Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

**«Пермский национальный исследовательский
политехнический университет»**

Электротехнический факультет
Кафедра «Информационные технологии и автоматизированные системы»
направление подготовки: 09.03.01– «Информатика и вычислительная техника»

# Лабораторная работа № 3
# по дисциплине
# «Информатика»
# на тему
# «Рекурсивные функции»

Выполнил студент гр. ИВТ-23-1б
Южаков Федор Алексеевич

Проверил:
доцент кафедры ИТАС
Денис Владимирович Яруллин

(оценка) (подпись)

(дата)

г. Пермь, 2024

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1 Вариант задания

Вариант 3.

Решить 4 задачи с использованием рекурсивных функций:

Задача 1.

$$S = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + (-1)^n * \frac{x^{2n+1}}{(2n+1)!}$$

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

Задача 2.

Функция принимает на вход n – порядковый номер числа Фибоначчи, задача – вывести все числа Фибоначчи до числа n, и само число n.

Задача 3.

Имеется три стержня, перенести пирамиду из дисков с первого на третий, при этом не размещая больший диск поверх меньшего. За один ход можно переместить только один диск.

Задача 4.

Расставить 8 ферзей на шахматной доске 8 на 8 так, чтобы они не били друг друга.

## 2 Анализ задач

Задача 1.

В данной задаче пользователь вводит числа n и x.

Так как все члены ряда задаются формулой n - го члена, можно найти разницу между n - ным и n-1 -ым членами, таким образам получим следующую формулу для нахождения следующего члена через предыдущий:

$$S_n = S_{n-1} \cdot \left(-\frac{x^2}{2n \cdot (2n+1)}\right)$$

где выходом из рекурсии будет являться следующее выражение: $S_0 = x$. Для вычисления суммы просто объявим глобальную переменную, присвоим ей значение 0. В рекурсивной функции после рекурсивного вызова будем добавлять к сумме вычисленный элемент. Аргументами функции будут числа n и x. При этом функция не будет возвращать никаких значений, но будет считать сумму, поэтому в теле программы после вызова функции будем выводить значение суммы.

Для вычисления суммы простым нахождением всех необходимых элементов и их сложением напишем функцию, принимающую n и x, и возвращающую значение n-го элемента. Для подсчета факториала напишем ещё одну функцию, принимающую какое-либо число и возвращающую его факториал. Для подсчета суммы элементов в теле программы объявим переменную, равную нулю, и составим цикл, который будет вызывать функцию и складывать все полученные элементы от 0 и до n включительно.

При сравнении этих двух методов можно заметить, что для реализации и понятия человеком более простым является второй метод, но более компактным и простым в плане вычислительных мощностей является первый метод, ведь все элементы вычисляются по порядку и сразу складываются, не увеличивая сложность вычисления, а во втором методе, элемент за элементом, сложность вычислений возрастает, и при вычислении суммы компьютеру приходится с нуля вычислять каждый элемент.

Задача 2.

В данной задаче пользователь вводит натуральное число n – порядковый номер числа Фибоначчи.

Вычисление чисел Фибоначчи происходит следующим образом: первое число 0, второе 1, а третье и все последующие вычисляются путем сложения двух предыдущих. Это уже рекурсивная формула с условиями выхода из рекурсии. Остается лишь печатать все вычисленные числа, и задача решена.

Задача 3.

В данной задаче пользователь вводит число n – количество дисков на первом стержне.

Для решения сначала создадим стартовое положение. Пусть массив n на 3 – три стержня, и на первом из них лежат диски n, n-1, n-2, … , 2, 1, где n – самый крупный диск, а 1 – самый маленький. Объявим этот массив глобально, вместо n возьмем какое-нибудь большое число, например, 100, работать же будем с введённым числом n. В теле программы заполним нулевой столбец массива (первая башня) числами от n до 1, а остальные столбцы заполним нулями. Теперь, для удобства напишем функцию вывода массива и функцию, перекладывающую самый верхний диск с башни start на башню finish. Самой важной будет рекурсивная функция, решающая задачу. Она будет действовать с конца: когда все диски, кроме самого большого, лежат на втором стержне она переложит этот диск на третий стержень. Поэтому в начале задачи функция углубится в рекурсию, и будет перекладывать диски до тех пор, пока не дойдёт до конечной ситуации, описанной выше, после чего выйдет из рекурсии. При каждом перекладывании диска функция будет вызывать функцию печати для того, чтобы наглядно увидеть решение задачи пошагово.

Задача 4.

В данной задаче пользователь не будет вводить никаких данных.

Для решения задачи создадим глобальный массив 8 на 8, заполненный нулями. Числа в этом массиве будут играть ту или иную роль. Ноль – пустая клетка, единица – битое поле, минус единица – ферзь. В этой задаче важным является тот факт, что в один столбец нельзя установить более одного ферзя, следовательно максимальное количество ферзей изначально не может быть больше восьми.

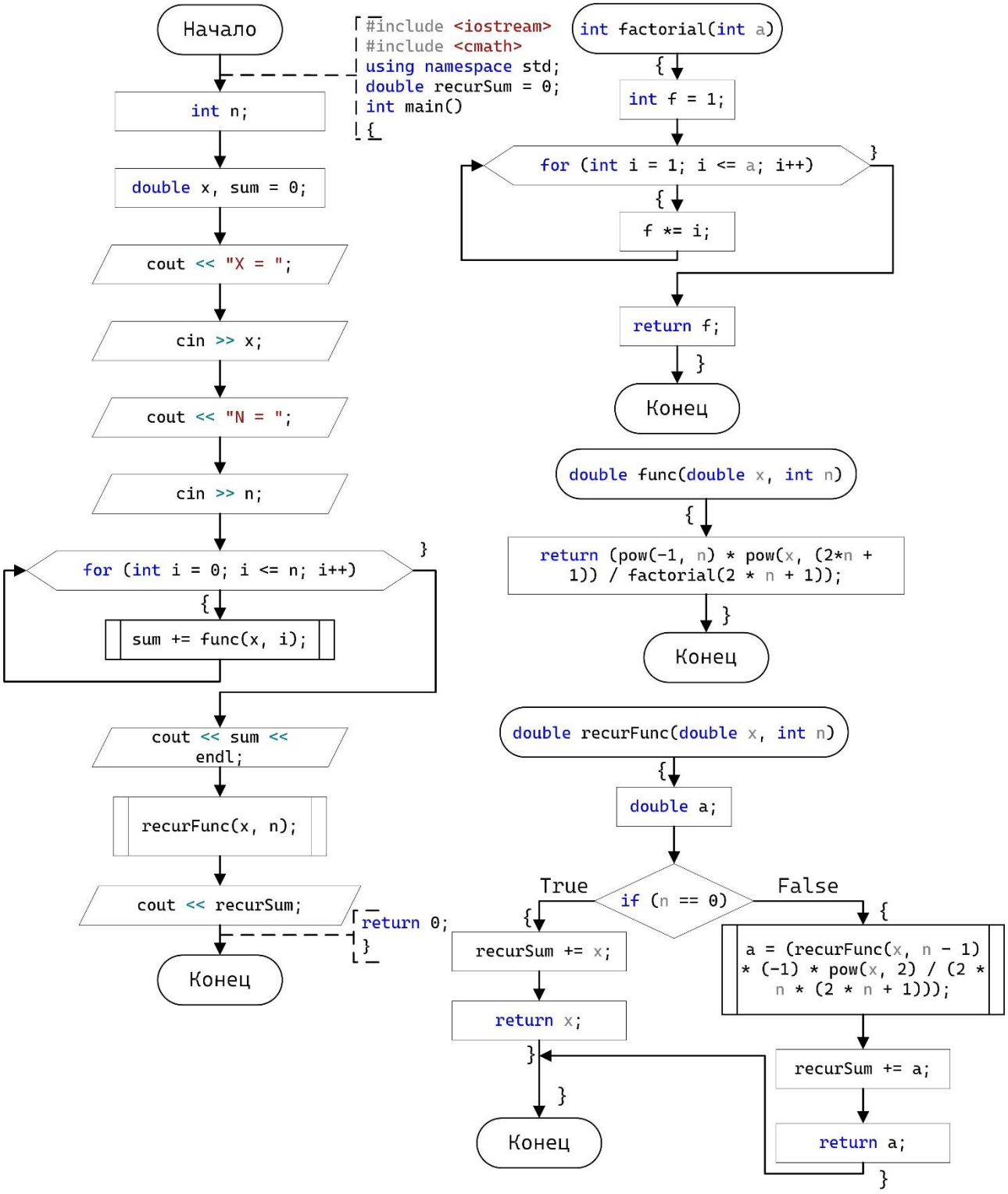
Для удобства создадим функцию, устанавливающую ферзя. Ее задача – по указанным координатам установить ферзя (число минус один) и отметить битые поля, создаваемые ферзём по вертикали, горизонтали и двум диагоналям. Так как данная задача решается методом проб и ошибок, сразу напишем функцию, убирающую ферзя – полностью обратную данной.

