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

*Факультет социально-экономических и компьютерных наук*

Панфилов Даниил Сергеевич

**ЛАБОРАТОРНАЯ РАБОТА №1**

*Выполнение простой программы. Организация ввода и вывода данных*

студента образовательной программы «Разработка информационных систем для бизнеса» по направлению подготовки *38.03.05 Бизнес-информатика*

Руководитель

ученая степень, ученое

звание, должность

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

И.О. Найданов

Пермь, 2023 год

Оглавление

[1. Задание №1 3](#_Toc148200665)

[1.1.1 Общая постановка задачи 3](#_Toc148200666)

[1.1.2 Частная постановка задачи 3](#_Toc148200667)

[1.2 Анализ 3](#_Toc148200668)

[1.3 Проектирование 6](#_Toc148200669)

[1.4 Разработка 7](#_Toc148200670)

[1.5 Тестирование 8](#_Toc148200671)

[2. Задание №2 10](#_Toc148200672)

[2.1.1. Общая постановка задачи 10](#_Toc148200673)

[2.1.2 Частная постановка задачи 10](#_Toc148200674)

[2.2 Анализ 10](#_Toc148200675)

[2.3 Проектирование 12](#_Toc148200676)

[2.4 Разработка 13](#_Toc148200677)

[2.5 Тестирование 13](#_Toc148200678)

[3. Задание №3 16](#_Toc148200679)

[3.1.1. Общая постановка задачи 16](#_Toc148200680)

[3.1.2 Частная постановка задачи 16](#_Toc148200681)

[3.2 Анализ 16](#_Toc148200682)

[3.4. Разработка 18](#_Toc148200683)

# 1. Задание №1

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

## 1.1.1 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.

## 1.1.2 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 инкремента, поэтому после подсчета каждого выражения важно выводить на экран промежуточные значения m и n.

Задачи:

1) ++n \* ++m;

2) m++ < n;

3) n++ > m.

Для пункта 4 следует посчитать значение данного выражения. В задании есть деление, поэтому отработать варианты деления на ноль.

4) 

## 1.2 Анализ

В данном задании пользователь вводит переменные n, m, 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) double;

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 | Программа выполняется | Стандартные выходные данные |
| *n –*любое целое число | Программа выполняется | Стандартные выходные данные |

Идентичная таблица представлена ниже. У переменной m столько же классов входных данных, как у переменной n.

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 | Программа выполняется | Стандартные выходные данные |
| *m –*любое целое число | Программа выполняется | Стандартные выходные данные |

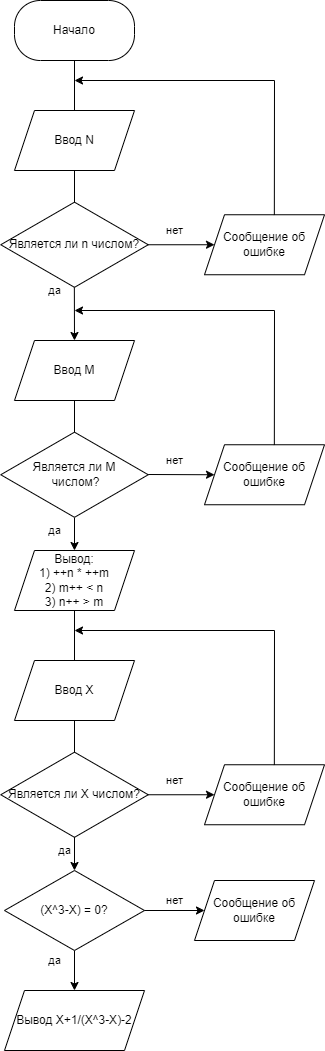
В 4-м задании вводится переменная x типа double. Ожидаемый, стандартный вывод для 4-го задания – число типа double. Ниже приведена таблица анализа классов входных и выходных данных для переменной x. Так как в данном задании производится операция деления, то нужно исключить вариант, когда знаменатель равен 0.

