Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

**ЛАБОРАТОРНАЯ РАБОТА**

«**Ханойская башня**»

**Выполнил:**

студент группы РИС-23-2б

Борисов Никита Андреевич

**Проверила:**

доцент кафедры ИТАС

О.А. Полякова

Пермь, 2024 г.

**Задача:**

Есть три стержня. Количество колец на первом из них задаёт пользователь. Программа должна переместить все кольца на последний стержень. За раз можно перенести только одно кольцо, при этом кольцо большего размера нельзя положить на кольцо меньшего размера. Программа обязательно должна работать рекурсивно.

**Анализ:**

В теле рекурсивной функции надо создать условие окончания создания ветви рекурсии, а именно такое, что ветвь будет строиться, пока количество колец на данном уровне, поступающее в тело функции не будет равно нулю. Программа будет показывать свои действия, выводя номера башен, откуда берёт и куда кладёт.

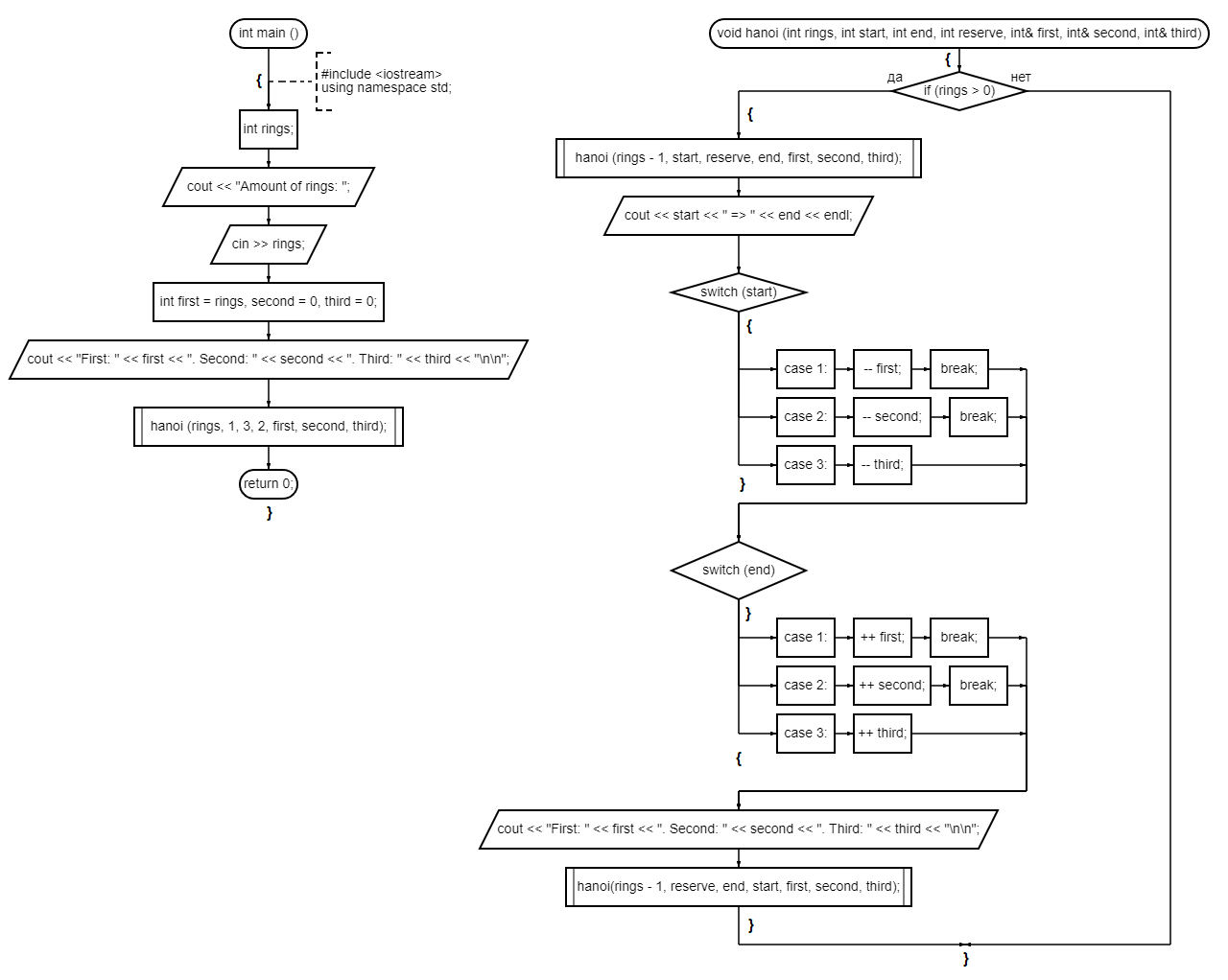
**Решение:**

Опишем функцию *hanoi*. У неё будут целочисленные параметры:

* *rings* – количество колец;
* *start*, *reserve* и *end* – переменные, содержащие и передающие номера стержней.

