Министерство образования и науки РФ

ФГАОУ ВПО «Уральский федеральный университет имени   
первого президента России Б.Н.Ельцина»  
Институт радиоэлектроники и информационных технологий – РТФ  
Департамент информационных технологий и автоматики

Оценка работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЛЕСЕНКИ ИЗ БЛОКОВ**

Отчет по лабораторной работе №3  
по дисциплине «Методы исследования и моделирования  
информационных процессов и технологий»

Подпись Дата Ф.И.О.

Преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Добряк П.В.  
Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимошенкова Ю.С.

Группа РИМ-161203

Екатеринбург

2017

**Задача:** Вводится количество блоков. Нужно вывести все возможные лесенки из этих блоков. На каждом следующем уровне лесенки количество блоков должно быть меньше, чем на предыдущем.

**СОДЕРЖАНИЕ**

[1 Алгоритм вывода возможных лесенок 3](#_Toc477898734)

[2 Алгоритм подсчета возможных лесенок 4](#_Toc477898735)

[2.1 Способ 1 (рекурсия) 4](#_Toc477898736)

[2.2 Способ 2 5](#_Toc477898737)

[ВЫВОДЫ 6](#_Toc477898738)

# 1 Алгоритм вывода возможных лесенок

Код реализации на языке C++

// MIiMIPiT\_lab3.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include "stdio.h"

#include <iostream>

#include <list>

#include <vector>

using namespace std;

void stairs(int n)

{

struct MyStruct

{

int a[101];

};

list<MyStruct> A;

list<MyStruct> fin;

MyStruct N;

int \*arr;

arr = new int[n];

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

{

arr[i] = 0;

}

arr[0] = n;

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

N.a[i] = arr[i];

A.push\_back(N);

int j = 1;

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

{

for (auto c : A)

{

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

arr[i] = c.a[i];

if (arr[i] > 2)

{

arr[i] = arr[i] - 1;

arr[i + 1] = arr[i + 1] + 1;

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

N.a[i] = arr[i];

A.push\_back(N);

}

}

}

for (auto c : A)

{

int flag = 0;

for (int i = 0; i < n - 1; i++)

{

if (c.a[i] <= c.a[i + 1] && c.a[i])

flag = 1;

}

if (flag == 0)

fin.push\_back(c);

}

for (auto c : fin)

{

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

if (c.a[i] != 0)

cout << c.a[i] << " ";

cout << '\n';

}

};

int main()

{

int n = 10;

cout << "Kolichestvo blokov n = " << n << " " << "\n" << "Solutions:" << "\n";

stairs(n);

system("pause");

}

Результат работы алгоритма

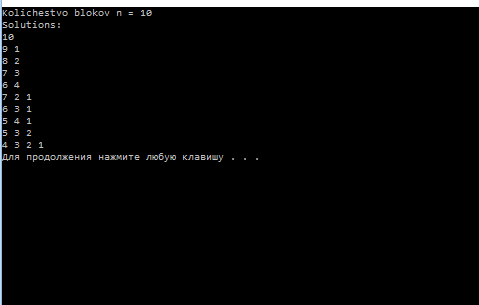
****

Рисунок 1 – Результат работы программы

# 2 Алгоритм подсчета возможных лесенок

## 2.1 Способ 1 (рекурсия)

Код функции на языке C++

int number\_from\_rec(int n)

{

int i, out;

if (n <= 2) out = 1;

else

{

out = 0;

for (i = 0; i \* 2 <= n; i++)

{

out += number\_from\_rec(i);

if (i \* 2 == n) out--;

}

}

return out;

}

}Результат работы алгоритма

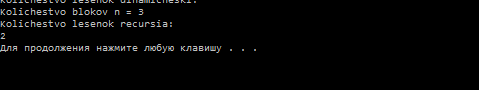
****

Рисунок 2 – Результат работы программы

## 2.2 Способ 2

Код функции на языке C++

int number\_from\_din(int n)

{

int m[101][101];

for (int i = 0; i < n + 1; i++)

{

m[0][i] = 0;

m[1][i] = 1;

}

for (int i = 0; i < n + 1; i++)

m[i][0] = 0;

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

{

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

{

if (i == 1)

break;

else if (i == j)

m[i][j] = m[i][j - 1] + 1;

else if (j > i)

m[i][j] = m[i][j - 1];

else

m[i][j] = m[i][j - 1] + m[i - j][j - 1];

}

}

return m[n][n];

}

Результат работы алгоритма



Рисунок 7 – Результат работы программы

# ВЫВОДЫ

В ходе выполнения лабораторной работы был реализован алгоритм составления лесенок из блоков, при условии, что на каждом следующем уровне количество блоков меньше, чем на предыдущем. Результат соответствует представленному материалу.

Также был реализован алгоритм подсчета количества лесенок. Которые можно составить из такого количества блоков двумя способами.