Таблица 3 – Классы входных и выходных данных для переменной x

| **Классы входных данных** | **Условия** | **Классы выходных данных** |
| --- | --- | --- |
| *x* – строка | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *x,* когда (x^3 - x) == 0 | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *x* – вещественное число, но (x^3 – x !)=0 | Программа выполняется | Стандартные выходные данные |
| *x –*любое целое число но (x^3 – x !)=0 | Программа выполняется | Стандартные выходные данные |

## 1.3 Проектирование

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



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

## 1.4 Разработка

Разработка осуществлялась с помощью написания программы.

Ниже приведён код. Переменные IsNCorrect, IsMCorrect и IsXCorrect отвечают за правильность введённого типа данных (для всех трёх переменных).

using System;

using System.Collections.Generic;

using System.Collections.Specialized;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лаба\_1

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите число N");

string inputN;

bool IsNCorrect;

string inputM;

bool IsMCorrect;

do

{

inputN = Console.ReadLine();

IsNCorrect = double.TryParse(inputN, out double N);

if (!IsNCorrect)

{

Console.WriteLine("Пожалуйста, введите число");

}

else

Console.WriteLine("Введите число M");

do

{

inputM = Console.ReadLine();

IsMCorrect = double.TryParse(inputM, out double M);

if (!IsMCorrect)

{

Console.WriteLine("Пожалуйста, введите число");

}

else

{

Console.WriteLine("1) " + (++N \* ++M));

Console.WriteLine("2) " + (M++ < N));

Console.WriteLine("3) " + (N++ > M));

}

} while (!IsMCorrect);

} while (!IsNCorrect);

Console.WriteLine();

Console.WriteLine("Введите Х");

string inputX;

bool isXCorrect;

do

{

inputX = Console.ReadLine();

isXCorrect = double.TryParse(inputX, out double X);

if (!isXCorrect)

{

Console.WriteLine("Напишите число");

}

else

if ((X \* X \* X - X) != 0)

{

Console.WriteLine("4) " + (X + (1 / (X \* X \* X - X)) - 2));

}

else Console.WriteLine("Введите, пожалуйста, другое число, которое не даёт в знаменателе 0");

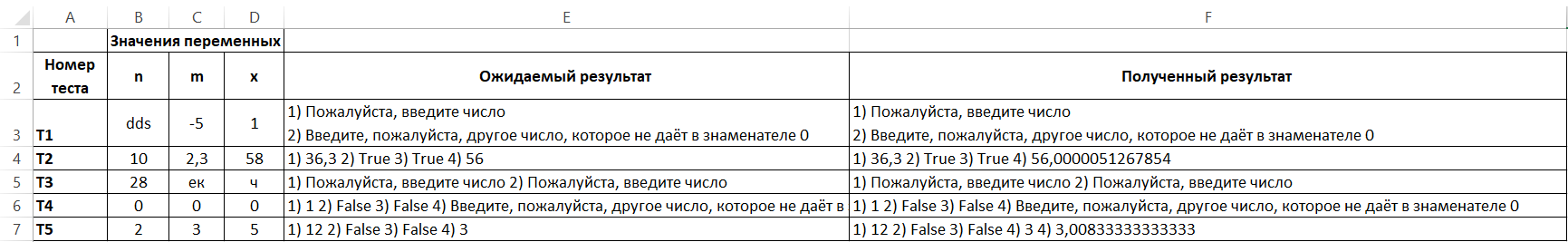
} while (!isXCorrect);