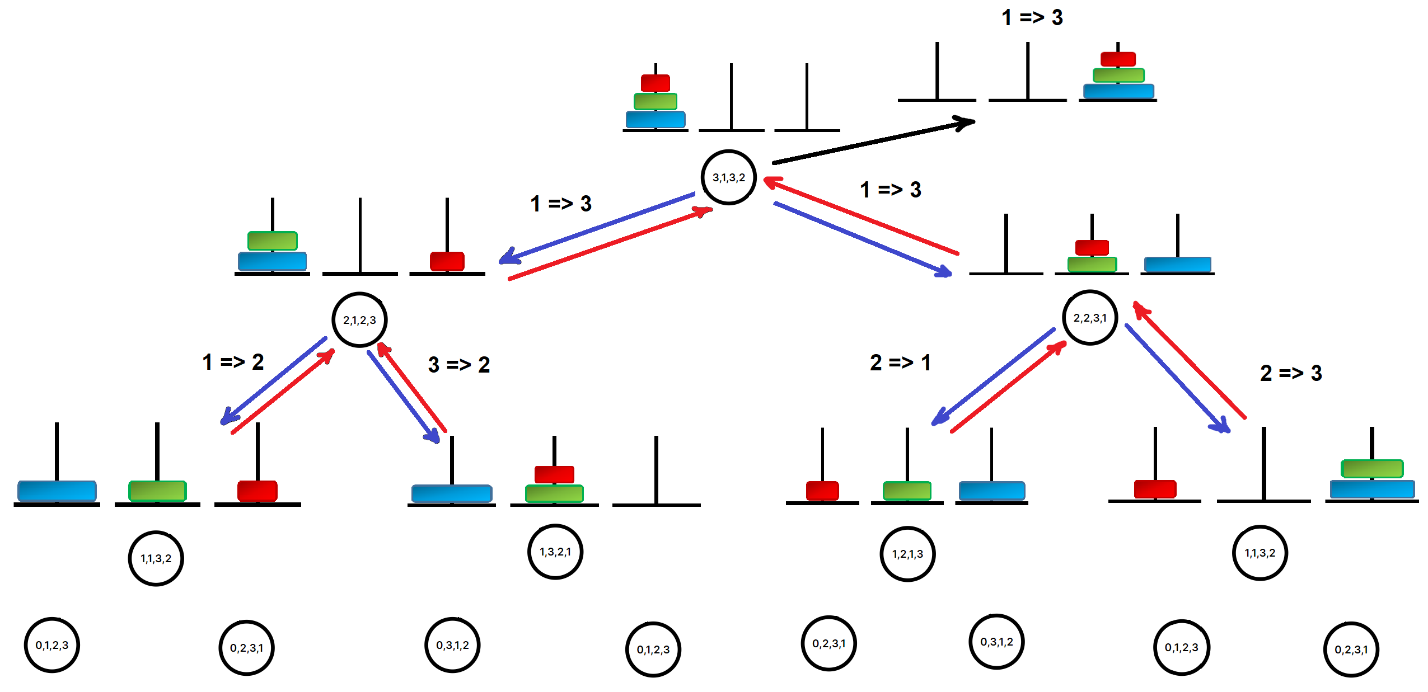
Внутри тела функции создаём условие выхода из рекурсии: *rings > 0*. Далее основная идея решения состоит в перемене мест значений, передаваемых в соответствующие по счёту параметры функции. Внутри функции сделаем два ветвления рекурсии. Вызовы нужны, чтобы правильно опередить, куда следует переместить кольцо в зависимости от чётности или нечётности колец на данном этапе рекурсии.

**Моделирование:**



**Визуализация работы рекурсии:**

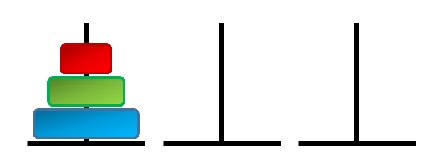
Рассмотрим, как поведёт себя рекурсия при вводе трёх колец.

