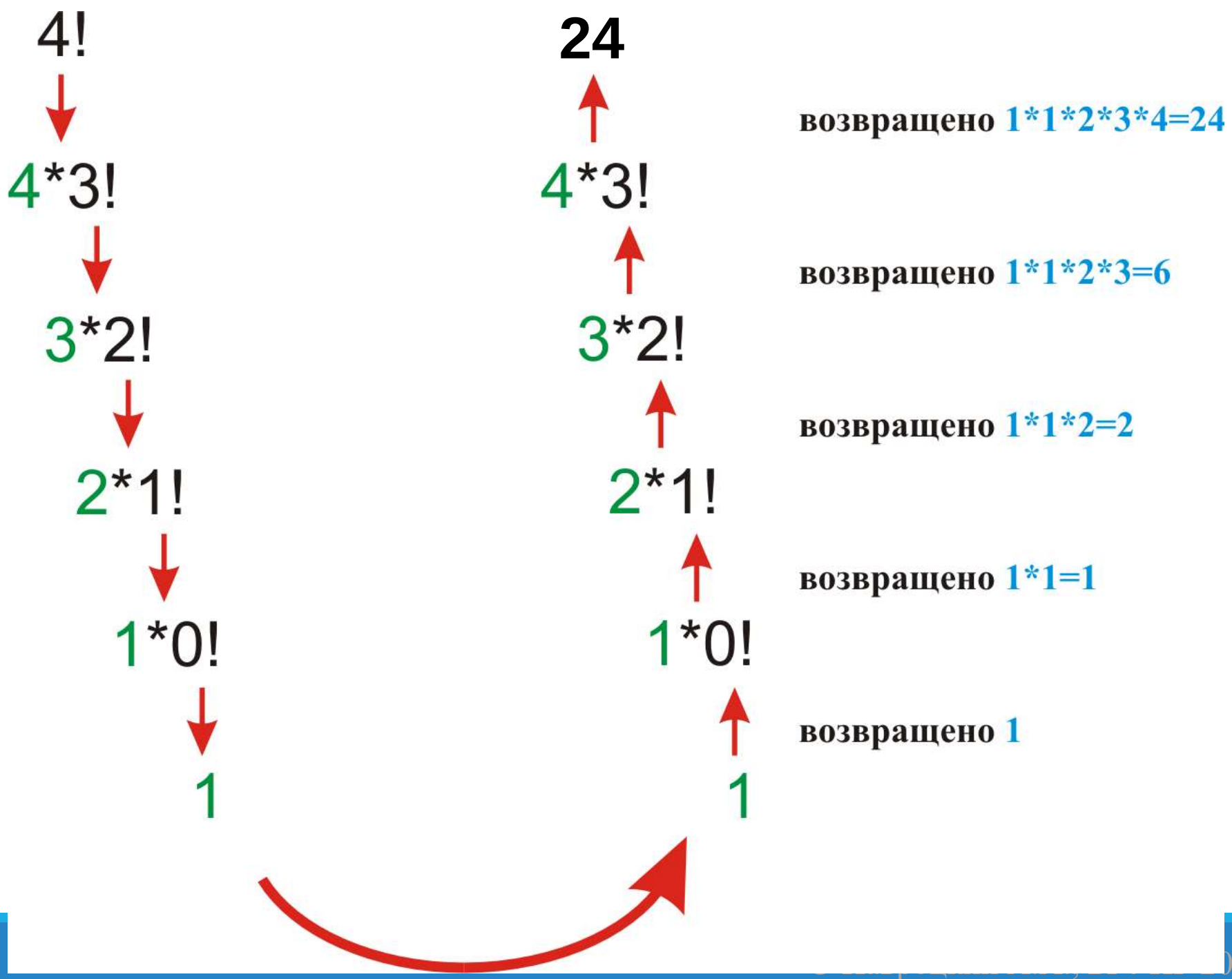
n!

