**Федеральное государственное автономное образовательное учреждение высшего образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

*Факультет социально-экономических и компьютерных наук*

ФИО студента

**ЛАБОРАТОРНАЯ РАБОТА №\_\_\_**

*тема*

студента образовательной программы «Разработка информационных систем для бизнеса» по направлению подготовки *09.03.04 Программная инженерия*

Руководитель

ученая степень, ученое

звание, должность

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

И.О. Фамилия

Пермь, 202\_ год

Оглавление

[1 Задание №1 3](#_Toc115539229)

[1.1.1 Общая постановка задачи 3](#_Toc115539230)

[2.1.1 Частная постановка задачи 3](#_Toc115539231)

[1.2 Анализ 3](#_Toc115539232)

[1.3 Проектирование 6](#_Toc115539233)

[1.4 Разработка 8](#_Toc115539234)

[1.5 Тестирование 9](#_Toc115539235)

[2 Задание №2 12](#_Toc115539236)

[1.1.1 Общая постановка задачи 12](#_Toc115539237)

[2.1.1 Частная постановка задачи 12](#_Toc115539238)

[2.2 Анализ 12](#_Toc115539239)

[2.3 Проектирование 14](#_Toc115539240)

[2.4 Разработка 15](#_Toc115539241)

[2.5 Тестирование 16](#_Toc115539242)

[3 Задание №3 20](#_Toc115539243)

[1.1.1 Общая постановка задачи 20](#_Toc115539244)

[2.1.1 Частная постановка задачи 20](#_Toc115539245)

[3.2 Анализ 20](#_Toc115539246)

[3.3 Проектирование 20](#_Toc115539247)

[3.4 Разработка 22](#_Toc115539248)

# Задание №1

Первое задание посвящено получению навыков работы с операторами языка C#. Оно проверяет понимание операций инкремента и декремента, учит анализировать выражение.

## Общая постановка задачи

1. Для задачи 1 найти значения выражений, обработав исходные данные.
2. Составить систему тестов и вычислить полученное выражение для нескольких значений Х.
3. Результаты всех вычислений вывести на печать.
4. Объяснить полученные результаты.

## Частная постановка задачи

Посчитать значения выражений 1-3, на введенном наборе n и m сразу для 3-х заданий. Стоит отметить, что значения m и n будут меняться за счет функций декремента и экскремента, поэтому после подсчета каждого выражения важно выводить на экран промежуточные значения m и n. В задании есть деление, поэтому отработать варианты деления на ноль.

Конкретные задачи:

1) n++/--m;

2) n-->n/m++;

3) m<n++.

Для 4 пункта посчитать значение данного выражения. Проверить существует ли область допустимых значений.

4) res 4 = 

## Анализ

В данном задании пользователь вводит переменные n, m типа int. И переменную x типа double.

Анализ классов входных и выходных данных приведен в таблицах 1, 2, 3. Для каждой переменной создана своя таблица.

Стандартные выходные данные для первых 3-х заданий (ожидаемый вывод):

1) int;

2) bool;

3) bool.

Таблица 1 – Классы входных и выходных данных для переменной n

| **Классы входных данных** | **Условия** | **Классы выходных данных** |
| --- | --- | --- |
| *n* – строка | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *n* – вещественное число или целое за пределами типа Int32 | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *n –*любое целое число в пределах типа Int32 | Программа выполняется | Стандартные выходные данные |

Похожая таблица представлена ниже. У переменной m больше классов входных данных, так как рассматривается возможность деления на 0, которое приводит к ошибке.

Таблица 2 – Классы входных и выходных данных для переменной m

| **Классы входных данных** | **Условия** | **Классы выходных данных** |
| --- | --- | --- |
| *m* – строка | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *m* – вещественное число или целое за пределами типа Int32 | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *m* = 1 | В первом выражении попытка деления на 0 | Сообщение об ошибке из-за деления на 0 |
| *m –*любое целое число, не равное 1 и не выходящее за пределы типа Int32 | Программа выполняется | Стандартные выходные данные |

