Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Лабораторная работа

«Рекурсия»

Выполнил:

студент первого курса

ЭТФ группы РИС-23-3б

Акбашева Софья Руслановна

Проверила:

Доцент кафедры ИТАС О. А. Полякова

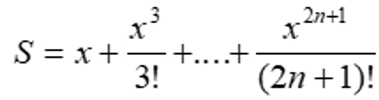
2024

# Сумма функционального ряда

Вариант №25

**Цель**: научиться решать задачи с использованием рекурсивных функций.

**Постановка задачи:** приближенно вычислить значение функции двумя способами.



1) Через сумму функционального ряда с помощью рекурсивной функции. Аргументы функции – n и x, где n – количество членов ряда, x – переменная.

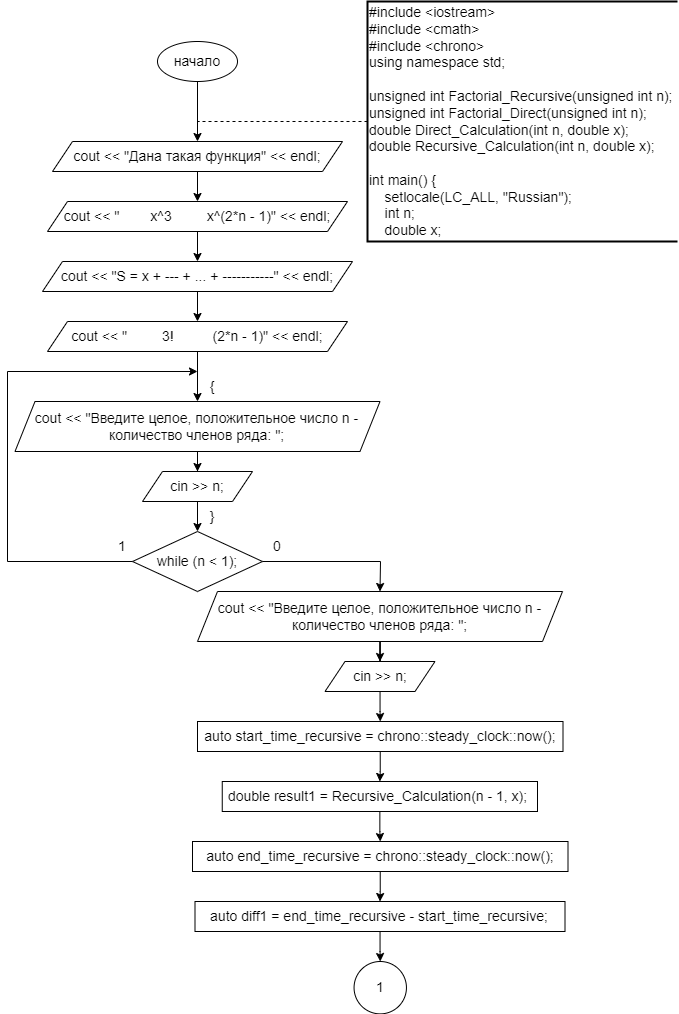
2) Через прямое вычисление значения функции. Аргумент функции - переменная x.

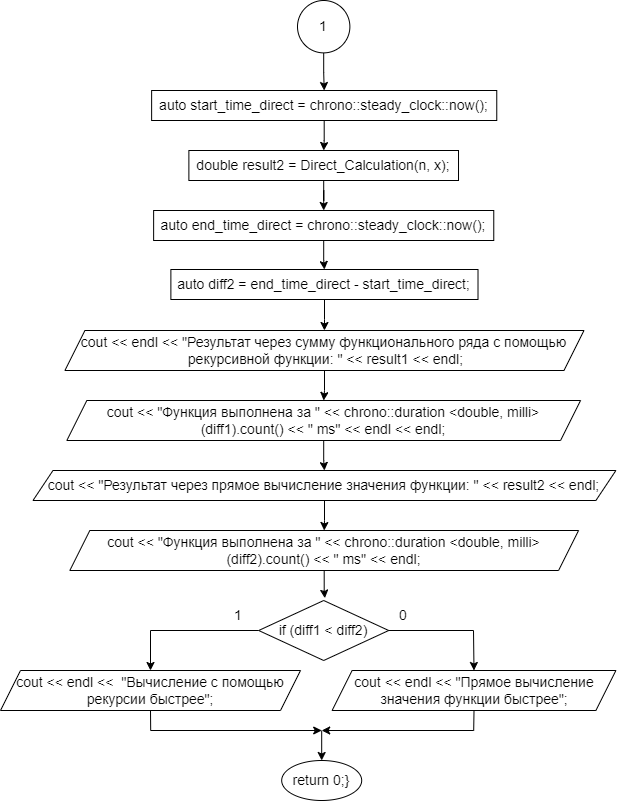
Сравнить полученные результаты.

