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

*Факультет социально-экономических и компьютерных наук*

Панфилов Даниил Сергеевич

**ЛАБОРАТОРНАЯ РАБОТА №2**

*Использование основных операторов языка С#*

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

# 1. Задание №1

Первое задание посвящено получению навыков работе с основными операторами языка С#. При решении задачи, использовать все типы циклов.

## 1.1.1. Общая постановка задачи

1. Для задачи 1 следует найти количество чётных чисел последовательности, оканчивающаяся на 0;
2. Составить систему тестов и вычислить сумму для последовательностей разной длины;
3. Вывести все результаты на экран;
4. Объяснить полученные результаты.

## 1.1.2 Частная постановка задачи

Рассчитать сумму чётных чисел из последовательности определённой длины. Для этого нам понадобиться её значение, задаваемое пользователем.

Задача: дана последовательность из n целых чисел. Найти сумму чётных элементов этой последовательности.

## 1.2 Анализ

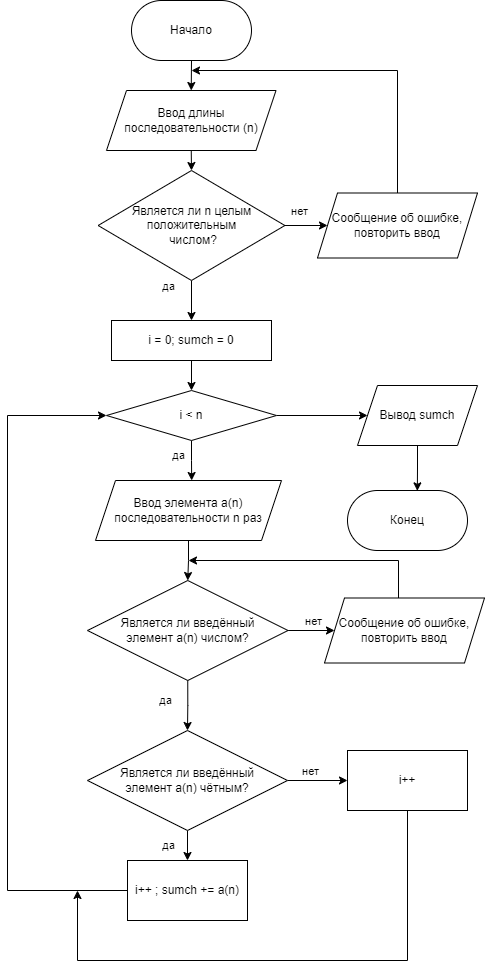
В данном задании пользователь вводит переменную n (длина последовательности) типа int. Анализ классов входных и выходных данных приведен в таблице 1. Выходное данное – тип int.

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 | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *n –*отрицательное целое число | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *n –* положительное целое число | 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 IsNCrorrect и IsNCrorrect отвечают за правильность введённого типа данных (для длины и элементов последовательности). Переменная sumch принимает значение суммы чётных чисел последовательности.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лаба\_2

{

internal class Program

{

static void Main(string[] args)

{

string input;

string n;

bool IsNCorrect;

bool IsNCorrect2;

int sumch;

Console.WriteLine("Введите длину последовательности: ");

sumch = 0;

do

{

n = Console.ReadLine();

IsNCorrect = int.TryParse(n, out int len);

if (!IsNCorrect)

{

Console.WriteLine("Введите, пожалуйста, положительное, целое число (длину)");

}

else

{

Console.WriteLine("Введите элементы последовательности");

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

{

do

{

Console.WriteLine($"Введите число {i + 1}: ");

input = Console.ReadLine();

IsNCorrect2 = int.TryParse(input, out int chislo);

if (!IsNCorrect2)

{

Console.WriteLine("Введите корректный элемент последовательности");

}

else

{

if (chislo %2 ==0)

{

sumch += chislo;

}

}

} while (!IsNCorrect2);

}

}

} while (!IsNCorrect);

Console.WriteLine(sumch);