Самой важной будет рекурсивная функция, расставляющая ферзей. Принцип работы этой функции следующий: начиная с первой строчки и первого столбца эта функция ищет пустую клетку, и пробует там поставить ферзя, далее она вызывает себя на следующий столбец, и если в нем уже не удается установить ферзя, то она убирает ферзя и возвращается в предыдущий столбец на следующую строку, после чего пробует установить ферзя там, далее снова смотрит в следующий столбец, снова в следующий, при неудаче снова в предыдущий, потом в следующий, и так далее, пока не сможет установить ферзя в последний столбец, на этом выполнение функции прекращается.

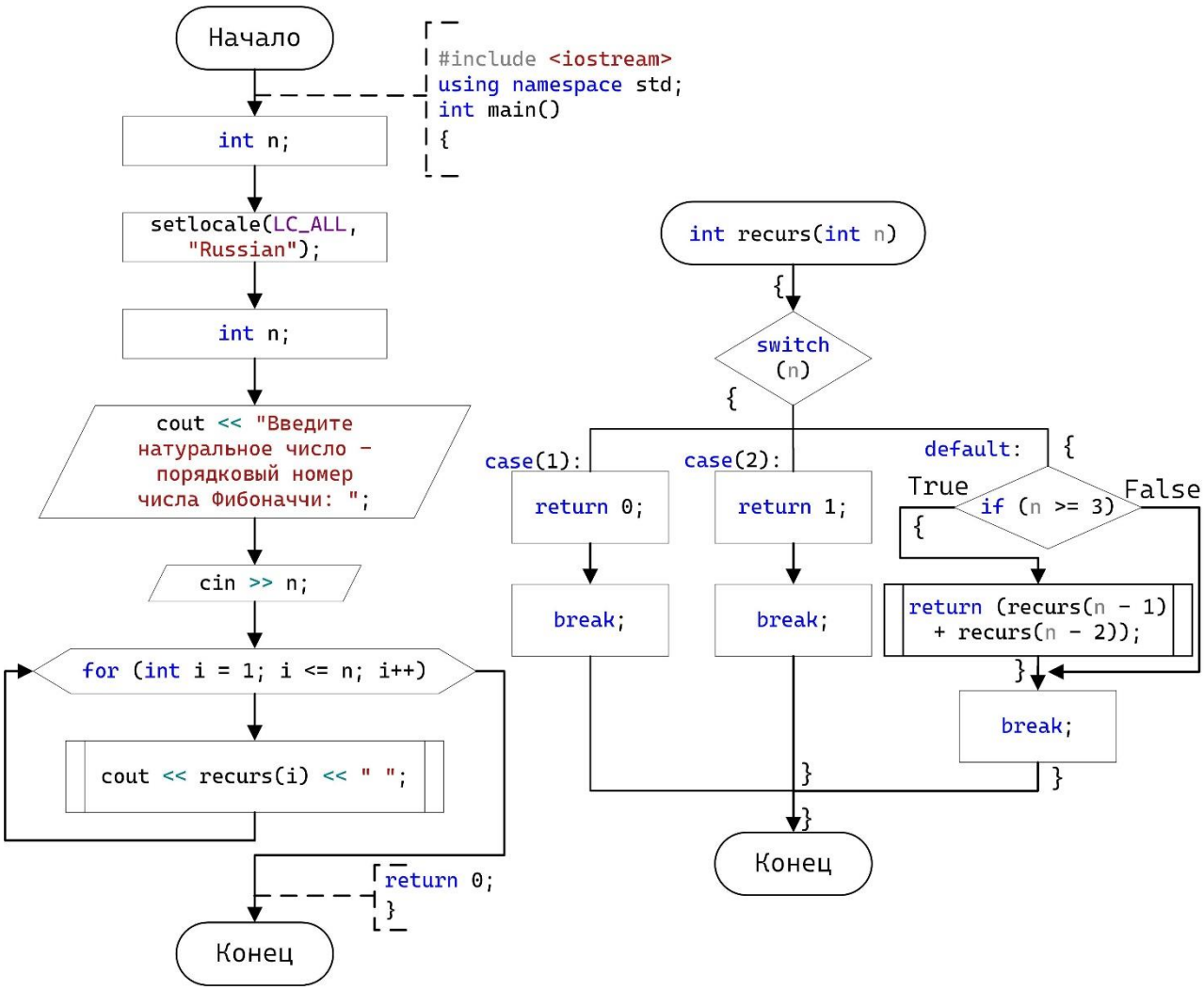
После выполнения рекурсивной функции остается лишь вывести массив, заменяя пустые и битые клетки на пустой квадрат, а клетки с ферзями на какой-нибудь отличительный знак.