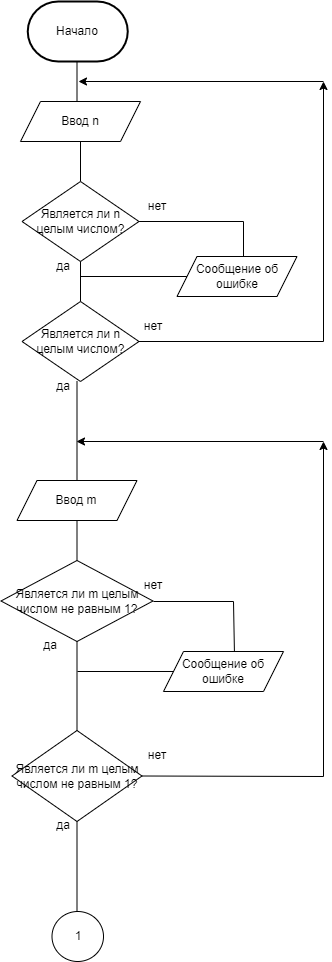
В 4-м задании вводится переменная x типа double. Ожидаемый, стандартный вывод для 4-го задания – число типа double. Ниже приведена таблица анализа классов входных и выходных данных для переменной x. Так как кубический корень не имеет области допустимых значений, по подкоренное выражение может быть любым. Также переменная x может быть любым вещественным числом, поэтому другие критерии (кроме принадлежности типу double) в таблице 3 не рассматриваются. Замечу, что в качестве sin(x) используется число в радианах. Стоит отметить, что для правильного вычисления кубического корня стоит рассмотреть знак подкоренного выражения, если он отрицательный, то взять подкоренное выражение с модулем и впереди написать минус.

Таблица 3 – Классы входных и выходных данных для переменной x

| **Классы входных данных** | **Условия** | **Классы выходных данных** |
| --- | --- | --- |
| *x* – строка или вещественное число за пределами типа double | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *x* >= 0 | Вычисление выражения | Стандартные выходные данные |
| *x <*0 | Вычисление выражения от модуля числа | Стандартные выходные данные от модуля числа со знаком минус |