}

}

}

## 1.5. Тестирование

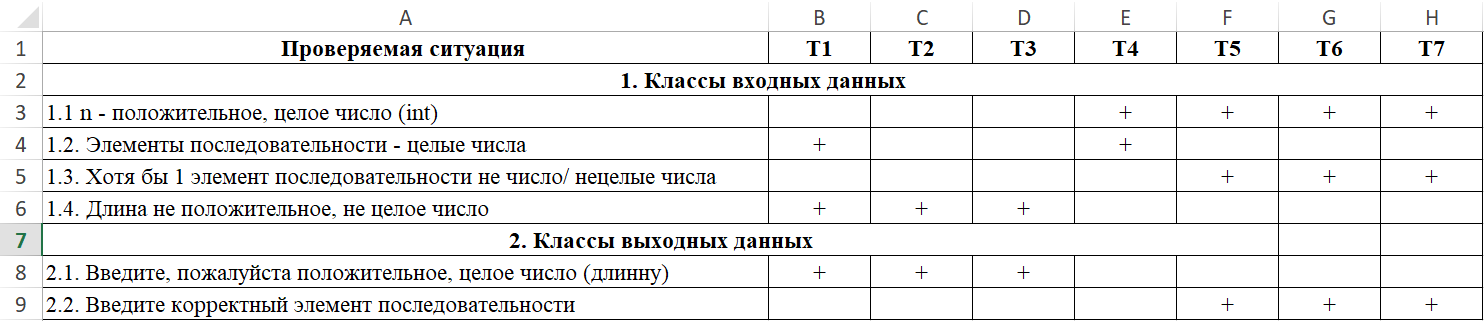
Результаты тестирования представлены на рисунке 1.5.1. По рисунку видно, что все ожидаемые результаты совпадают с реальными.



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

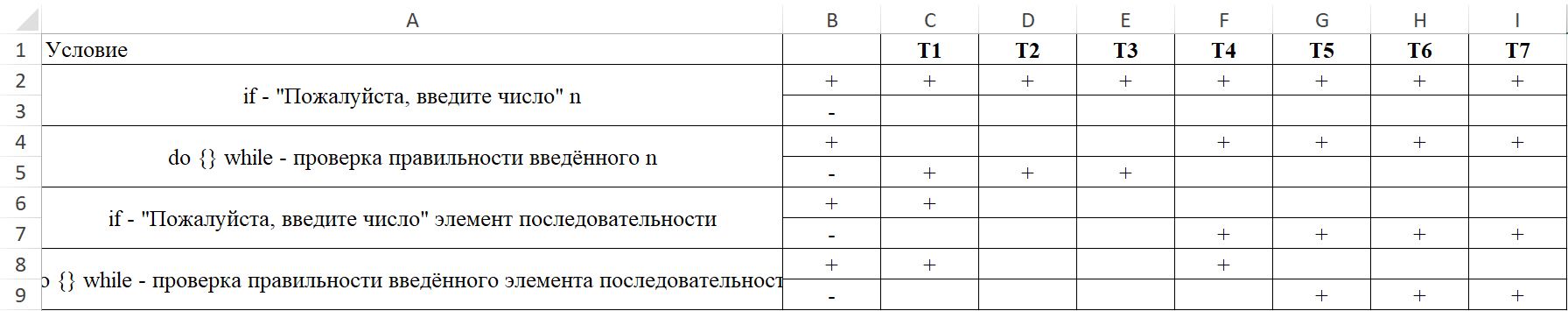
Таблица тестов составлялась с помощью дополнительных таблиц по критериям черного и белого ящиков. Они представлены ниже.

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. 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 получению навыков работы с циклами языка C# (while, do while). Также учит определять правильные тесты и анализировать типы входных данных.

## 2.1.1. Общая постановка задачи

1. Для задачи 2 следует посчитать количество чисел последовательности, которые кратны K1 и не кратны K2, и вывести на экран. Так как в задаче не сказано какие числа K1 и K1, пусть они будут целые;
2. Составить систему тестов и вычислить количество подходящих под условие чисел;
3. Результаты всех вычислений вывести на печать.
4. Объяснить полученные результаты.

## 2.1.2. Частная постановка задачи

Найти количество чисел в последовательности, которую задаёт пользователь (в конце 0), которые кратны K1 и не кратны K1. Числа K1 и K1 вводит пользователь.

Задача: Дана последовательность целых чисел, за которой следует 0. Найти количество элементов этой последовательности, кратных числу K1 и не кратных числу K2.

## 2.2. Анализ

В данном задании вводятся сначала 2 переменные типа int (K1 и K2). Если пользователь ввёл значения корректно, ему предстоит ввести элементы последовательности (в конце 0), в противном случае нужно будет повторить ввод переменных до тех пор, пока она не будет введена корректно (сообщение о правильности ввода выводится на экран).