}

}

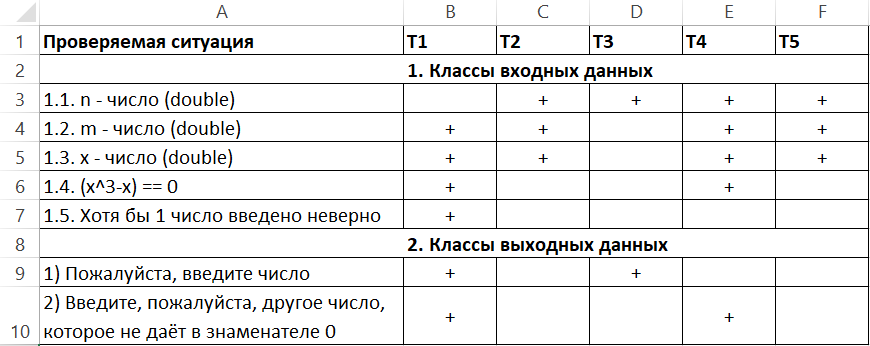
}

## 1.5 Тестирование

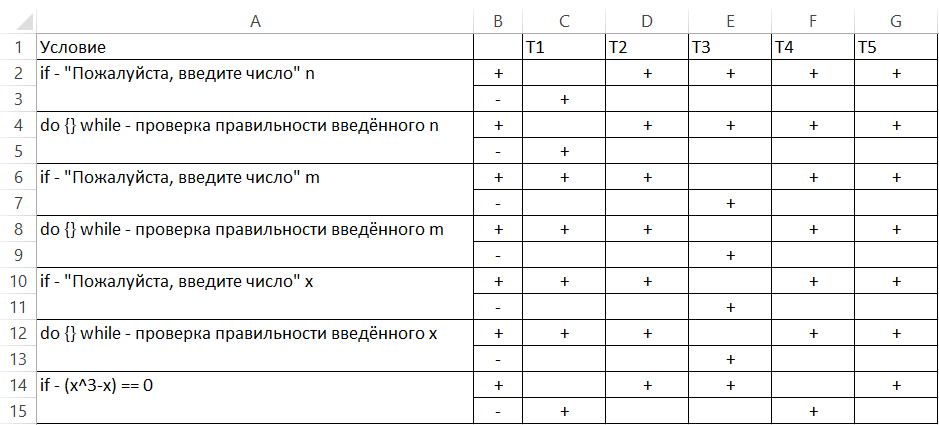
Результаты тестирования представлены на рисунке 1.5.1. По рисунку видно, что все ожидаемые результаты совпадают с реальными.

***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***

# 2. Задание №2

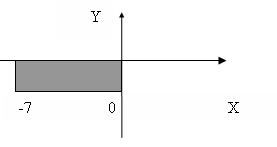
Второе задание посвящено получение навыков работы с условиями языка C# и работы со сложными условиями (с помощью и/или). Также задание учит определять правильные тесты и анализировать типы входных данных.

## 2.1.1. Общая постановка задачи

1. Для задачи 2 записать выражение, зависящее от координат точки X1 и Y1 и выводящее на экран сообщение, о том, что точка принадлежит заштрихованной области или не принадлежит.
2. Составить систему тестов и вычислить полученное выражение.
3. Результаты всех вычислений вывести на печать.
4. Объяснить полученные результаты.

## 2.1.2 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 2.1.1 – Заданная область***

Представлена фигура – прямоугольник, ограниченный линиями в 3 четверти. Любая точка в 1, 2 и 3 координатных областях (кроме границ) не включена в координатную область.

## 2.2 Анализ

В данном задании вводятся 2 переменные типа double, результат получается в виде сообщения о попадании/ не попадании точки в указанную область. Если анализировать график, то можно заметить, что для получения результата достаточно 4 условия. А именно: -7<= x <=0 и -2<= y <=0. Таким образом при помощи условия “if” получение результата выглядит так:

if (X >= -7 && X <= 0 && Y <= 0 && Y >= -2)

{

Console.WriteLine("Точка попадает в указанную область");

}

else Console.WriteLine("Точка не попадает в указанную область");