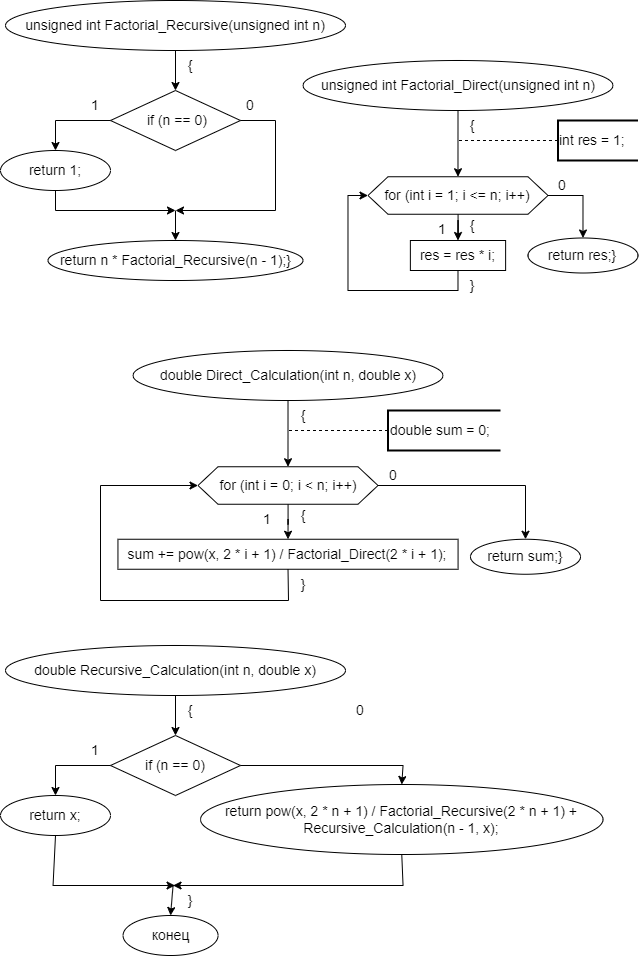
## Анализ задачи

1. Пользователь вводит число n - количество членов ряда.
2. Пользователь вводит число x – переменную.
3. Рекурсивная функция возвращает саму себя, прибавляя новый член функционального ряда.
4. Функция рекурсивного вычисления факториала возвращает текущее значение переменной, умноженное на эту функцию от переменной меньшей текущей на 1.
5. Функция прямого вычисления функционального ряда содержит арифметический цикл.
6. Функция прямого вычисления факториала содержит арифметический цикл.
7. Для измерения работы функции необходимо подключение библиотеки chrono.