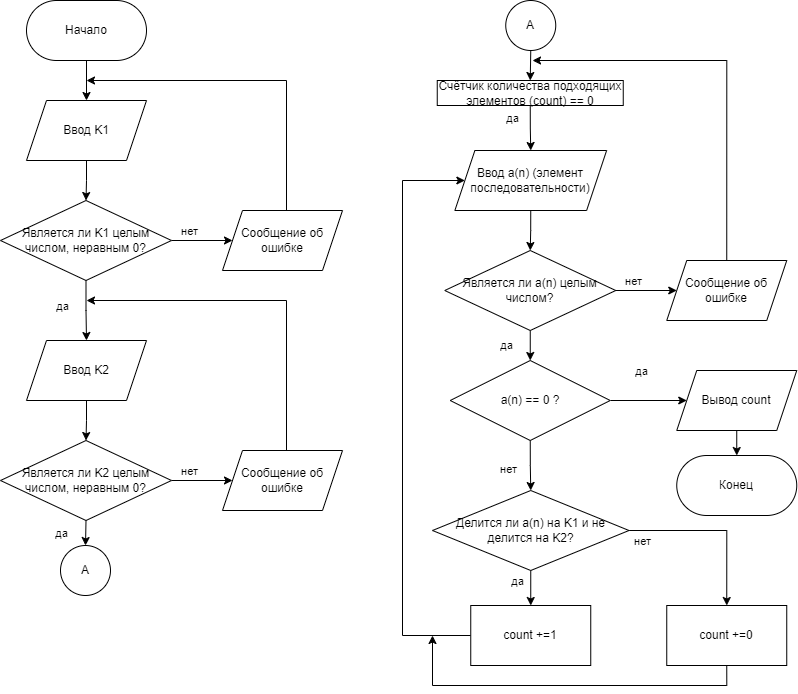
Стоит отметить, что K1, K2 и a(n)– целые числа, но K1 и K2 не равны 0 (a(n) – элементы последовательности).

Таблица 4 – Классы входных и выходных данных для переменных K1, K2, a(n)

| **Классы входных данных** | **Условия** | **Классы выходных данных** |
| --- | --- | --- |
| *K1* – строка или нецелое число или 0 | Некорректный ввод | Сообщение об ошибке из-за неправильного формата введенных данных |
| *K1* – целое число | Сохранение значения в переменную, далее проверка на делимость | Ожидаемый результат |
| *K2* – строка или нецелое число или 0 | Некорректный ввод | Сообщение об ошибке из-за неправильного формата введенных данных |
| *K2* – целое число | Сохранение значения в переменную, далее проверка на делимость | Ожидаемый результат |
| *a(n)* – строка или нецелое число | Некорректный ввод | Сообщение об ошибке из-за неправильного формата введенных данных |
| *a(n)* –целое число | Проверка не делимость | Ожидаемый результат |
| *a(n)* == 0 | Конец последовательности | Результат |

## 2.3 Проектирование

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



***Рисунок 1.1 – Схема алгоритма для задачи 2***

## 2.4 Разработка

Разработка осуществляется с помощью листинга программы. Ниже представлен код. Переменная IsK1Correct, IsK2Correct и IsInCorrect отвечает за правильность введенного типа данных (для ввода минимум трёх переменных).

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Задание\_2

{

internal class Program

{

static void Main(string[] args)

{

string input;

string kr1;

string kr2;

int count = 0;

int chislo;

bool IsK1Correct;

bool IsK2Correct;

bool IsInCorrect;

Console.WriteLine("Введите целое число K1");

do

{

kr1 = Console.ReadLine();

IsK1Correct = int.TryParse(kr1, out int k1);

if (!IsK1Correct | k1 == 0)

{

Console.WriteLine("Введено некорректное число, вы ввели не целое число, либо нечисло, либо 0");

}

else

{

Console.WriteLine("Введите целое число K2");

do

{

kr2 = Console.ReadLine();

IsK2Correct = int.TryParse(kr2, out int k2);

if (!IsK2Correct | k2 == 0)

{

Console.WriteLine("Введено некорректное число, вы ввели не целое число, либо нечисло, либо 0");

}

else

{

do

{

Console.WriteLine("Введите числа последовательности, в конце 0");

do

{

input = Console.ReadLine();

IsInCorrect = int.TryParse(input, out chislo);

if (!IsInCorrect)

{

Console.WriteLine("Введите корректный элемент последовательности");

}

else

{

if (chislo % k1 == 0 && chislo % k2 != 0)

{

count++;

}

}

} while (input != "0");