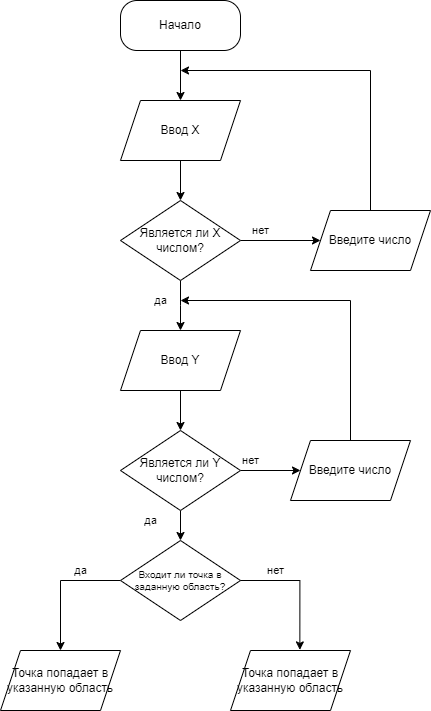
Стоит отметить, что в данном условии учитываются точки на координатных прямых, так как неравенства нестрогие.

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 |

## 2.3 Проектирование

Для дальнейшей реализации проектирования необходимо составить блок схему, она представлена ниже на рисунке 2.3.



***Рисунок 2.3 – Схема алгоритма для задачи 2***

## 2.4 Разработка

Разработка осуществляется с помощью листинга программы.

Ниже представлен код. Переменная IsXCorrect и IsYCorrect отвечает за правильность введенного типа данных (для ввода двух переменных).

namespace задание\_2

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите число X");

string inputX;

bool IsXCorrect;

string inputY;

bool IsYCorrect;

do

{

inputX = Console.ReadLine();

IsXCorrect = double.TryParse(inputX, out double X);

if (!IsXCorrect)

{

Console.WriteLine("Пожалуйста, введите число");

}

else

{

Console.WriteLine("Введите число Y");

do

{

inputY = Console.ReadLine();

IsYCorrect = double.TryParse(inputY, out double Y);

if (!IsYCorrect)

{

Console.WriteLine("Пожалуйста, введите число");

}

else

{

if (X >= -7 && X <= 0 && Y <= 0 && Y >= -2)

{

Console.WriteLine("Точка попадает в указанную область");

}

else Console.WriteLine("Точка не попадает в указанную область");

}

} while (!IsYCorrect);

}

} while (!IsXCorrect);

}

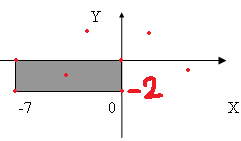
}

}

## 2.5 Тестирование

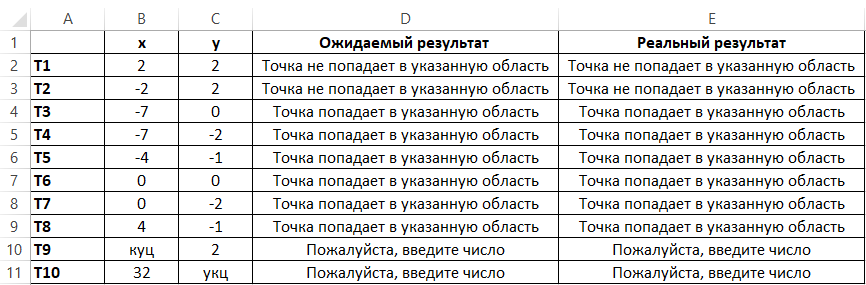
Чтобы проверить все возможные варианты, нужно рассмотреть точки, лежащие в каждой из предложенных областей, также проверить точки на границах области.

Тестируемые точки представлены на рисунке 2.5.1



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

Результаты тестирования представлены на рисунке 2.5.2. Первые 8 тестов проверяют находится ли точка в указанной области. Тесты 9, 10 проверяют неправильно введённую переменную. По рисунку видно, что все ожидаемые результаты совпадают с реальными.