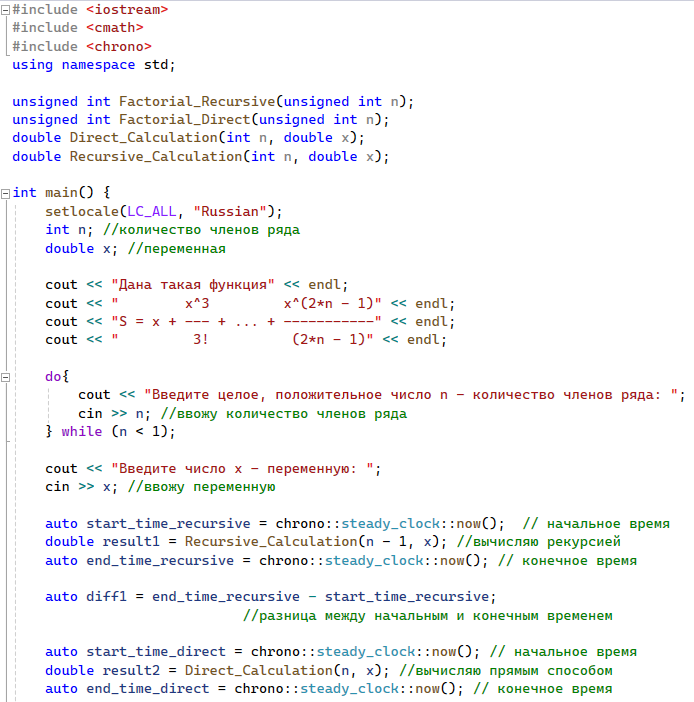
## Блок схема

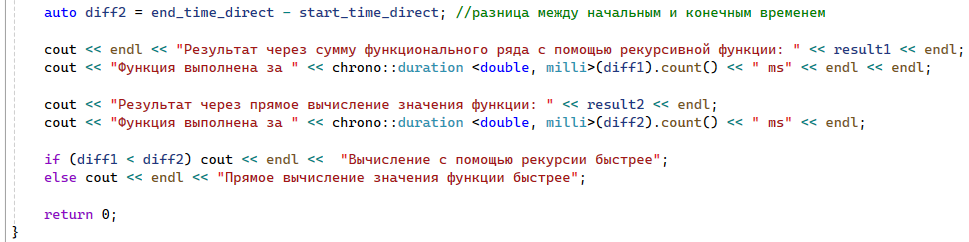
****

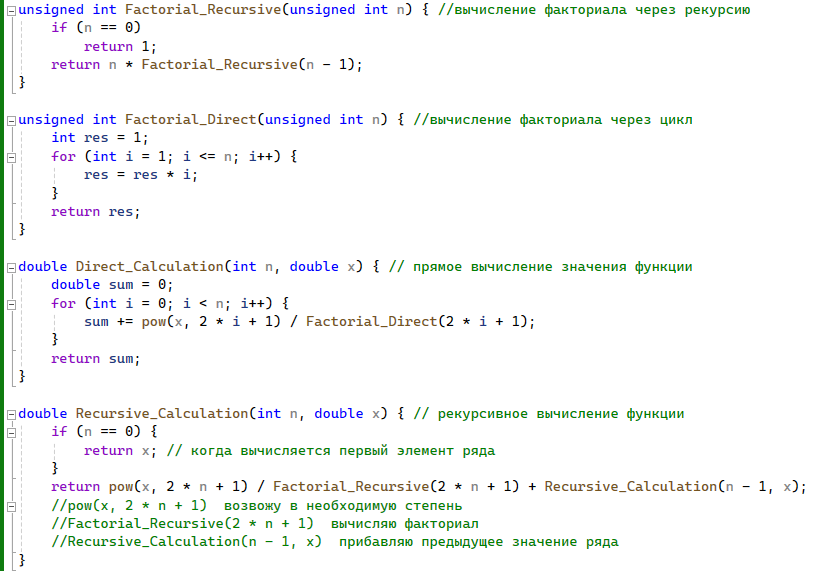
****

****

## Код программы

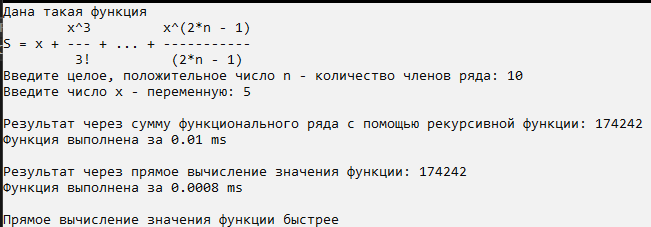
****

****

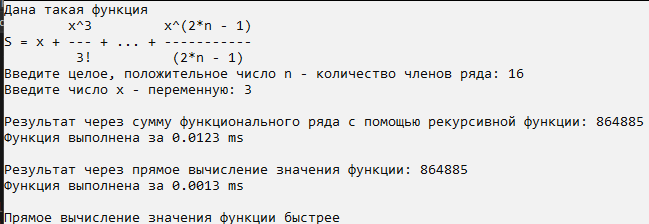
****

## Результат работы программы при разных входных данных

1. Пусть n (количество членов) равно 10, а х (переменная) равна 5



1. Пусть n (количество членов) равно 16, а х (переменная) равна 3



Вариант с циклом быстрее рекурсии, так как операций столько же, сколько и в рекурсивной функции, но нет дополнительных вычислительных затрат на организацию вложенных вызовов. Поэтому рекурсия в данном случае работает медленнее. Однако обе функции дают верные ответы.

## Вывод

В ходе работы я применила знания о работе с рекурсивными функциями. Также, я использовала библиотеку chrono, для измерения времени работы функций (рекурсивной и не рекурсивной), для последующего сравнения их время затратности. В результате, мне удалось реализовать поставленную задачу и я выяснила, что не рекурсивная функция работает быстрее (поскольку вызовы методов потребляют больше ресурсов, чем исполнение обычных операторов).

# Числа Фибоначчи

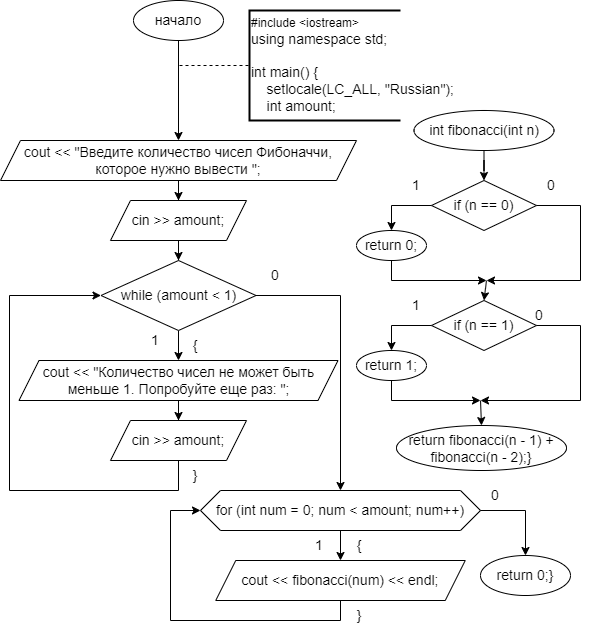
**Цель**: научиться решать задачи с использованием рекурсивных функций.

**Постановка задачи:** реализовать рекурсивную функцию для вычисления n-го числа Фибоначчи.

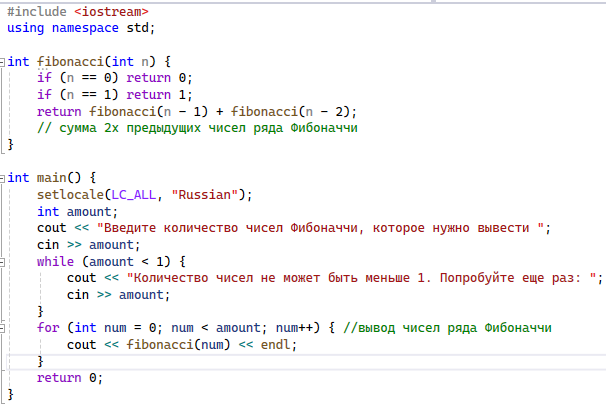
## Анализ задачи

1. Каждый элемент ряда представляет собой сумму двух предыдущих чисел (кроме первого и второго числа – первое число равно 0, а второе число равно 1).
2. Функция fibonacci() вычисляет n-ое число Фибоначчи. Если в функцию передано значение 0, возвращается 0; если передано 1 – возвращается 1. Иначе возвращается сумма двух предыдущих чисел.
3. Пользователь вводит значение количества чисел Фибоначчи, которые нужно вывести.
4. Если введенное значение меньше 1, пользователь осуществляет ввод до тех пор, пока значение не удовлетворит заданному условию.
5. При помощи цикла for осуществляется вывод заданного количества чисел Фибоначчи.

## Блок схема

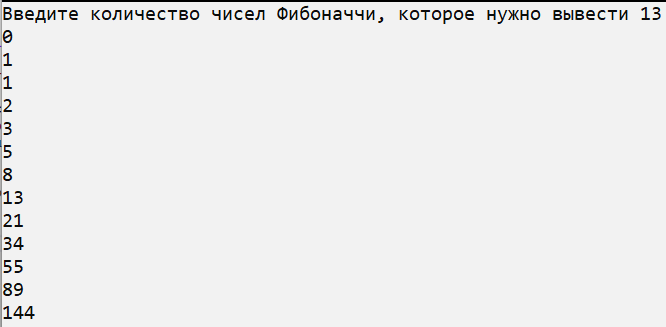


## Код программы



## Результат работы

Пусть, необходимо вывести 13 чисел Фибоначчи



## Вывод

В ходе работы я применила знания о работе с рекурсивными функциями. Мне удалось реализовать функцию нахождения чисел Фибоначчи, а также вывод этих чисел от первого, до необходимого.

