Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

ОТЧЁТ ПО ЛАБОЛАТОРНОЙ РАБОТЕ №3

Тема: «Ханойская башня»

Выполнила

Студентка РИС-24-1б

Конькова С. С.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2025

**Постановка задачи**

Дано: 3 стержня, на 1 стержень нанизано n колец, разного диаметра (пирамида) использую доп. стержень.

1 условие: 1 ход – перенос только одного диска

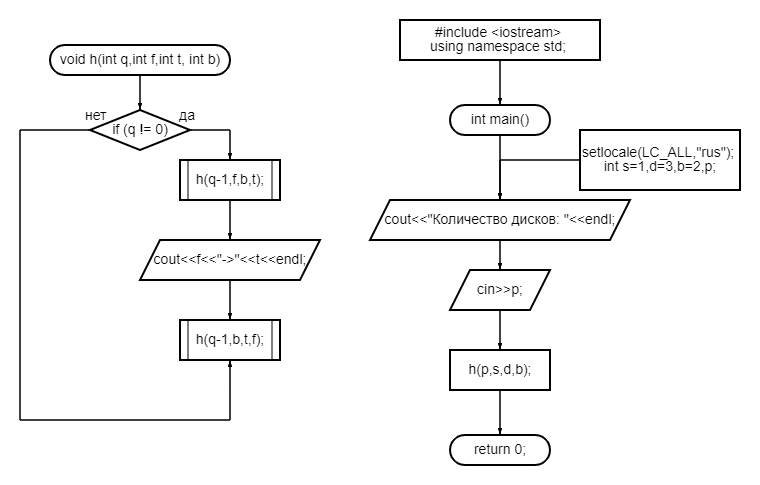
2 условие: нельзя складывать больший диск на меньший

Дополнительный temp, с которого начинаем перекладывать start, на который

**Анализ**

1. Создаём функцию для вывода башни через переменные 1, 2, 3
2. Перемещаем диск рекурсивно
3. Копируем положения остальных дисков
4. Рекурсивно решаем Ханойскую башню

**Блок – схема**



**Коды программ**

#include <iostream>

using namespace std;

//Функция вывода

void out(char\*\*& first, char\*\*& second, char\*\*& third, int n)

{

cout << endl << endl;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n \* 2 + 1; j++)

cout << first[i][j];

cout << '\t';

for (int j = 0; j < n \* 2 + 1; j++)

cout << second[i][j];

cout << '\t';

for (int j = 0; j < n \* 2 + 1; j++)

cout << third[i][j];

cout << endl;

}

cout << endl;

}

//Функция перемещения диска

void perem(char\*\*& start, char\*\*& point, int n)

{

int Istart = 0;

for (int i = 0; i < n; i++)

{

if (start[i][n - 1] == '\*')

{

Istart = i;

break;

}

}

int Ipoint = n - 1;

for (int i = n - 1; i >= 0; i--)

{

if (point[i][n - 1] == ' ')

{

Ipoint = i;

break;

}

}

// Копируем диск из start в point

for (int j = 0; j < n \* 2 + 1; j++)

point[Ipoint][j] = start[Istart][j];

// Очищаем start

for (int j = 0; j < n \* 2 + 1; j++)

start[Istart][j] = (j == n) ? '|' : ' ';

}

//Функция возращения по номеру

char\*\*& number\_return(char\*\*& first, char\*\*& second, char\*\*& third, int number)

{

if (number == 1)

return first;

else if (number == 2)

return second;

return third;

}

// Рекурсивная функция для решения Ханойской башни

void hanoi(int n, int start, int temp, int point, char\*\*& first, char\*\*& second, char\*\*& third, int N)

{

if (n == 1)

{

cout << "Переместить диск 1 со стержня " << start << " на стержень " << point << endl;

perem(number\_return(first, second, third, start), number\_return(first, second, third, point), N);

out(first, second, third, N); // Вывод состояния башен после перемещения

return;

}

hanoi(n - 1, start, point, temp, first, second, third, N);

cout << "Переместить диск " << n << " со стержня " << start << " на стержень " << point << endl;

perem(number\_return(first, second, third, start), number\_return(first, second, third, point), N);

out(first, second, third, N);

hanoi(n - 1, temp, start, point, first, second, third, N);

}

int main()

{

setlocale(LC\_ALL, "Russian");

int n;

cout << "Введите количество дисков: ";

cin >> n;

// Создание 3 матриц для отображения башен

char\*\* first = new char\* [n];

char\*\* second = new char\* [n];

char\*\* third = new char\* [n];

for (int i = 0; i < n; i++)

{

first[i] = new char[n \* 2 + 1];

second[i] = new char[n \* 2 + 1];

third[i] = new char[n \* 2 + 1];

// Заполнение матриц пробелами и центральным стержнем

for (int j = 0; j < n \* 2 + 1; j++)

{

if (j == n)

{

first[i][j] = '|';

second[i][j] = '|';

third[i][j] = '|';

}

else

{

first[i][j] = ' ';

second[i][j] = ' ';

third[i][j] = ' ';

}

}

}

// Заполнение первой матрицы дисками

for (int i = 0; i < n; i++)

{

for (int j = n - i - 1; j <= n + i + 1; j++)

{

if (j != n)

first[i][j] = '\*';

}

}

// Вывод начального состояния башен

cout << "Изначальное состояние: ";

out(first, second, third, n);

// Решение Ханойской башни

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

// Освобождение памяти

for (int i = 0; i < n; i++)

{

delete[] first[i];

delete[] second[i];

delete[] third[i];

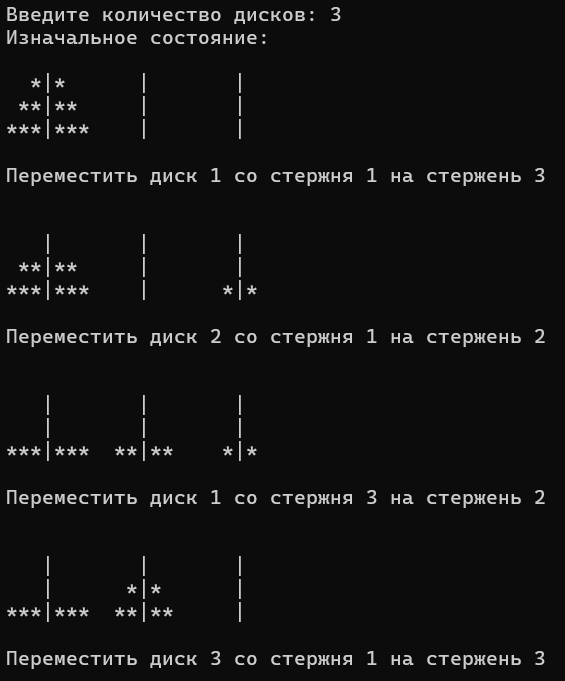
}

delete[] first;

delete[] second;

delete[] third;

return 0;

}**Результаты программы**