***Рисунок 2.5.2 – Тесты задачи 2***

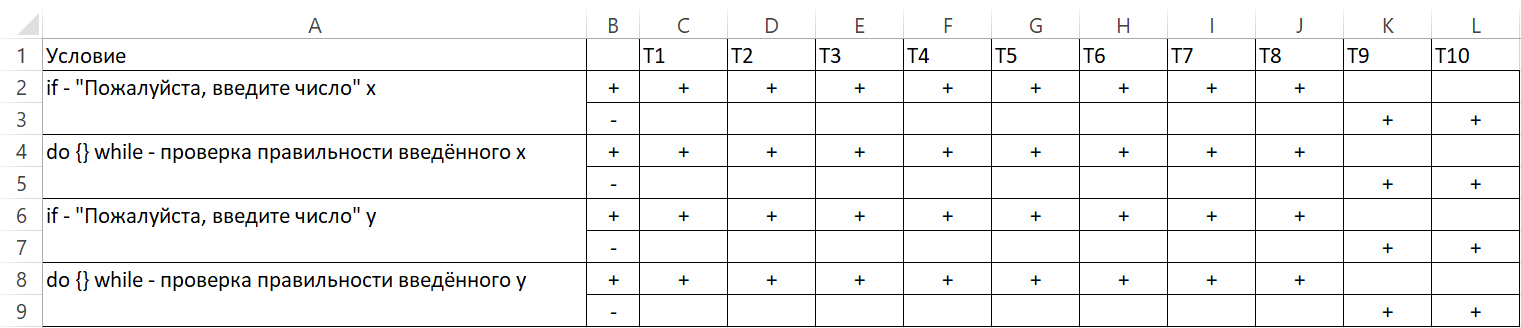
Ниже приведены критерии черного ящика, рассмотренные по классам входных и выходных данный.

На рисунке 2.5.3. представлено тестирование по критериям черного ящика.



***Рисунок 2.5.3 –Чёрный ящик 2***

Ниже приведены критерии белого ящика. Ниже приведены критерии белого ящика.



***Рисунок 2.5.4 – Критерии белого ящика 2***

# 3. 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.

## 3.1.1. 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.

## 3.1.2 Частная постановка задачи

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

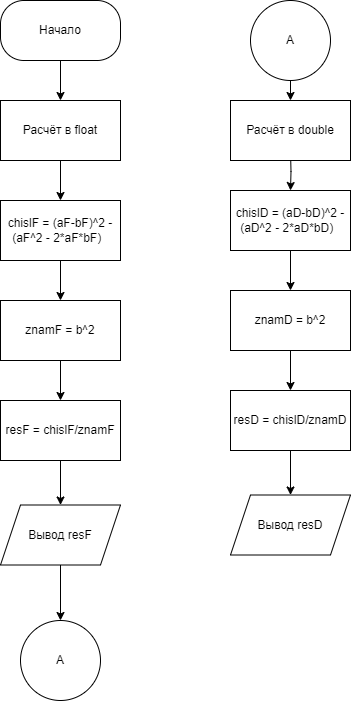
## 3.2 Анализ

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 обрежет часть числа и результат получится совершенно другой.

Для дальнейшей реализации проектирования необходимо составить блок схему, она представлена ниже на рисунке 3.2. Вспомогательные переменные для chislF, znamF, resF типа float и chislD, znamD, resD 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 (chislF, chislD, и тд). 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 3.2.



***Рисунок 3.2. – Схема алгоритма***

## 3.4. Разработка

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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace задание\_3

{

internal class Program

{

static void Main(string[] args)

{

float aF = 1000;

float bF = 0.0001f;

float chislF = (float)((float)Math.Pow(aF - bF, 2) - ((float)Math.Pow(aF, 2) - 2 \* aF \* bF));

float znamF = (float)Math.Pow(bF, 2);

float resF = chislF / znamF;

Console.WriteLine($"Результат с точностью до float: {resF}");

double aD = 1000;

double bD = 0.0001f;

double chislD = (double)((double)Math.Pow(aD - bD, 2) - ((double)Math.Pow(aD, 2) - 2 \* aD \* bD));

double znamD = (double)Math.Pow(bD, 2);

double resD = chislD / znamD;

Console.WriteLine($"Результат с точностью до double: {resD}");

}

}

}

Вывод:

Результат с точностью до float: 1250000

Результат с точностью до double: 1,00117181829509