# Ханойская башня

**Цель**: научиться решать задачи с использованием рекурсивных функций.

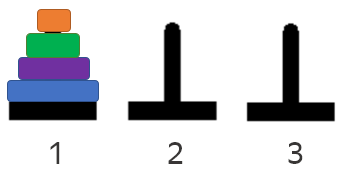
**Постановка задачи:** Даны 1 стержень с дисками разного размера и 2 пустых стержня. Нужно переместить диски с одного стержня на другой, перекладывать можно только по одному диску за ход, складывать диски можно только меньший на больший. Реализовать программу, которая определяет перестановки этих дисков с помощью наименьшего количества ходов.

## Анализ задачи

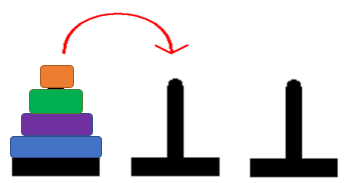
1. Нужно решать задачу не с начала, а с конца. Чтобы переложить пирамидку на нужный стержень, нужно переложить на нужный стержень нижний диск, а сделать это можно только тогда, когда n – 1 дисков будут на свободном стержне.
2. Перекладываем n – 1 дисков на свободный стержень.
3. Перекладываем n-ый диск на нужный стержень.
4. Перекладываем n – 1 дисков на нужный стержень.
5. Чтобы переложить n – 1 дисков, нужно: перекладываем n – 2 дисков на свободный стержень.
6. Перекладываем n – 1 диск на нужный стержень.
7. Перекладываем n – 2 дисков на нужный стержень.
8. Рекурсивный алгоритм продолжается до тех пор, пока n не достигнет 0.
9. Для визуализации проводимых операций необходимо реализовать функцию вывода башен и функцию изменения выводимых башен. Для этих функций необходим массив, где будут хранится текущие позиции колец.
10. Вывод башен должен происходить послойно, т.е. сначала верхний слой всех башен, затем слой, который ниже верхнего и так далее.

## Рассмотрю алгоритм работы Ханойской башни.

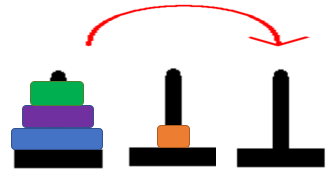
Пусть дано четыре диска, необходимо переместить диски с первого стержня на третий.



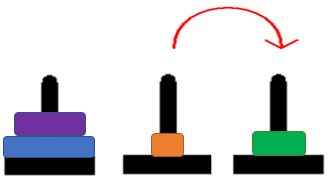
Для того, чтобы переложить синий диск на третий стержень, необходимо переложить три верхних диска на второй стержень. Для того, чтобы переложить фиолетовый диск на второй стержень, необходимо переложить два верхних диска на третий стержень. Для того, чтобы переложить зеленый диск на третий стержень, необходимо переложить оранжевый диск на второй стержень.



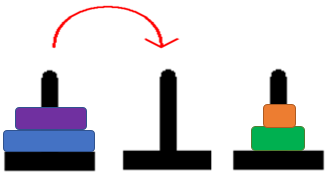
Перекладываю оранжевый диск на второй стержень.



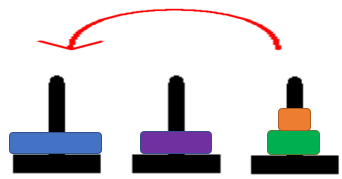
Перекладываю зеленый диск на третий стержень.



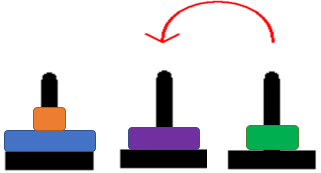
Для того, чтобы переложить фиолетовый диск на второй диск, перекладываю оранжевый диск на третий стержень.



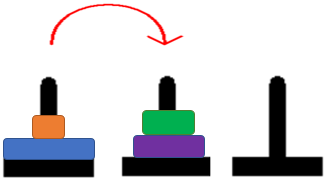
Перекладываю фиолетовый диск на второй стержень.



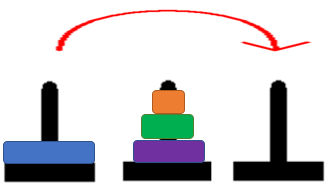
Для того, чтобы переложить зеленый диск на второй стержень, необходимо переложить оранжевый диск на первый стержень.



Перекладываю зеленый диск на второй стержень.



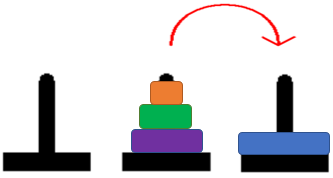
Перекладываю оранжевый диск на второй стержень.



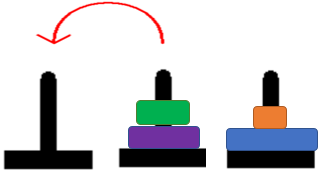
Теперь, когда фиолетовый, зеленый и оранжевые диски находятся на втором стержне, перекладываю синий диск на третий стержень.



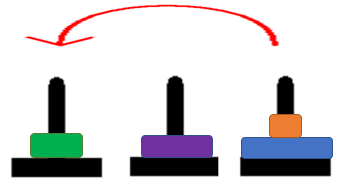
Для того, чтобы переложить фиолетовый диск на третий стержень, необходимо переложить два верхних диска на первый стержень. Для того, чтобы переложить зеленый диск на первый стержень, необходимо переложить оранжевый диск на третий стержень.