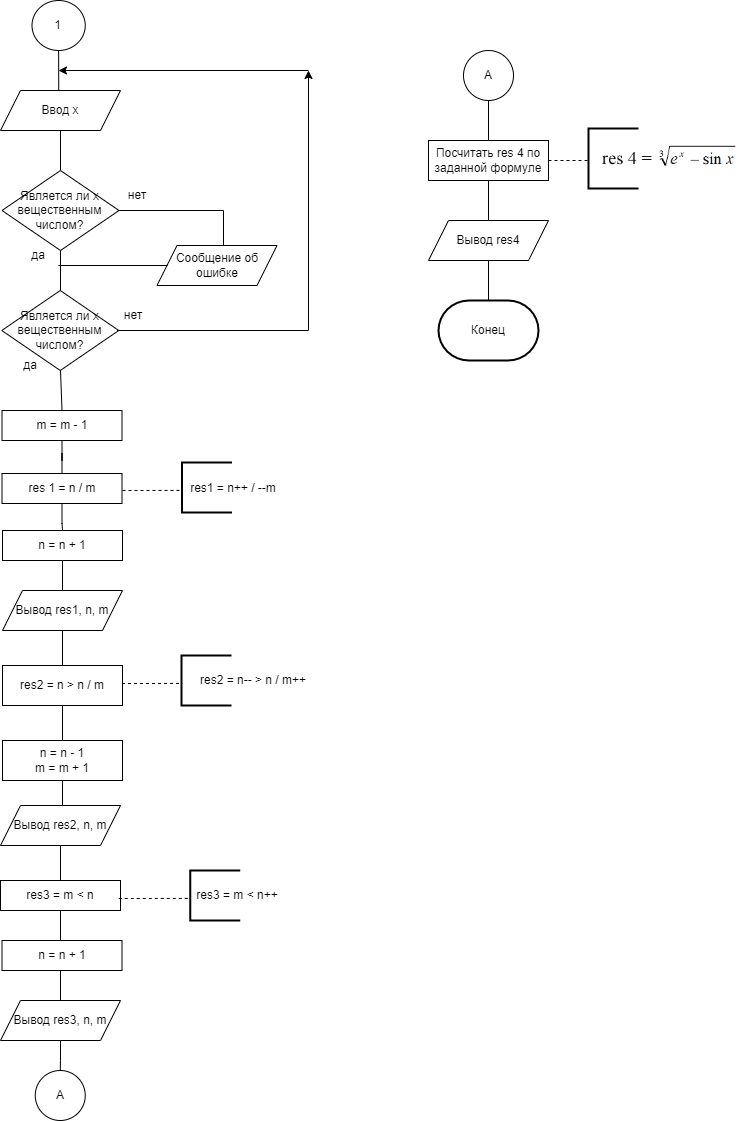
## Проектирование

Для дальнейшей реализации проектирования необходимо составить блок схему, она представлена ниже на рисунке 1.1.



***Рисунок 1.1 – Схема алгоритма для задачи 1***

Продолжение блок-схемы представлено на рисунке 1.2.



***Рисунок 1.2 – Продолжение схемы алгоритма для задачи 1***

## Разработка

Разработка осуществляется с помощью листинга программы.

Ниже представлен код. Переменная isSucceed отвечает за правильность введенного типа данных (для ввода всех трех переменных). canDel относится к переменной m и при определенном значении m (m=1) выполняется деление на 0. Переменные res1, res2, res3, res4 отвечают за соответственные результаты 1-4 выражений.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