## 3 Блок схема

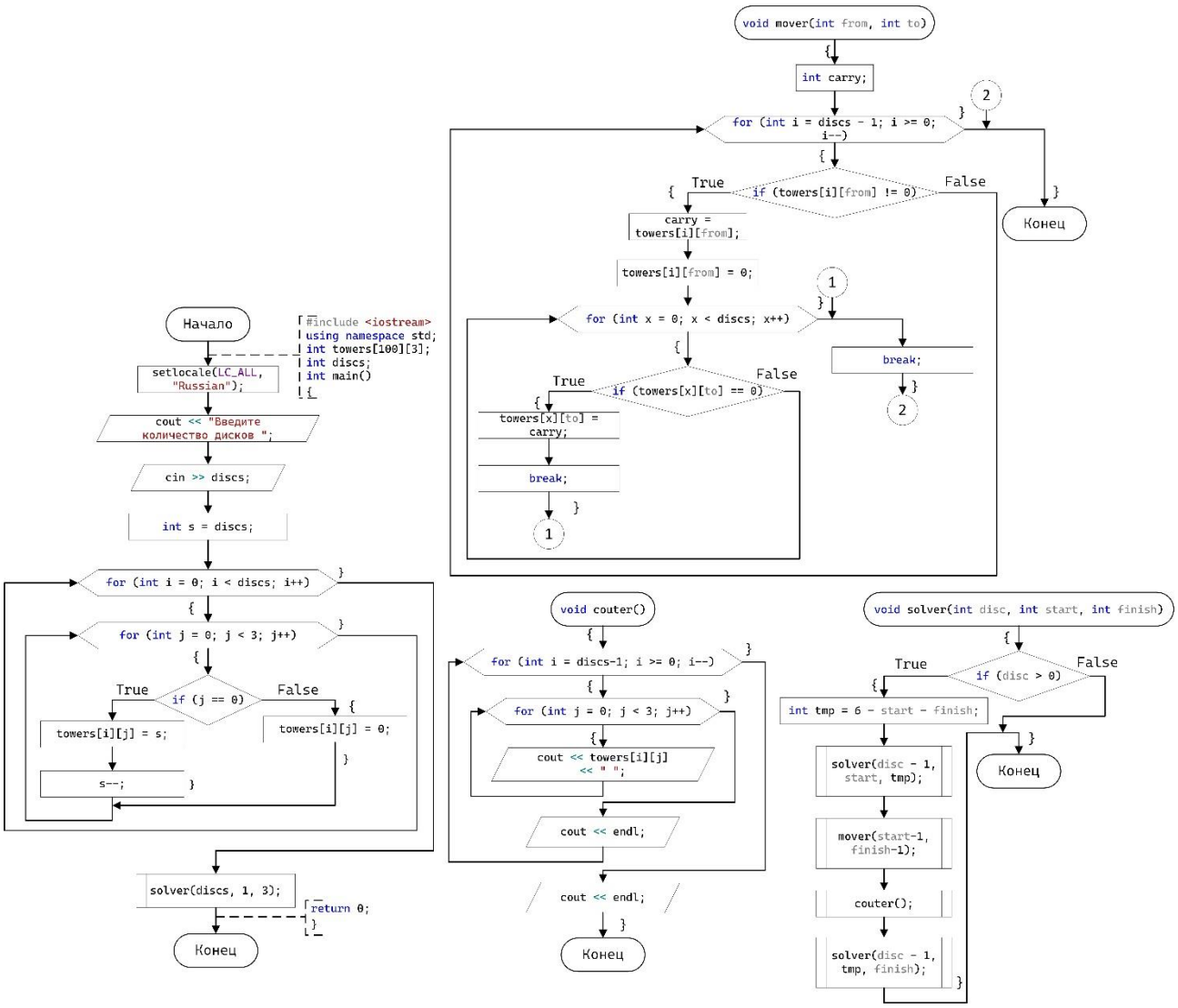
Проанализировав все задачи, составим подробные блок схемы всех задач.
Задача 1.


Начало
#include <iostream>
#include <cmath>
using namespace std;
double recurSum = 0;
int main()
{
int n;
double x, sum = 0;
cout << "X = ";
cin >> x;
cout << "N = ";
cin >> n;
for (int i = 0; i <= n; i++)
{
sum += func(x, i);
}
cout << sum << endl;
recurFunc(x, n);
cout << recurSum;
return 0;
}
Конец
int factorial(int a)
{
int f = 1;
for (int i = 1; i <= a; i++)
{
f *= i;
}
return f;
}
Конец
double func(double x, int n)
{
return (pow(-1, n) * pow(x, (2*n + 1)) / factorial(2 * n + 1));
}
Конец
double recurFunc(double x, int n)
{
double a;
if (n == 0)
True
{
recurSum += x;
return x;
}
Конец
False
{
a = (recurFunc(x, n - 1) * (-1) * pow(x, 2) / (2 * n * (2 * n + 1)));
recurSum += a;
return a;
}

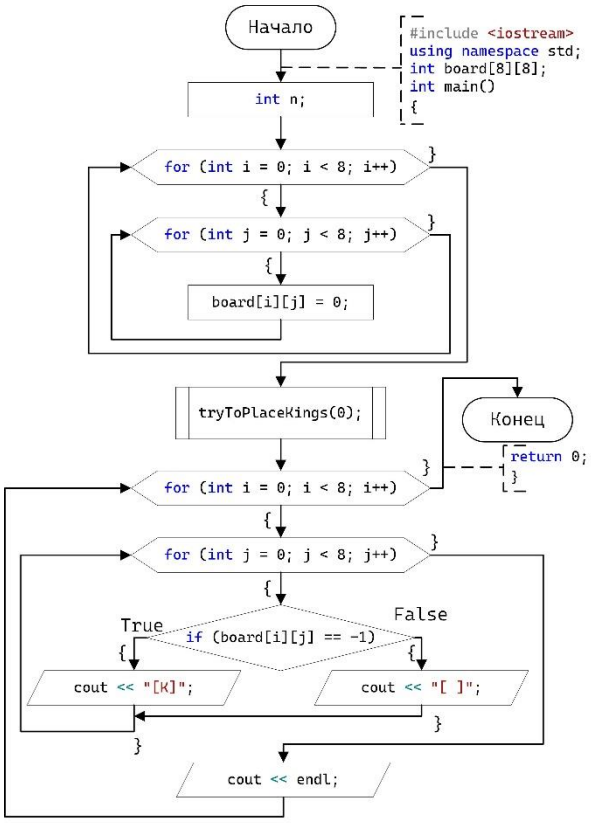
Задача 2.