} while (!IsInCorrect);

}

} while (!IsK2Correct);

}

} while (!IsK1Correct);

Console.WriteLine($"Количество чисел, кратных К1 и не кратных К2: {count}");

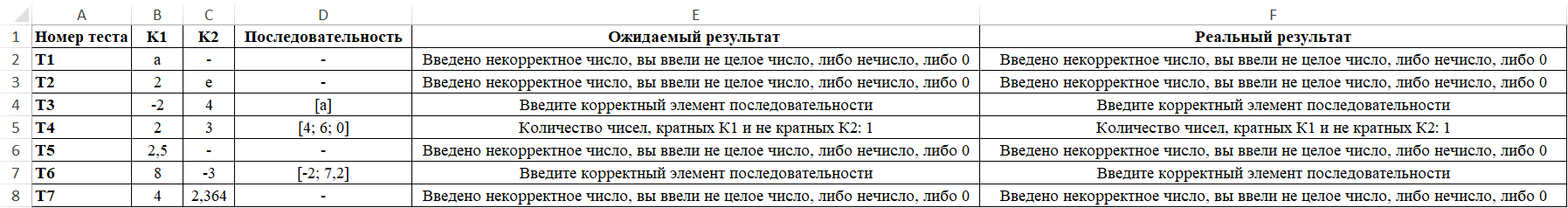
}

}

}

## 2.5. Тестирование

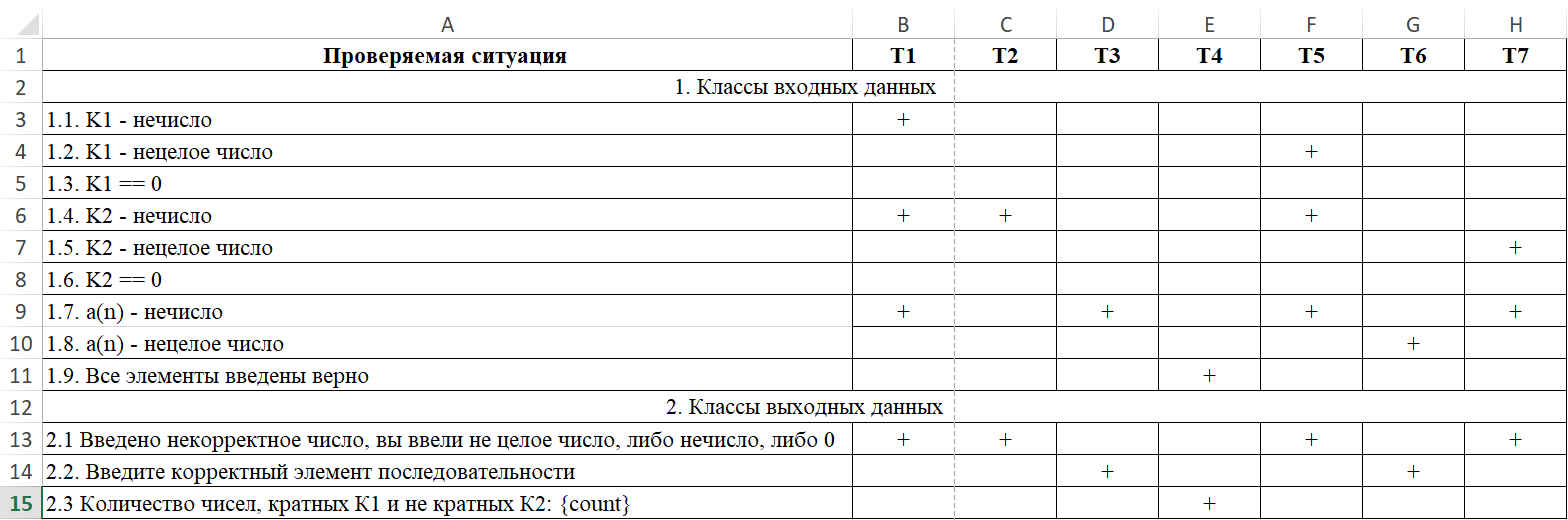
Результаты тестирования представлены на рисунке 2.5.1. По рисунку видно, что все ожидаемые результаты совпадают с реальными.



***Рисунок 2.5.1 – Тесты задачи 2***

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

На рисунке 2.5.2. представлено тестирование по критериям черного ящика.



***Рисунок 2.5.2 –Чёрный ящик 2***

Ниже на рисунке 2.5.3. приведены критерии белого ящика.