…

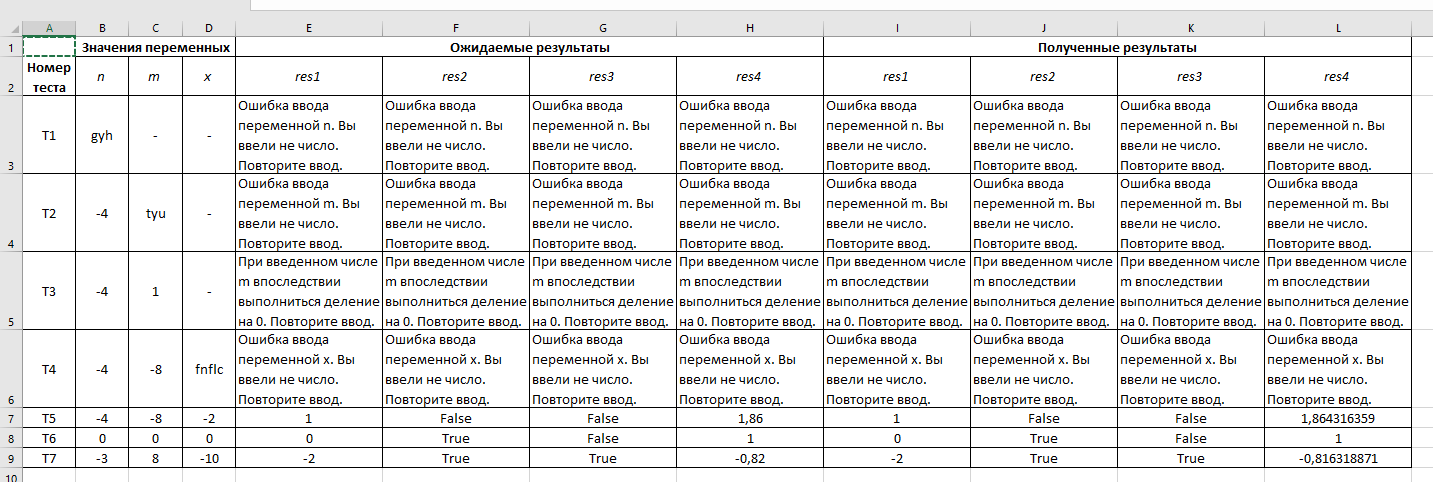
}

}

}

## Тестирование

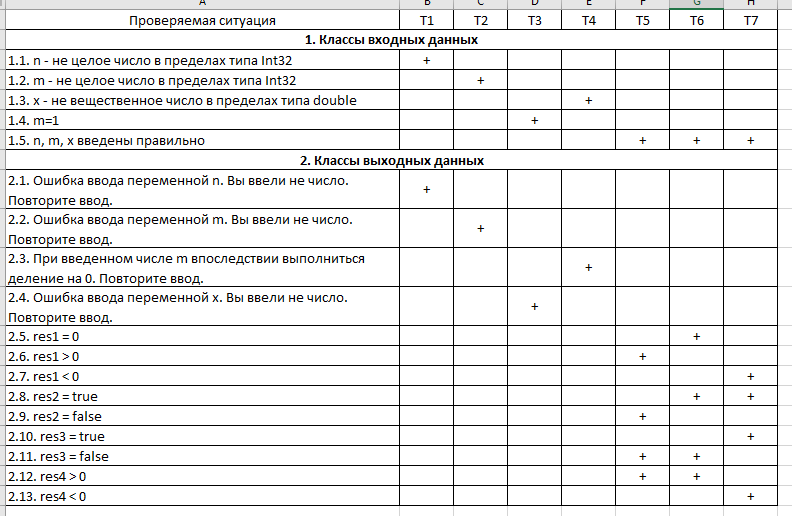
Результаты тестирования представлены на рисунке 1.5.1. Первые 4 теста выполняются не полностью из-за ошибок ввода различных переменных. По рисунку видно, что все ожидаемые результаты совпадают с реальными.



***Рисунок 1.5.1. – Таблица тестов для задания 1***

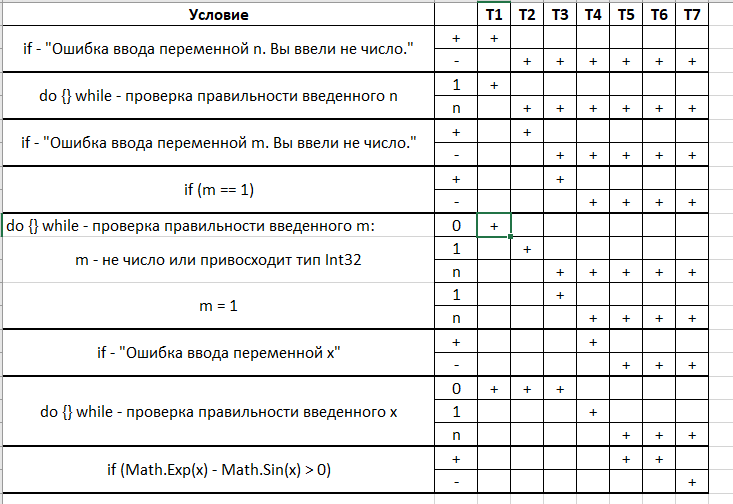
Таблица тестов составлялась с помощью дополнительных таблиц по критериям черного и белого ящиков. Они представлены ниже.

Рисунок 1.5.2. содержит таблицу с критериями черного ящика. Она состоит из классов входных и выходных данных, рассмотренных в анализе.



***Рисунок 1.5.2. – Таблица критериев черного ящика задания 1***

Также тестирование проводилось по критериям белого ящика. Стоит отметить, что в тестах 1-4 программа не дойдет до каких-либо условий. Так как циклы, отвечающие за проверку правильного ввода, не дадут это сделать. Таблица критериев белого ящика представлена на рисунке 1.5.3. (см. ниже).



***Рисунок 1.5.3. – Таблица критериев белого ящика задания 1***

# Задание №2

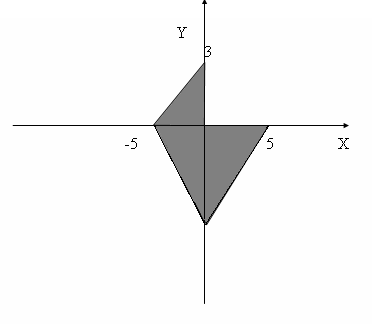
Второе задание посвящено получение навыков работы с операторами языка C# и работы со сложными условиями (с помощью и/или). Также задание учит определять правильные тесты и анализировать типы входных данных.

## Общая постановка задачи

1. Для задачи 2 записать выражение, зависящее от координат точки X1 и Y1 и принимающее значение TRUE, если точка принадлежит заштрихованной области, и FALSE, если не принадлежит.
2. Составить систему тестов и вычислить полученное выражение.
3. Результаты всех вычислений вывести на печать.
4. Объяснить полученные результаты.

## Частная постановка задачи

Частная постановка задачи представлена на рисунке 2.1. Нужно найти попадает ли точка с заданными пользователем координатами в заштрихованную область. Все граница включены в заштрихованную область.