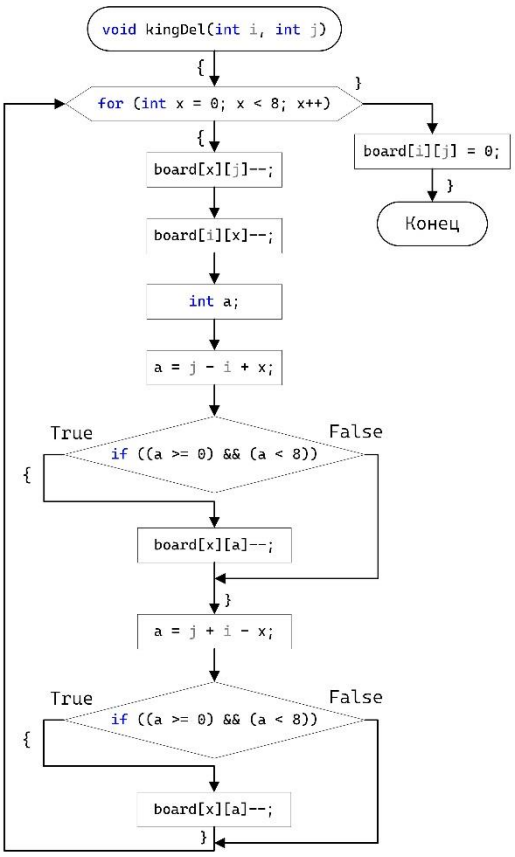

Начало
#include <iostream>
using namespace std;
int main()
{
int n;
setlocale(LC_ALL, "Russian");
int n;
cout << "Введите натуральное число – порядковый номер числа Фибоначчи: ";
cin >> n;
for (int i = 1; i <= n; i++)
cout << recurs(i) << " ";
return 0;
}
Конец
int recurs(int n)
{
switch (n)
{
case(1):
return 0;
break;
case(2):
return 1;
break;
default:
{
True
if (n >= 3)
False
{
return (recurs(n - 1) + recurs(n - 2));
}
break;
}
}
Конец

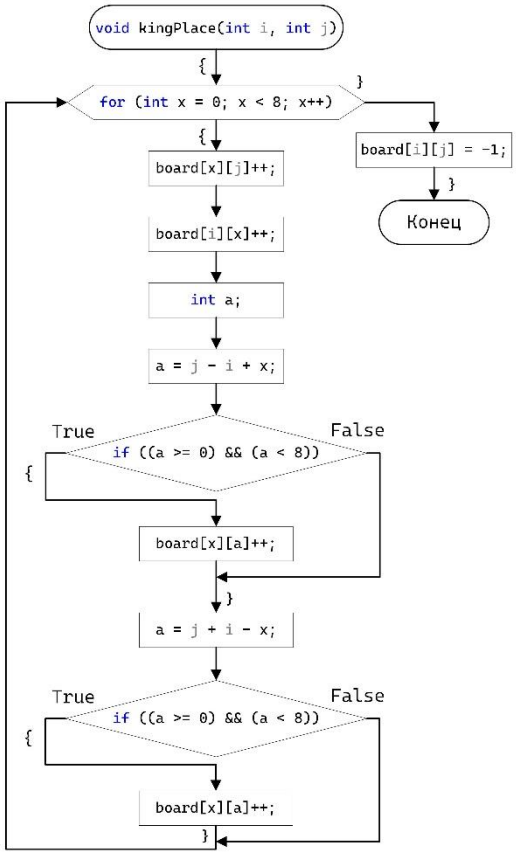
Задача 3.

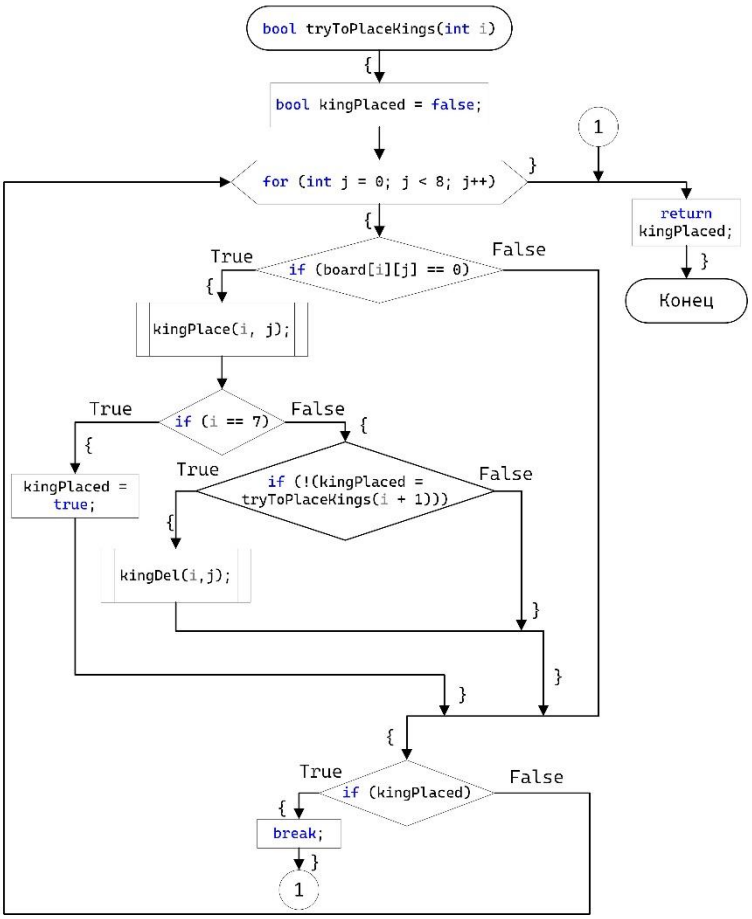

Начало
#include <iostream>
using namespace std;
int towers[100][3];
int discs;
int main()
{
setlocale(LC_ALL, "Russian");
cout << "Введите количество дисков ";
cin >> discs;
int s = discs;
for (int i = 0; i < discs; i++)
for (int j = 0; j < 3; j++)
if (j == 0)
True
False
towers[i][j] = s;
towers[i][j] = 0;
s--;
solver(discs, 1, 3);
return 0;
}
Конец
void mover(int from, int to)
int carry;
for (int i = discs - 1; i >= 0; i--)
if (towers[i][from] != 0)
carry = towers[i][from];
towers[i][from] = 0;
for (int x = 0; x < discs; x++)
if (towers[x][to] == 0)
towers[x][to] = carry;
break;
1
2
void couter()
for (int i = discs-1; i >= 0; i--)
for (int j = 0; j < 3; j++)
cout << towers[i][j] << " ";
cout << endl;
void solver(int disc, int start, int finish)
if (disc > 0)
int tmp = 6 - start - finish;
solver(disc - 1, start, tmp);
mover(start-1, finish-1);
couter();
solver(disc - 1, tmp, finish);

Задача 4.


Начало
#include <iostream>
using namespace std;
int board[8][8];
int main()
{
int n;
for (int i = 0; i < 8; i++)
for (int j = 0; j < 8; j++)
board[i][j] = 0;
tryToPlaceKings(0);
Конец
return 0;
}
for (int i = 0; i < 8; i++)
for (int j = 0; j < 8; j++)
True
if (board[i][j] == -1)
False
cout << "[K]";
cout << "[ ]";
cout << endl;


void kingDel(int i, int j)
for (int x = 0; x < 8; x++)
board[i][j] = 0;
Конец
board[x][j]--;
board[i][x]--;
int a;
a = j - i + x;
True
if ((a >= 0) && (a < 8))
False
board[x][a]--;
a = j + i - x;
True
if ((a >= 0) && (a < 8))
False
board[x][a]--;


void kingPlace(int i, int j)
for (int x = 0; x < 8; x++)
board[i][j] = -1;
Конец
board[x][j]++;
board[i][x]++;
int a;
a = j - i + x;
True
if ((a >= 0) && (a < 8))
False
board[x][a]++;
a = j + i - x;
True
if ((a >= 0) && (a < 8))
False
board[x][a]++;


bool tryToPlaceKings(int i)
bool kingPlaced = false;
1
for (int j = 0; j < 8; j++)
return kingPlaced;
Конец
True
if (board[i][j] == 0)
False
kingPlace(i, j);
True
if (i == 7)
False
kingPlaced = true;
True
if (!(kingPlaced = tryToPlaceKings(i + 1)))
False
kingDel(i,j);
True
if (kingPlaced)
False
break;
1

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## 4 Результат решения

### 4.1 Готовая программа

Исходя из подробных блок схем, составим программы на языке C++.