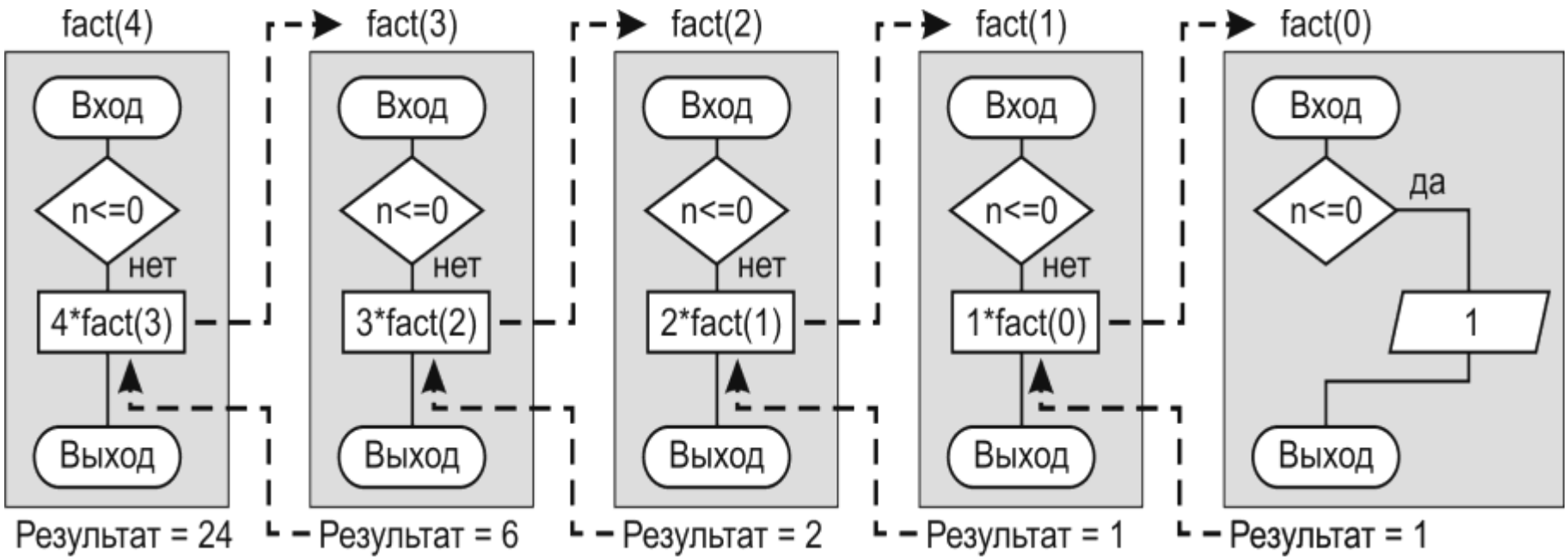
$0!=1$

$n!=n*(n-1)!$



```
int fact(int n)
{
        if (n<=0) return 1;
                else return n*fact(n-1);
}
```


4!
4*3!
3*2!
2*1!
1*0!
1
24
4*3!
3*2!
2*1!
1*0!
1
возвращено 1*1*2*3*4=24
возвращено 1*1*2*3=6
возвращено 1*1*2=2
возвращено 1*1=1
возвращено 1



2.2. Условие окончания рекурсивного алгоритма

```
int fact(int n)
{
if (n == 0) return 1;
else return n * fact(n - 1);
}
```

2.3. Целесообразность использования рекурсии

2.4. Примеры рекурсивных алгоритмов

Пример 1.1. Найти сумму $S_n = \sum_{i=1}^{n} a_i$

Рекурсивный алгоритм:

```
int sumr(int i)
{
if (i < 0) return 0;
else return a[i] + sumr(i - 1);
}
```

Нерекурсивный алгоритм:

```
int sum()
{
int s = 0;
for (int i = 0; i < n; i++)  s += a[i];
return s;
}
```

Пример 1.2.

Найти наибольший общий делитель двух чисел.