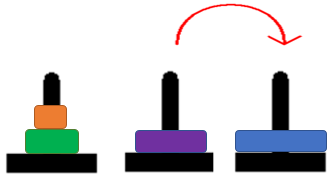
Перекладываю оранжевый диск на третий стержень.



Перекладываю зеленый диск на первый стержень.



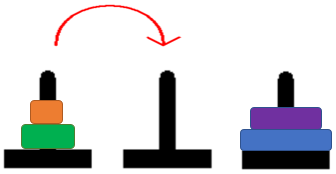
Перекладываю оранжевый диск на первый стержень.



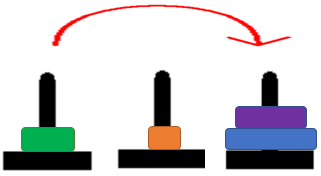
Теперь, когда зеленый и оранжевые диски находятся на первом стержне, а синий на третьем, перекладываю фиолетовый диск на третий стержень.



Для того, чтобы переложить зеленый диск на третий стержень, необходимо переложить оранжевый диск на второй стержень.



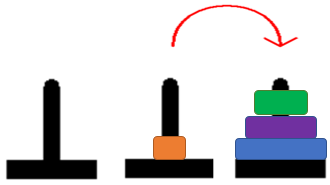
Перекладываю оранжевый диск на второй стержень.



Перекладываю зеленый диск на третий стержень.



Теперь, когда синий, фиолетовый и зеленый диски находятся на третьем стержне, перекладываю оранжевый диск на третий стержень.



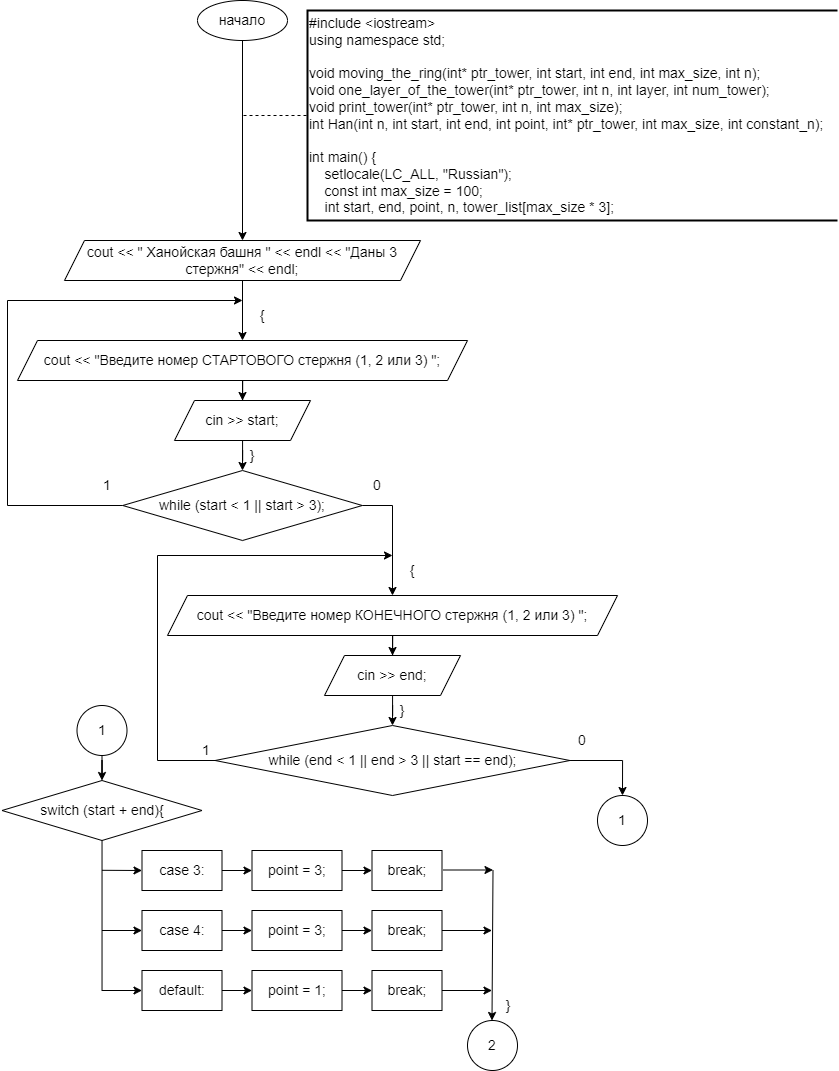
Перекладываю оранжевый диск на третий стержень.