Таблица 1 – Готовая программа задачи 1

```
#include <iostream>
#include <cmath>
using namespace std;
double recurSum = 0;
int factorial(int a)
{
    int f = 1;
    for (int i = 1; i <= a; i++)
    {
        f *= i;
    }
    return f;
}
double func(double x, int n)
{
    return (pow(-1, n) * pow(x, (2*n + 1)) / factorial(2 * n + 1));
}
double recurFunc(double x, int n)
{
    double a;
    if (n == 0)
    {
        recurSum += x;
        return x;
    }
    else
    {
        a = (recurFunc(x, n - 1) * (-1) * pow(x, 2) / (2 * n * (2 * n + 1)));
        recurSum += a;
        return a;
    }
}
int main()
{
    int n;
    double x, sum = 0;
    cout << "X = ";
    cin >> x;
    cout << "N = ";
    cin >> n;
    for (int i = 0; i <= n; i++)
    {
        sum += func(x, i);
    }
    cout << sum << endl;
    recurFunc(x, n);
    cout << recurSum;
    return 0;
}
```

Таблица 2 – Готовая программа задачи 2

```
#include <iostream>
using namespace std;
int recurs(int n)
{
    switch (n)
    {
    case(1): return 0; break;
    case(2): return 1; break;
    default:
    {
        if (n >= 3)
        {
            return (recurs(n - 1) + recurs(n - 2));
        }
        break;
    }
    }
}
int main()
{
    setlocale(LC_ALL, "Russian");
    int n;
    cout << "Введите натуральное число - порядковый номер числа Фибоначчи: ";
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cout << recurs(i) << " ";
    }
    return 0;
}
```

Таблица 3 – Готовая программа задачи 3

```
#include <iostream>
using namespace std;
int towers[100][3];
int discs;
void mover(int from, int to)
{
	int carry;
	for (int i = discs - 1; i >= 0; i--)
	{
		if (towers[i][from] != 0)
		{
			carry = towers[i][from];
			towers[i][from] = 0;
			for (int x = 0; x < discs; x++) {
				if (towers[x][to] == 0) {
					towers[x][to] = carry;
					break;
				}
			}
			break;
		}
	}
}
void couter()
{
	for (int i = discs-1; i >= 0; i--)
	{
		for (int j = 0; j < 3; j++)
		{
			cout << towers[i][j] << " ";
		}
		cout << endl;
	}
	cout << endl;
}
void solver(int disc, int start, int finish)
{
	if (disc > 0)
	{
		int tmp = 6 - start - finish;
		solver(disc - 1, start, tmp);
		mover(start-1, finish-1);
		couter();
		solver(disc - 1, tmp, finish);
	}
}
int main()
{
	setlocale(LC_ALL, "Russian");
	cout << "Введите количество дисков ";
	cin >> discs;
	int s = discs;
	for (int i = 0; i < discs; i++)
	{
		for (int j = 0; j < 3; j++)
		{
			if (j == 0)
			{
				towers[i][j] = s;
				s--;
			}
			else towers[i][j] = 0;
		}
	}
	solver(discs, 1, 3);
	return 0;
}
```

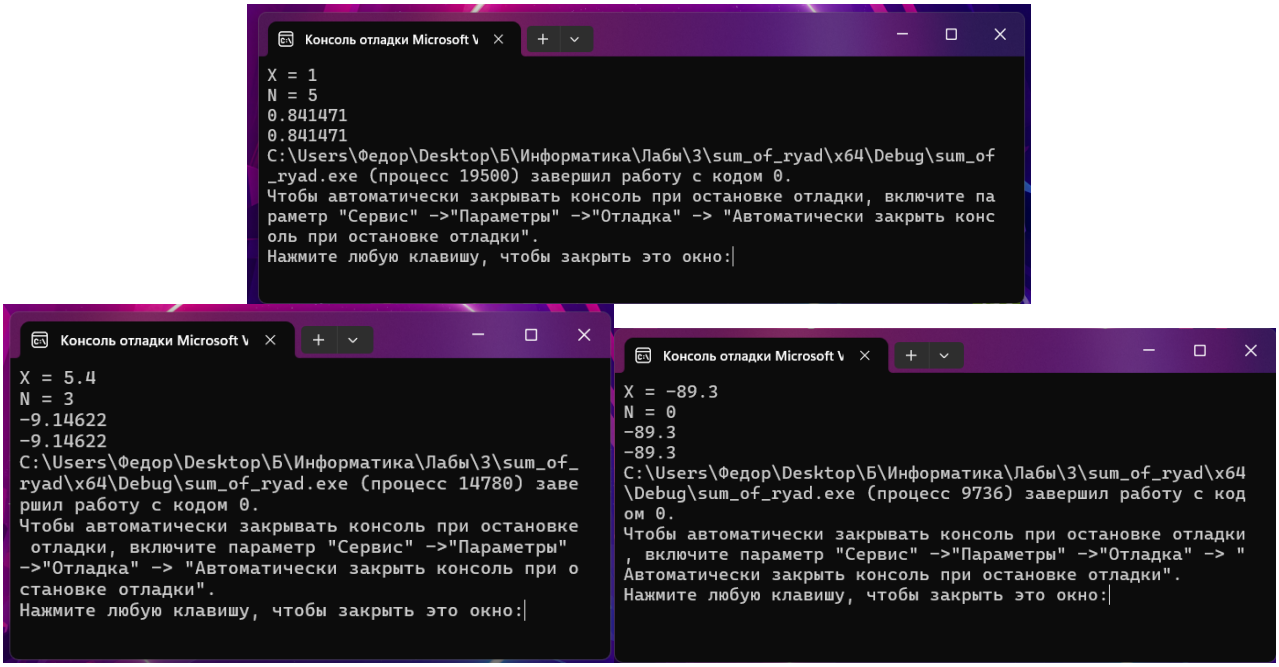
Таблица 4 – Готовая программа задачи 4

```
#include <iostream>
using namespace std;
int board[8][8];
void kingDel(int i, int j)
{
    for (int x = 0; x < 8; x++)
    {
        board[x][j]--;
        board[i][x]--;
        int a;
        a = j - i + x;
        if ((a >= 0) && (a < 8)) board[x][a]--;
        a = j + i - x;
        if ((a >= 0) && (a < 8)) board[x][a]--;
    }
    board[i][j] = 0;
}
void kingPlace(int i, int j)
{
    for (int x = 0; x < 8; x++)
    {
        board[x][j]++;
        board[i][x]++;
        int a;
        a = j - i + x;
        if ((a >= 0) && (a < 8)) board[x][a]++;
        a = j + i - x;
        if ((a >= 0) && (a < 8)) board[x][a]++;
    }
    board[i][j] = -1;
}
bool tryToPlaceKings(int i)
{
    bool kingPlaced = false;
    for (int j = 0; j < 8; j++)
    {
        if (board[i][j] == 0)
        {
            kingPlace(i, j);
            if (i == 7) kingPlaced = true;
            else if (!(kingPlaced = tryToPlaceKings(i + 1))) kingDel(i,j);
        }
        if (kingPlaced) break;
    }
    return kingPlaced;
}
int main()
{
    for (int i = 0; i < 8; i++) for (int j = 0; j < 8; j++) board[i][j] = 0;
    tryToPlaceKings(0);
    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            if (board[i][j] == -1) cout << "[K]";
            else cout << "[ ]";
        }
        cout << endl;
    }
    return 0;
}
```

