Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа**

**«Решение задачи ханойских башен при помощи рекурсивного алгоритма»**

Выполнил:

Студент группы РИС-23-1б

Шароглазов Егор Алексеевич

Проверила:

Доцент кафедры ИТАС

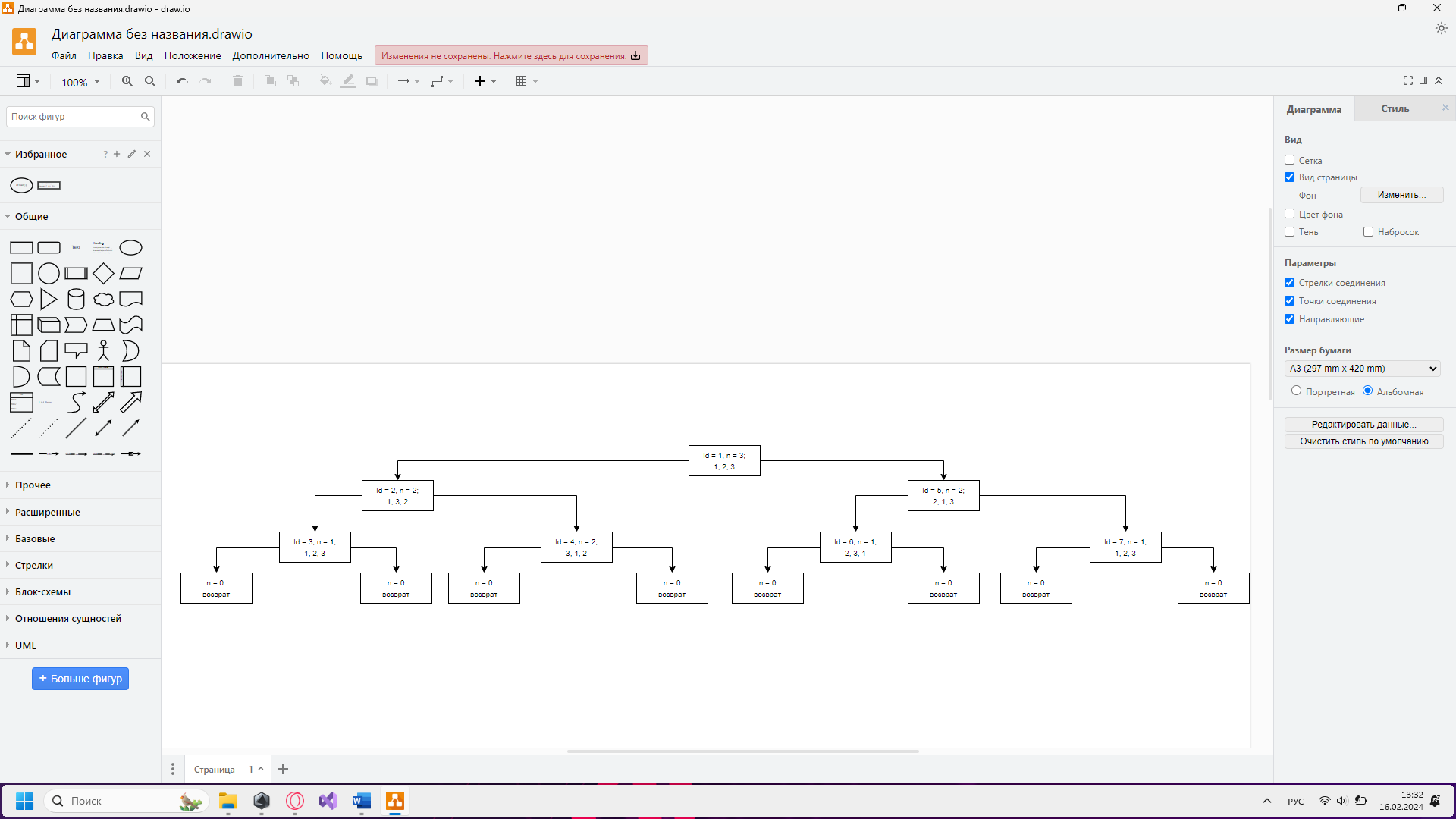
О.А. Полякова

2024 г

**Анализ задачи:**

1. Для решения задачи изначально будет указано, на какой башне были кольца изначально, на какой башне кольца окажутся в конце алгоритма и какая башня была вспомогательной.
2. Каждая итерация рекурсии вызывает 2 ветки. При прохождении по ветке 1 в следующей итерации будут поменяны местами вспомогательная и целевая башни.
3. При прохождении ветки 2 в следующей итерации будут поменяны местами вспомогательная и стартовая башни.
4. Каждая новая итерация рекурсии снижает параметр n на 1, который отвечает за номер перекладываемого диска.
5. Цель задачи – достичь условия выхода из рекурсии по ветке 1, вывести действие, учитывая текущую стартовую и финальную башню, затем достичь условия выхода из рекурсии по ветке 2.