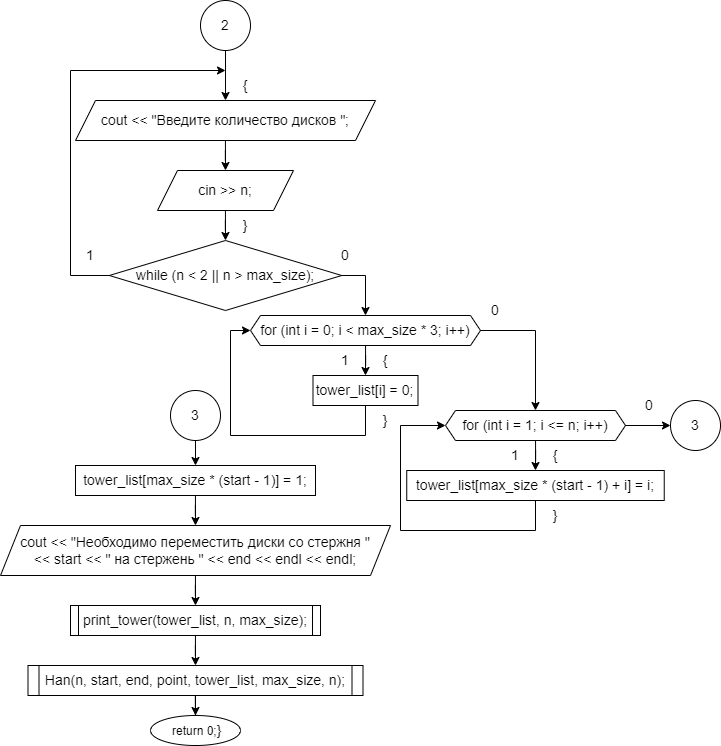


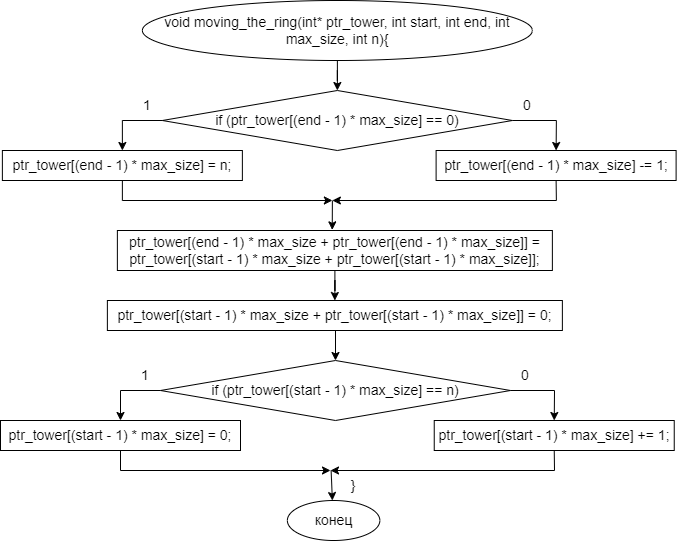
Башня полностью собрана на третьем стержне.

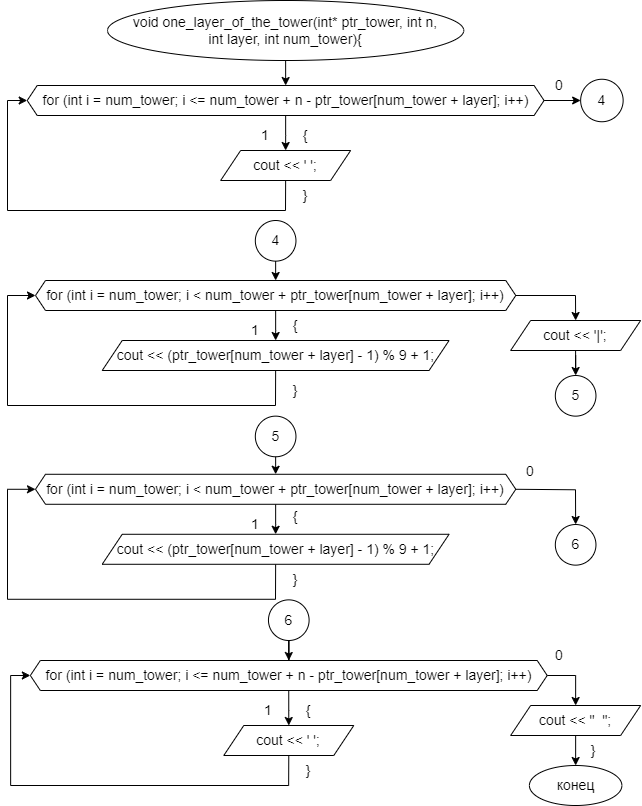
Алгоритм идентичен и для другого количества колец.