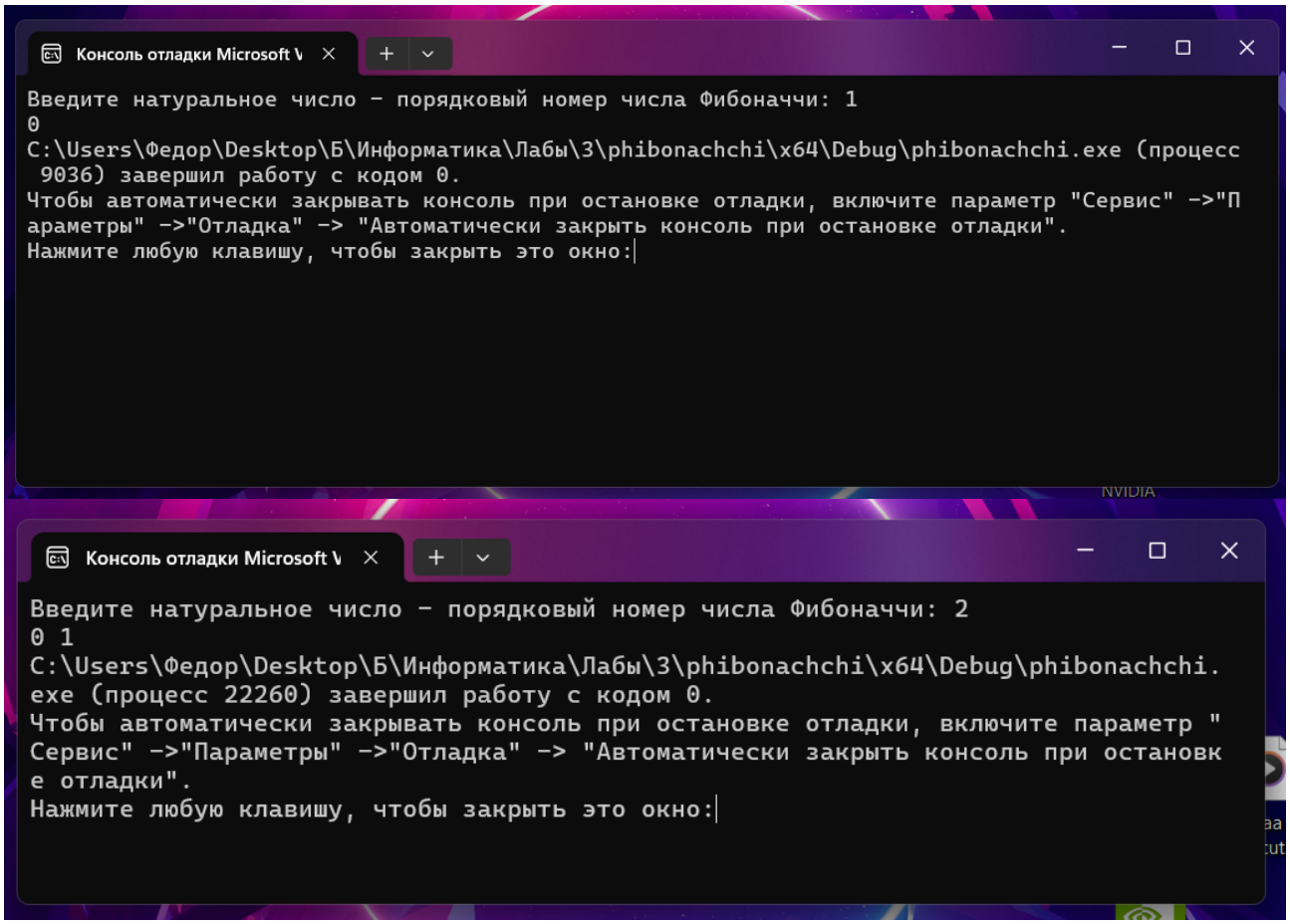
## 4.2 Скриншоты

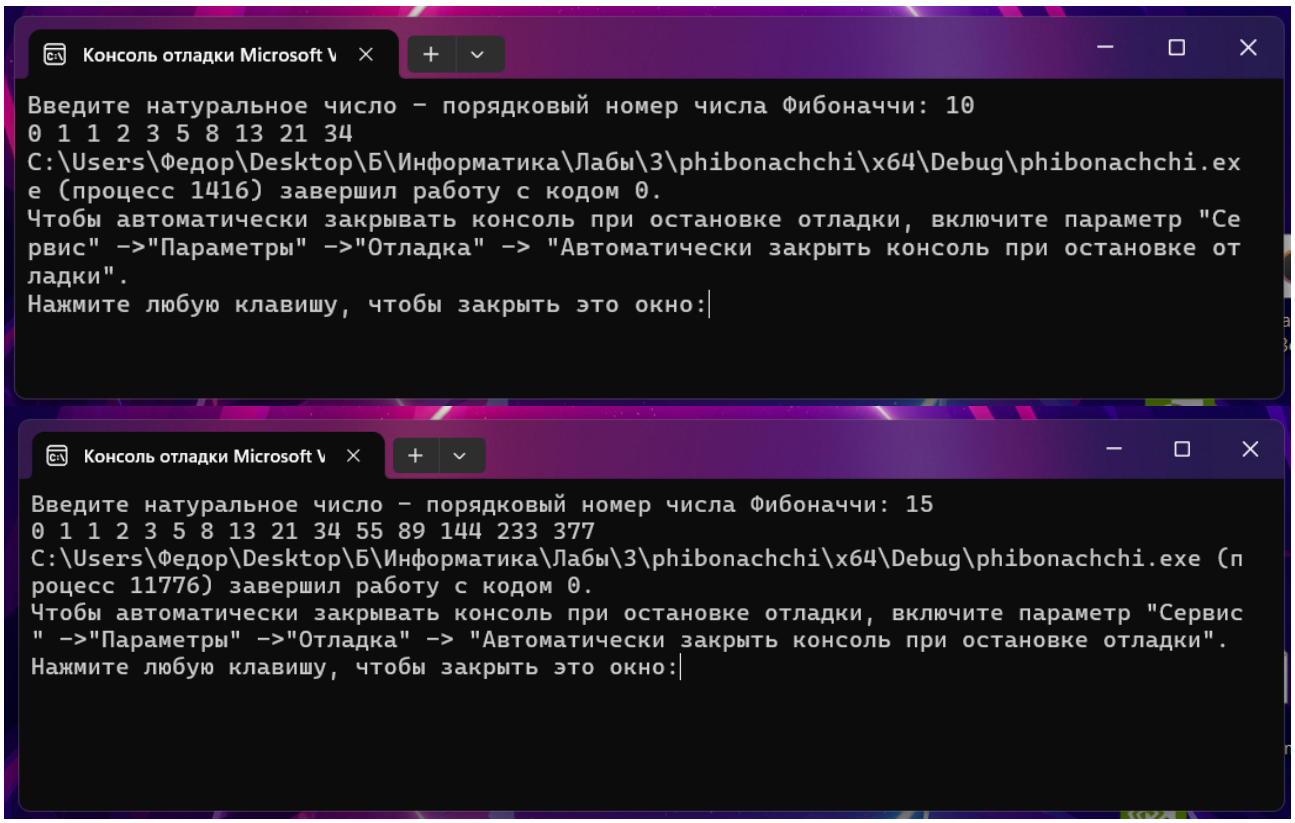
Запустим программы с разными входными данными, чтобы убедится, что они работают верно.