***Рисунок 3.1 – Заданная область***

Представлена фигура – ромб, ограниченный линиями во 2, 3 и 4 четверти. Любая точка в 1 координатной области (кроме границ) не включена в координатную область.

## Анализ

В данном задании вводятся 2 переменные типа double, результат получается в виде логической переменной. Если анализировать график области, то для каждой четверти (кроме 1) можно описать свою прямую пересечение которой с осями координат образует треугольник, где лежат все точки области. Для 2 четверти: y = 3/5\*x + 3. Для 3 четверти: y = - 6/5\*x – 6. Так как значение нижней точки не определено, я взяла ее равной -6. Для 4 четверти: y = 6/5\*x – 6. Таким образом, с помощью логический операторов вычисление переменной (также ожидаемый вывод) выглядит так:

bool pointInShape = (x <= 0 && y >= 0 && y <= 3.0 / 5 \* x + 3) || (x >= 0 && y <= 0 && y >= 6 / 5.0 \* x - 6) || (x <= 0 && y <= 0 && y >= -6 / 5.0 \* x - 6) || (y == 0 && x >= 0 && x <= 5 || x == 0 && y >= 0 && y <= 3);

Отмечу, что в данном условии точки внутри области, лежащие в 1 четверти уже учтены, так как знаки сравнения не строгие.

Чтобы создать понятный пользователю ответ программы, можно посчитать bool border – переменная отвечающая на вопрос попадает ли точка на границу области. Рассчитывается по данной формуле:

bool border = (x <= 0 && y >= 0 && y == Math.Round((0.6 \* x + 3), 4)) || (x >= 0 && y <= 0 && y == Math.Round((1.2 \* x - 6), 4)) || (x <= 0 && y <= 0 && y == Math.Round((-1.2 \* x - 6), 4)) || (y==0 && x >= 0 && x <= 5 || x==0 && y >= 0 && y <= 3);

Рассмотрим таблицу классов входных данных. Для переменных x и y она полностью идентична, поэтому рассмотрю данные переменные в одной таблице, представленной ниже.

Таблица 4 – Классы входных и выходных данных для переменных x, y

| **Классы входных данных** | **Условия** | **Классы выходных данных** |
| --- | --- | --- |
| *x* или *y* – строки или вещественные числа за пределами типа double | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *x* и *y* – вещественные числа | Вычисление логического выражения | Ожидаемый вывод |

## Проектирование

Для дальнейшей реализации проектирования необходимо составить блок схему, она представлена ниже на рисунке 2.

…

***Рисунок 2– Схема алгоритма для задачи 2***

## Разработка

Разработка осуществляется с помощью листинга программы.

Ниже представлен код. Переменная isSucceed отвечает за правильность введенного типа данных (для ввода двух переменных). Переменная bool pointInShape отвечает за полученный результат. Переменные p2, p3, p4 являются частями pointInShape, с их помощью искала ошибки, в программе дальше не использую. is1, is2, is3, is4 – логические переменные относительно принадлежности точки какой-либо четверти. Переменная border отвечает за принадлежность точки границам области.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

internal class Program

{

static void Main(string[] args)

{

…

}

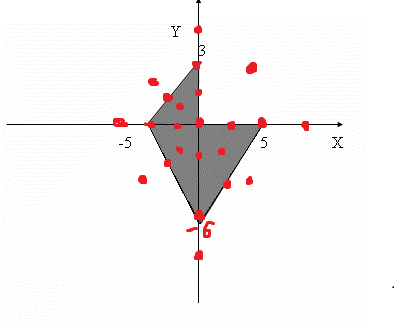
}

}

## Тестирование

Чтобы проверить все возможные варианты, нужно рассмотреть точки лежащие в каждой из предложенных областей, также проверить точки на границах области (для этого имеется отдельный вывод). Для каждой точки, где нельзя однозначно определить четверть предусмотрен отдельный вывод, в зависимости от того попадает точка в область или нет.

Тестируемые точки представлены на рисунке 2.5.1.