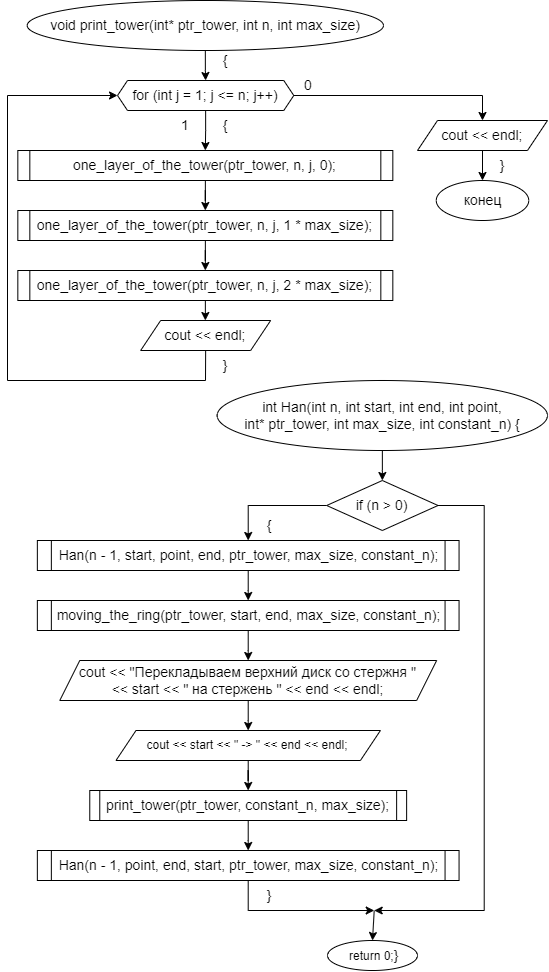
## Блок схема



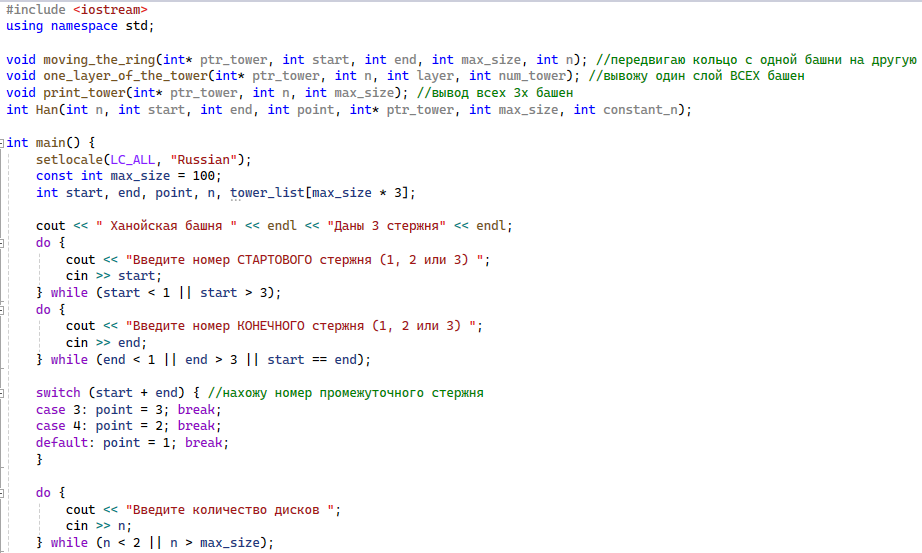


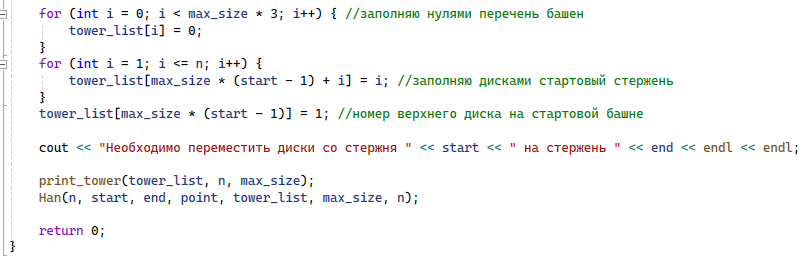


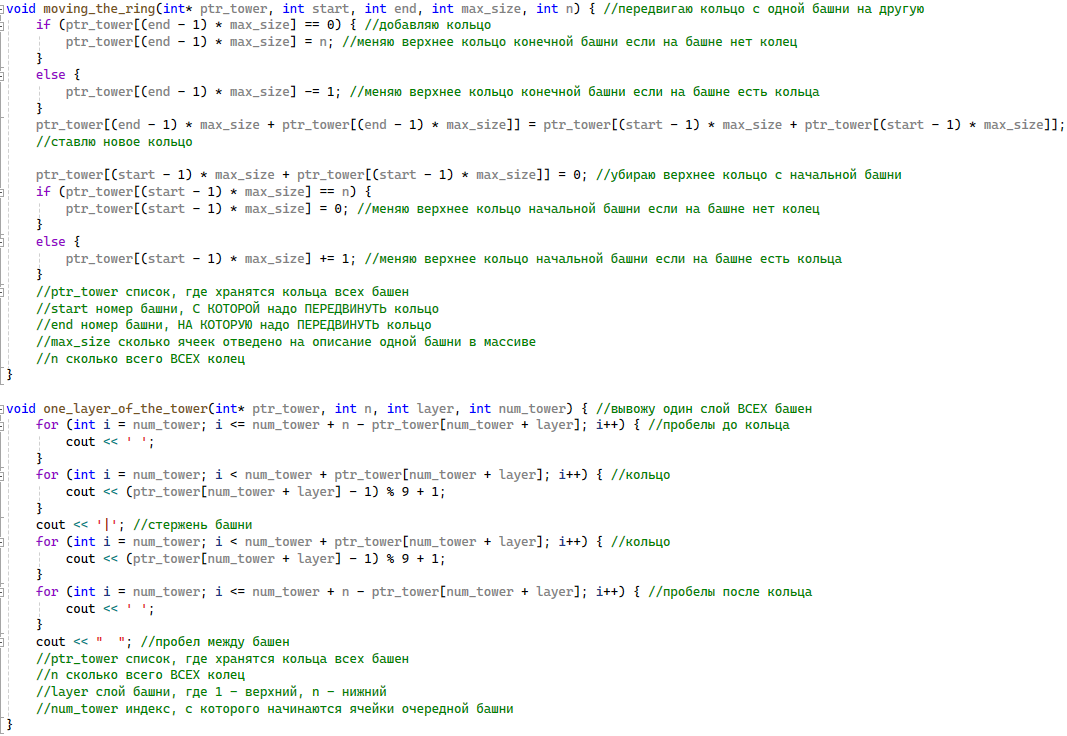


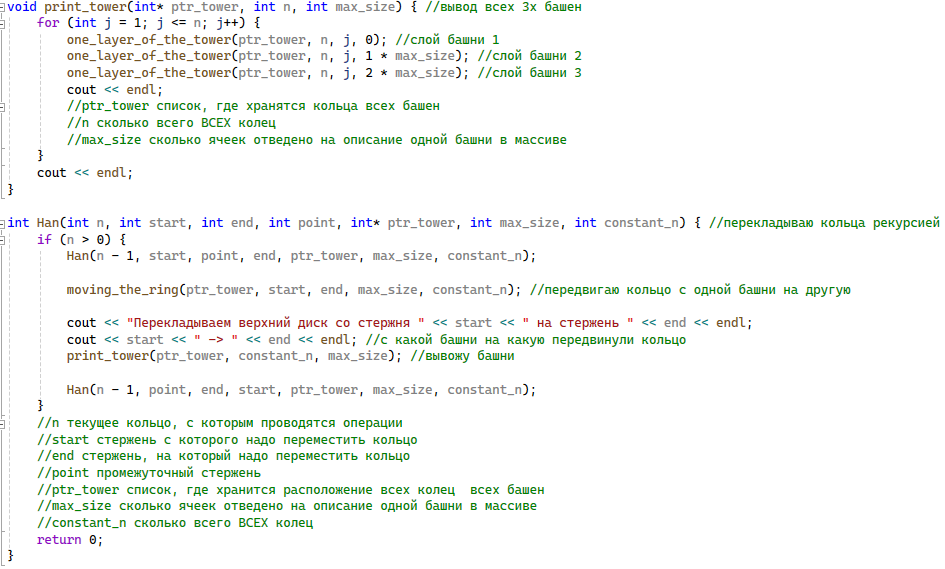


## Код программы



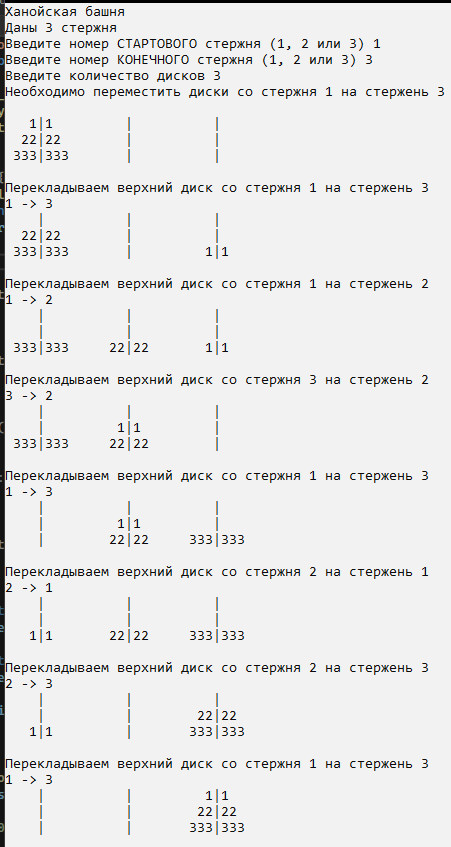




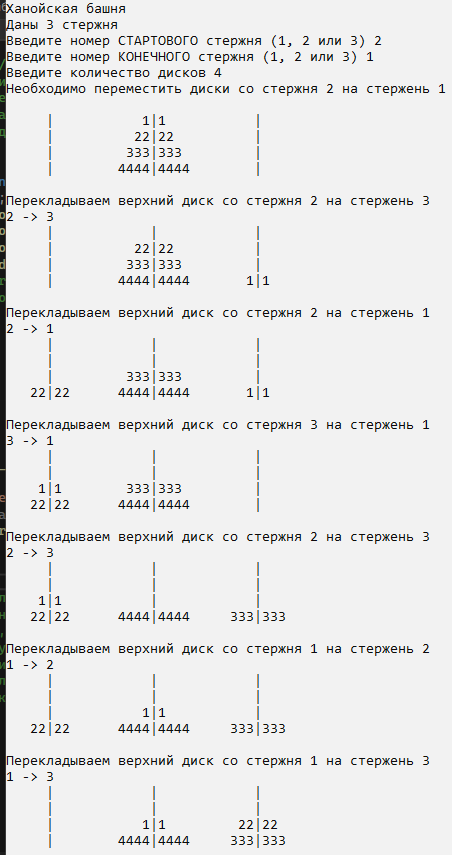


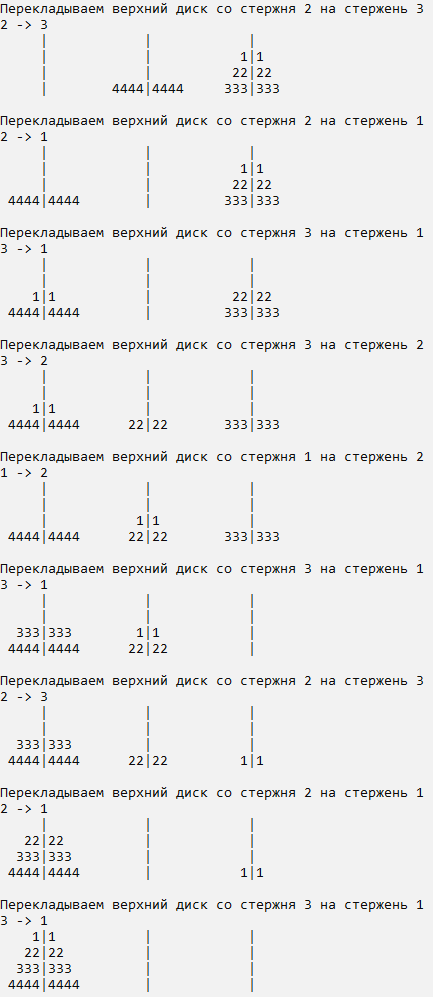
## Результат работы программы

1. Пусть дано 3 диска, начальный стержень – первый, конечный стержень – второй.



1. Пусть дано 4 диска, начальный стержень – второй, конечный стержень – первый.





## Вывод

В ходе работы я применила знания о работе с рекурсивными функциями. Мне удалось реализовать алгоритм, который переносит кольца с одного стержня на другой. Также я выполнила визуализацию проводимых действий с помощью массива, в котором хранились текущие позиции колец, и соответствующих функций.

# GitHub

Ссылка: <https://github.com/SonyAkb/laboratory-work-recursion.git>