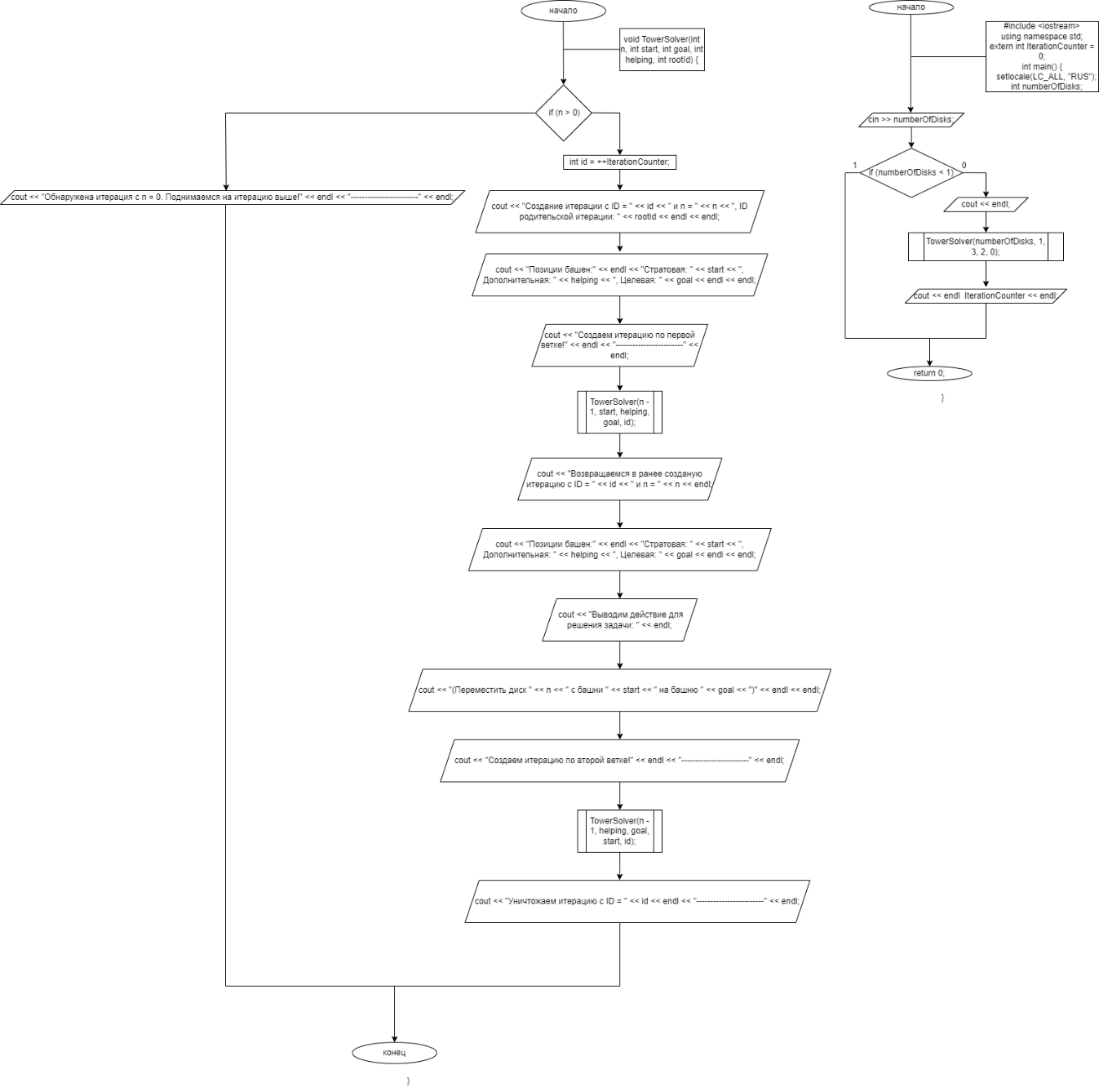
**Блок схема рекурсивных вызовов.**



id – порядковый номер выполняемой итерации

Каждая итерация содержит 1 ветку, меняющую местами вспомогательную и целевую башни, и другу, меняющую стартовую и вспомогательную. Рекурсия в начале выполняется строго по первой ветке до конца. Достигнув его, начинаем подниматься назад по древу, отводя вторую ветку от каждой итерации. Вернувшись к начальной итерации, также отводим от нее вторую ветку и по аналогии достигаем конца. Рекурсия закончит работу по достижению начальной итерации из второй ветки.

**Блок схема алгоритма**



Код программы

#include <iostream>

using namespace std;

extern int IterationCounter = 0;

void TowerSolver(int n, int start, int goal, int helping, int rootId) {

    if (n>0) {

        int id = ++IterationCounter;

        cout<<"Создание итерации с ID = " << id <<" и n = " << n << ", ID родительской итерации: "<<rootId<<endl<<endl;

        cout<<"Позиции башен: "<<endl <<"Стартовая: " <<start << ", Дополнительная: "<< helping <<", Целевая: " <<goal<<endl<<endl;

        cout<<"Создаем итерацию по первой ветке"<<endl<<"---------------------------"<<endl;

        TowerSolver(n-1, start, helping, goal, id);