Имеется соотношение: если B делится на A нацело, то НОД(A, B)=A;
иначе НОД(A, B)= НОД(B %A, A).

```
int nodr(int a, int b)
{
if (b % a == 0) return a;
else return nodr(b % a, a);
}
```

Пример 1.3. Найти $\max(a_1 \ldots a_n)$.

```
int maxr2(int i)
{
if (i == 0) return a[0];
else {
int mx = maxr2(i - 1);
if (a[i] > mx)  return a[i];
else return mx;
}
}
```

Пример 1.4. **Вычисление чисел Фибоначчи, которые определяются Следующим рекурсивным соотношением:**

а) $b_0=0;\ b_1=1;$

б) $b_n=b_{n-1}+b_{n-2}.$

```
int fibri(int x1, int x2, int n)
{
if (n == 1) return x2;
else  if (n == 0) return x1;
else {
x2 += x1;
x1 = x2 - x1;
return fibri(x1, x2, n - 1);
}
}
```

Обращение к функции: fibri(0,1,n);

Пример 1.5. Найти значение функции Аккермана $A(m, n)$, которая определяется для всех неотрицательных целых аргументов m и n следующим образом:

$A(0, n) = n+1$, если m=0;

$A(m,0) = A(m - 1, 1)$, если n=0;

$A(m, n) = A(m - 1, A(m, n - 1))$ если

и $m > 0$ и $n > 0$.

```
int akkr(int m, int n)
{
if (m == 0) return (n + 1);
  else
  if (n == 0) return akkr(m - 1, 1);
     else  return akkr(m - 1, akkr(m, n - 1));
}
```

Пример 1.6. Вычислить значение $x=\sqrt{a}$,

используя формулу $x_n = \frac{1}{2}\left(x_{n-1} + a/x_{n-1}\right)$,

в качестве начального приближения использовать значение $x_0 = (1+a)/2$.

```
double sqrtR(double n)
{
        if (n<=0) return (1+a)/2;
                else {
                  double r = sqrtR(n-1);
                  return 0.5*(r+a/r);
                     }
}
```

Пример 1.7. Найти максимальный элемент в массиве $a_1, ..., a_n$, используя метод деления пополам

$$\max(a_1, ..., a_n) =$$

$$=\max(\max(a_1, ..., a_{n/2}), \max(a_{n/2+1}, ..., a_n)).$$

```
double maxr(int i, int j)
{
        if (i==j)  return a[i];
        else
            {
                int n=(i+j)/2;
               double r=maxr(i, n);
               double l=maxr(n+1, j);
                return (r>l)?r:l;
             }
}
```

Пример 1.8. Задача о Ханойской башне.

Даны три стержня, на один из которых нанизаны n колец. Кольца отличаются размером и лежат меньшее на большем. Задача состоит в том, чтобы перенести башню на другой стержень. За один раз разрешается переносить только одно кольцо, причём нельзя класть большее кольцо, на меньшее. Для промежуточного хранения дисков можно использовать третий стержень.

Решение задачи для одного диска:

– переложить диск с *1* стержня на *1* стержень.

Решение задачи для двух дисков:

- переложить диск с *1* стержня на *3* стержень;
- переложить диск с *1* стержня на *2* стержень;
- переложить диск с *3* стержня на *1* стержень;

Башня из n дисков представляется как башня из 2-х – один нижний и верхний, состоящий из n-1 дисков.

```
void hanoy(int n, int sterg1, int sterg2,
int sterg3)
{
        if (n>0) {
        hanoy(n-1,sterg1,sterg3,sterg2);
  cout << "perenesty disk s "<<sterg1<<"
na "<<sterg2<<endl;
        hanoy(n-1,sterg3,sterg2,sterg1);
                }
}
```

hanr(3,1,2,3);

perenesty disk s **1** na **2**
perenesty disk s **1** na **3**
perenesty disk s **2** na **3**
perenesty disk s **1** na **2**
perenesty disk s **3** na **1**
perenesty disk s **3** na **2**
perenesty disk s **1** na **2**