Задача 1.

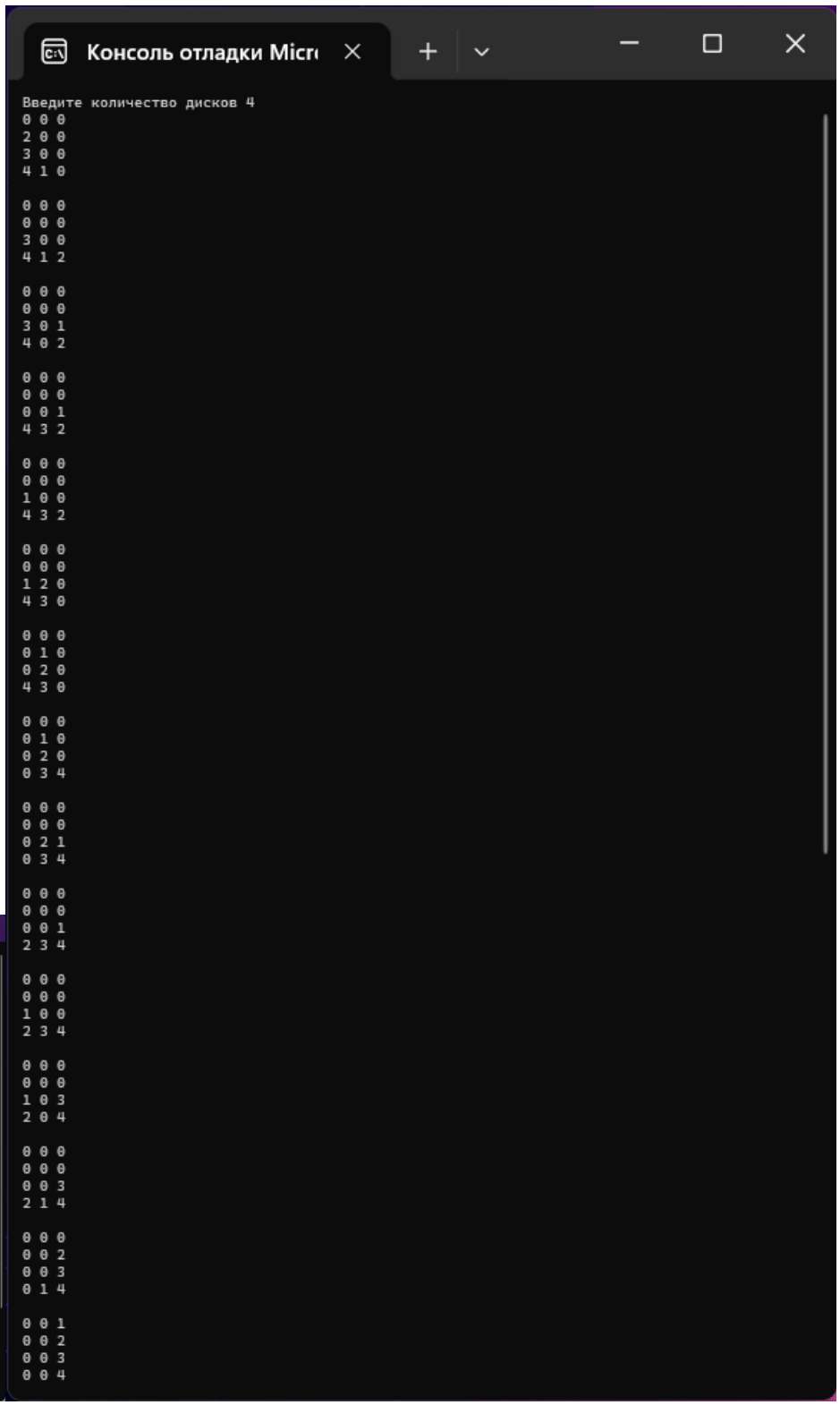


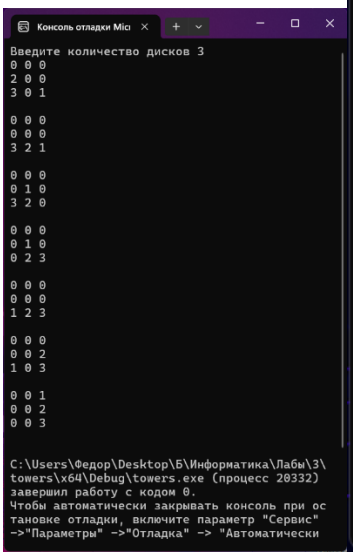

Задача 2.