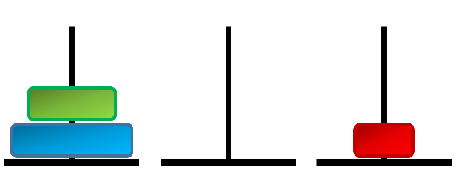


**Работа по алгоритму:**

1. **Вызов функции - H (3, 1, 3):**
2. Вызов функции по n-1 - H (2, 1, 3), t = 2:

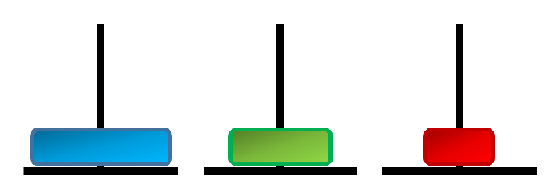
* Вызов по n-1 - H (1, 1, 2) t = 3:





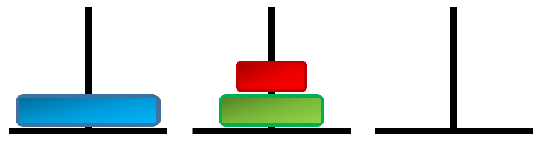
* 1 => 3.

1. 1 => 2;

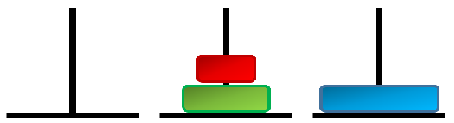


1. Вызов по n-1 - H (1, 3, 1) t = 2:

* 3 =>1;

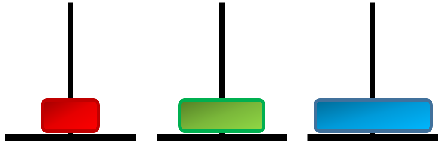


1. **Вывод: 1 => 3.**

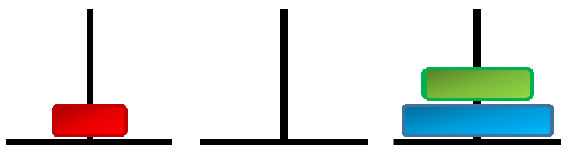


1. **Вызов по n-1 - H (1, 2, 1) t = 3:**
2. Вызов по n-1 - H (1, 2, 3) t = 1:

* 2 => 1;

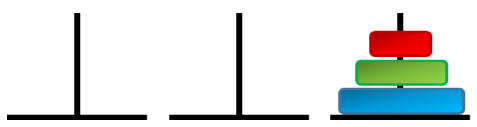


1. 2 => 3;

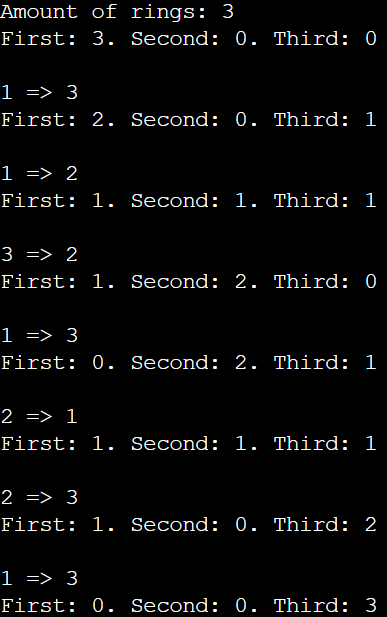


1. Вызов по n-1 - H (1, 1, 2) t = 3:

* 1 => 3;



**Результат работы программы:**



**Код программы:**

*visualization\_of\_recursion\_for\_hanoi\_tower.cpp*

#include <iostream>

using namespace std;

void hanoi (int rings, int start, int end, int reserve, int& first, int& second, int& third)

{

if (rings > 0)

{

hanoi (rings - 1, start, reserve, end, first, second, third);

cout << start << " => " << end << endl;

switch (start)

{

case 1: -- first; break;

case 2: -- second; break;

case 3: -- third;

}

switch (end)

{

case 1: ++ first; break;

case 2: ++ second; break;

case 3: ++ third;

}

cout << "First: " << first << ". Second: " << second << ". Third: " << third << "\n\n";

hanoi(rings - 1, reserve, end, start, first, second, third);

}

}

int main ()

{

int rings;

cout << "Amount of rings: ";

cin >> rings;

int first = rings, second = 0, third = 0;

cout << "First: " << first << ". Second: " << second << ". Third: " << third << "\n\n";

hanoi (rings, 1, 3, 2, first, second, third);

return 0;

}

**Ссылка на Git:** <https://github.com/Exateym/Study>