***Рисунок 2.5.1. – Тестируемые точки для задачи 2***

Таблица тестов представлена ниже, из 25 тестов большинство является правильно введенными числами и проверяют в какой именно области находится точка. Тесты 23, 24 проверяют неправильно введенную переменную (поэтому до части проверки критериев белого ящика эти тесты не доходят). В столбце четверть часто стоит “-”, это означает, что точка лежит на пересечении четвертей, и нельзя однозначно определить к какой четверти она относится.

Первая половина тестов представлена на рисунке 2.5.2.



***Рисунок 2.5.2. – Тесты 1-12 задачи 2***

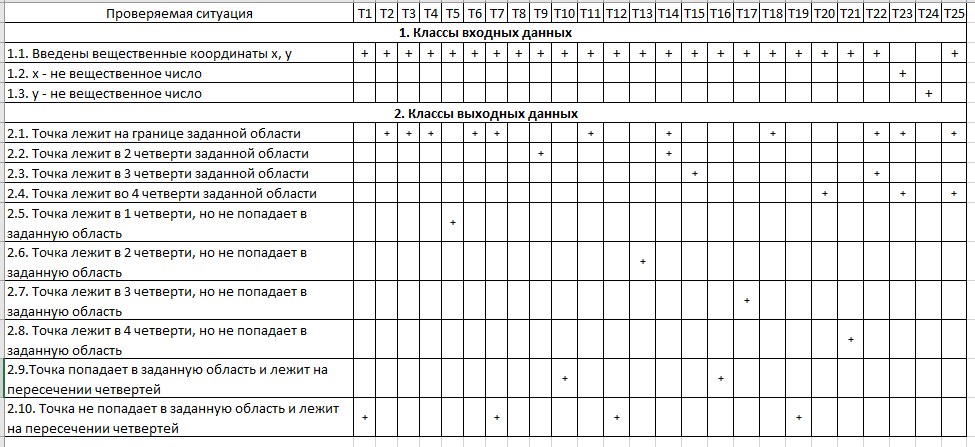
Вторая половина тестов представлена на рисунке 2.5.3.



***Рисунок 2.5.3. – Тесты 13-25 задачи 2***

Ниже приведены критерии черного ящика, рассмотренные по классам входных и выходных данный. В тестах 14, 22, 25 выводится 2 сообщения, что точки принадлежат и границам, и определенной четверти заданной области.

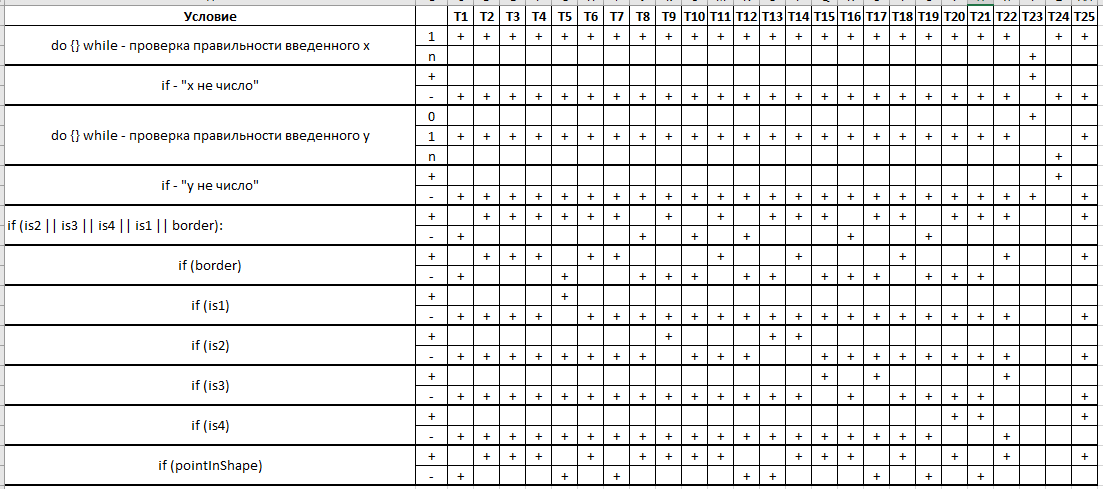
На рисунке 2.5.4. представлено тестирование по критериям черного ящика.