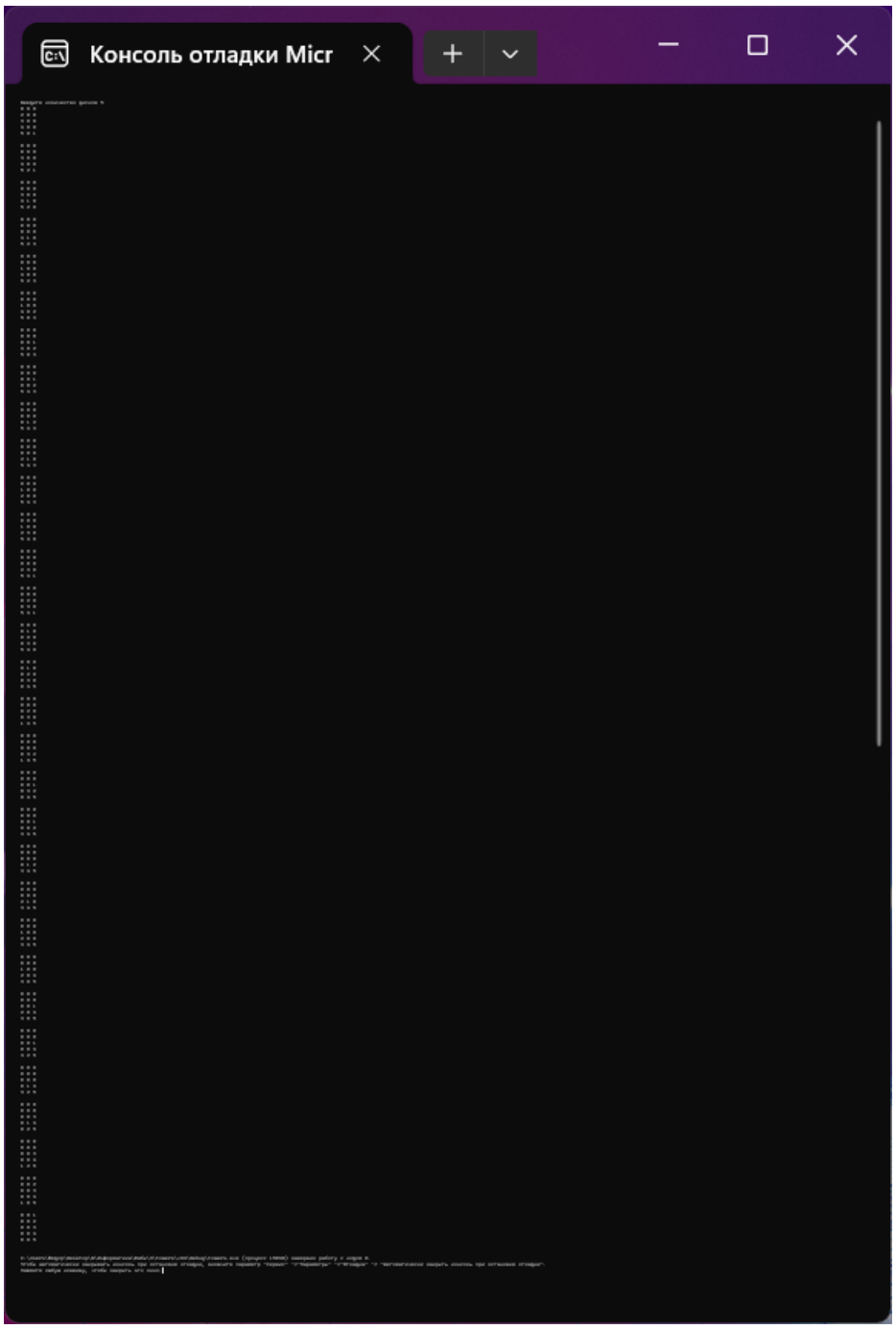



Консоль отладки Microsoft \
Введите натуральное число – порядковый номер числа Фибоначчи: 10
0 1 1 2 3 5 8 13 21 34
C:\Users\Федор\Desktop\Б\Информатика\Лабы\3\phibonachchi\x64\Debug\phibonachchi.exe (процесс 1416) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
Консоль отладки Microsoft \
Введите натуральное число – порядковый номер числа Фибоначчи: 15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
C:\Users\Федор\Desktop\Б\Информатика\Лабы\3\phibonachchi\x64\Debug\phibonachchi.exe (процесс 11776) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:

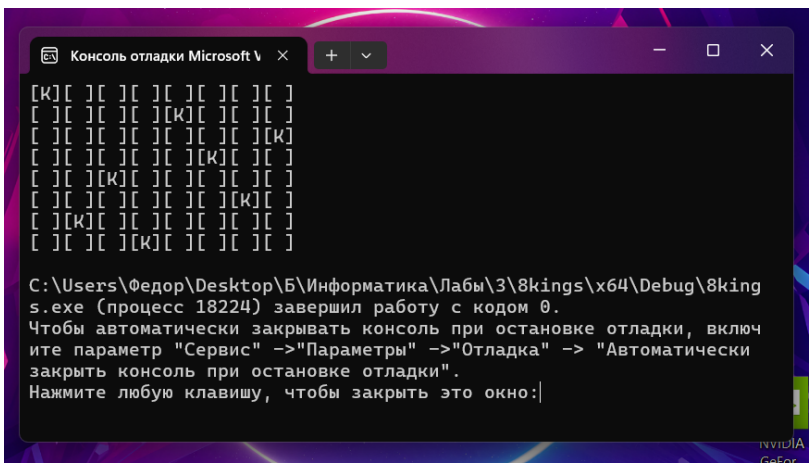
Задача 3.


Консоль отладки Micr
Введите количество дисков 4
0 0 0
2 0 0
3 0 0
4 1 0
0 0 0
0 0 0
3 0 0
4 1 2
0 0 0
0 0 0
3 0 1
4 0 2
0 0 0
0 0 0
0 0 1
4 3 2
0 0 0
0 0 0
1 0 0
4 3 2
0 0 0
0 0 0
1 2 0
4 3 0
0 0 0
0 1 0
0 2 0
4 3 0
0 0 0
0 1 0
0 2 0
0 3 4
0 0 0
0 0 0
0 2 1
0 3 4
0 0 0
0 0 0
0 0 1
2 3 4
0 0 0
0 0 0
1 0 0
2 3 4
0 0 0
0 0 0
1 0 3
2 0 4
0 0 0
0 0 0
0 0 3
2 1 4
0 0 0
0 0 2
0 0 3
0 1 4
0 0 1
0 0 2
0 0 3
0 0 4


Консоль отладки Micr
Введите количество дисков 3
0 0 0
2 0 0
3 0 1
0 0 0
0 0 0
3 2 1
0 0 0
0 1 0
3 2 0
0 0 0
0 1 0
0 2 3
0 0 0
0 0 0
1 2 3
0 0 0
0 0 2
1 0 3
0 0 1
0 0 2
0 0 3
C:\Users\Федор\Desktop\Б\Информатика\Лабы\3\towers\x64\Debug\towers.exe (процесс 20332) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически


Консоль отладки Micr

Задача 4.


Консоль отладки Microsoft V
[K][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][K][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][K]
[ ][ ][ ][ ][ ][ ][K][ ]
[ ][ ][K][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][K][ ]
[ ][K][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][K][ ][ ][ ][ ]
C:\Users\Федор\Desktop\Б\Информатика\Лабы\3\8kings\x64\Debug\8king
s.exe (процесс 18224) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включ
ите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически
закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:

# ВЫВОД

В итоге этой работы было составлено несколько программ с использованием рекурсии в различных целях. В ходе работы были получены навыки работы с рекурсивными функциями.

Проведенная лабораторная работа была опубликована в общий доступ по адресу: https://github.com/Fedor0000/TheUltimateFolder/tree/main/Sem_2/Labs/3