        cout<<"Возвращаемся в ранее созданную итерцию с ID = " <<id<<" и n = "<<n<<endl;

        cout<<"Позиции башен: "<<endl << "Стартовая: " <<start <<", Дополнительная: "<<helping<< ", Целевая: "<<goal<<endl<<endl;

        cout<<"Выводим действие для решения задачи: "<<endl;

        cout<<"(Переместить диск) " << n << " с башни " << start << " на башню " << goal<< ")"<< endl << endl;

        cout<<"Создаем итерацию по второй ветке" << endl << "--------------------------------"<<endl;

        TowerSolver(n-1, helping, goal, start, id);

        cout<< "Уничтожаем итерацию с ID " << id <<endl << "----------------------" <<endl;

    }

    else {

        cout<<"Обнаружена итерация с n = 0. Поднимаемся на итерацию выше" <<endl <<"--------------------------" << endl;

    }

}

int main() {

    setlocale(LC\_ALL, "RU");

    int numberOfDisks;

    cout<< "Введите количество дисков" <<endl;

    cin >> numberOfDisks;

    if (numberOfDisks < 1) {

        cout<<"Число дисков должно быть больше или равно 1" <<endl;

        return 0;

    }

    cout<<endl;

    TowerSolver(numberOfDisks, 1, 3, 2, 0);

    cout<<endl <<"Всего потребовалось итераций: " <<IterationCounter<<endl;

}

Примеры работы алгоритма