***Рисунок 2.5.4. – Тестирование по критериям чёрного ящика задачи 2***

Ниже приведены критерии белого ящика. В программе имеется несколько сложных условий, они разделены с помощью простых и собрать их можно с помощью плюсиков в таблице.

Таблица с тестированием по критериям белого ящика представлена на рисунке 2.5.5.



***Рисунок 2.5.5. – Тестирование по критериям белого ящика задачи 2***

# Задание №3

Третье задание посвящено работе с различными типами данных в языке C#. Оно показывает разницу в вычислениях между типами float и double, учит анализировать, почему так получается.

## Общая постановка задачи

1. Для задачи 3 вычислить значение выражения, используя различные вещественные типы данных (float и double).
2. Результаты всех вычислений вывести на печать.
3. Объяснить полученные результаты.

## Частная постановка задачи

Посчитать значение выражения  в разных типах данных при заданных константных значениях a = 1000 и b = 0.0001.

## Анализ

Входных данных нет, так как переменные объявлены в виде констант. В качестве выходных данных получится число вещественного типа. Стоит отметить, что результаты в типе float и double будут не одинаковыми, потому что в какой-то момент тип float (за счет 4 битов вместо 8 типа double) обрежет часть числа и результат получится совершенно другой.

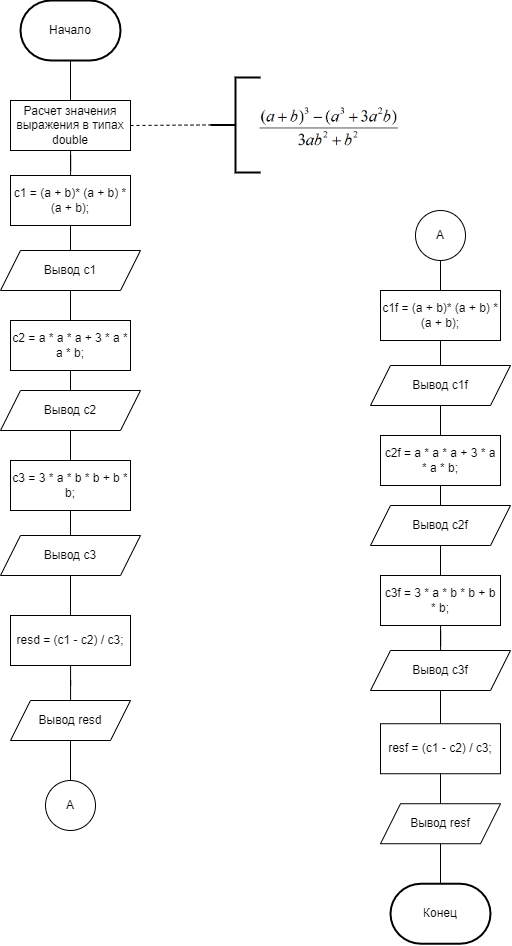
Так как в задании содержаться ФСУ, данная формула после раскрытия скобок и сокращения принимает вид , что при подстановке констант равно чуть меньше единицы.

## Проектирование

Для дальнейшей реализации проектирования необходимо составить блок схему, она представлена ниже на рисунке 3. Вспомогательные переменные для c1, c2, c3 для типа double и c1f, c2f, c3f для типа float нужны, чтобы понять в каком месте значения в типе float отличаются от double.

Отмечу, что в блок-схеме представлено выведение каждого промежуточного результата после вычислений (с1, с2, итд). Это создано с целью понимания, в каком именно месте программа начинает выдавать неправильный результат из-за выхода за пределы типа данных.

Блок схема для данной задачи на рисунке 3.



***Рисунок 3 – Схема алгоритма***

## Разработка

Разработка осуществляется с помощью листинга программы. Ниже представлен код. Вычисления получаются разными, потому что у числа c1f пропадает дробная часть (за счет переполнения типа float). Поэтому конечные результаты отличаются. Ниже приведен результат работы программы.

c1 = 1000000300,00003

c2 = 1000000300

c3 = 3,001E-05

double: 0,997052038562016

c1f = 1E+09

c2f = 1E+09

c3f = 3,001E-05

float: 2132623

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. namespace ConsoleApp3
7. {
8. internal class Program
9. {
10. static void Main(string[] args)
11. {
12. …
13. }
14. }
15. }