***Рисунок 2.5.3 –Белый ящик 2***

# 3. Задание №3

Третье задание посвящено умению работать с арифметической прогрессией и понимать алгоритм нахождения суммы её элементов.

## 3.1.1. Общая постановка задачи

1. Для задачи 3 вычислить сумму арифметической прогрессии, имея ровно n элементов. Количество элементов задаёт сам пользователь. Разность прогрессии = 2;
2. Сумму вывести на экран;
3. Объяснить полученный результат.

## 3.1.2. Частная постановка задачи

S=1+3+5+7+ …, всего n слагаемых. Число n задаётся пользователем.

## 3.2. Анализ

В данном задании вводится 1 переменная типа int – количество слагаемых в прогрессии. Также вначале были объявлены 3 переменные: a1 = 1 (первый элемент последовательности, он дан в задании), a(n) = 1 (для его нахождения понадобиться знать длину последовательности - n) и S = 0 (сумма арифметической прогрессии).

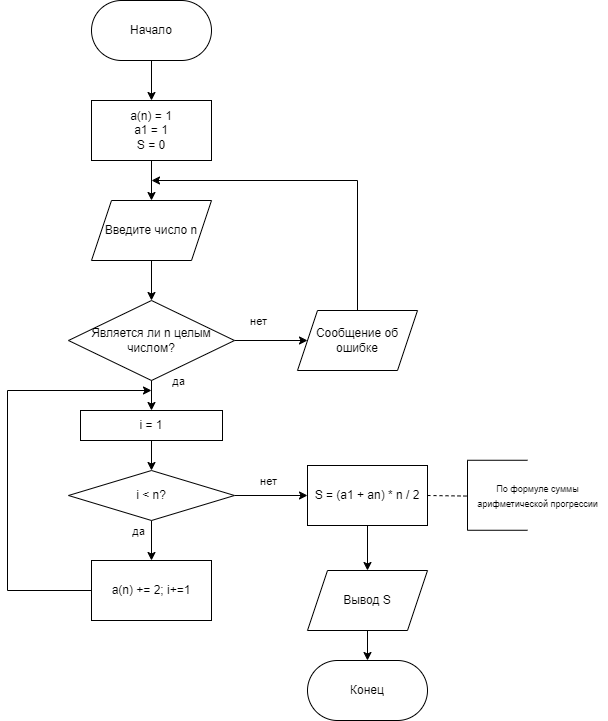
Анализ входных и выходных данных приведёт в таблице 5.

Таблица 5 – 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 | Невозможно вычислить | Сообщение об ошибке из-за неправильного формата введенных данных |
| *n –*любое целое число | Программа выполняется | Стандартные выходные данные |

## 3.3. Проектирование

Блок схема для данной задачи представлена на рисунке 3.2.



***Рисунок 3.2. – Схема алгоритма для задачи 3***

## 3.4 Разработка

Разработка осуществляется с помощью листинга программы.

Ниже представлен код. Переменная IsNCorrect отвечает за правильность введенного типа переменной (длины прогрессии).

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)

{

string num;

bool IsNCorrect;

int a1 = 1;

int an = 1;

int S = 0;

Console.WriteLine("Введите сило N");

do

{

num = Console.ReadLine();

IsNCorrect = int.TryParse(num, out int n);

if (!IsNCorrect)

{

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

}

else

{

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

{

an += 2;

}

S = (a1 + an) \* n / 2;

}

} while (!IsNCorrect);

Console.WriteLine($"Число S = {S}");

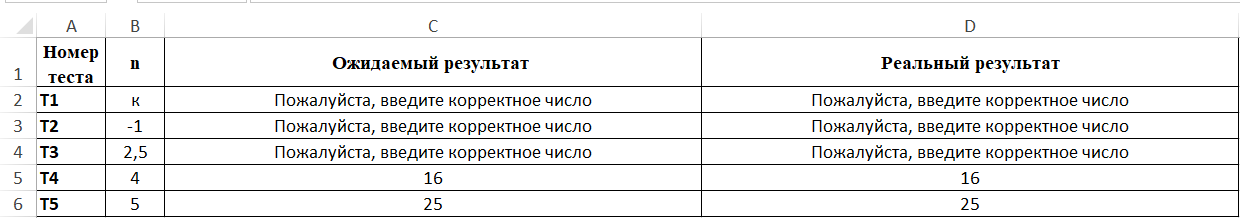
}

}

}

## 3.5. Тестирование

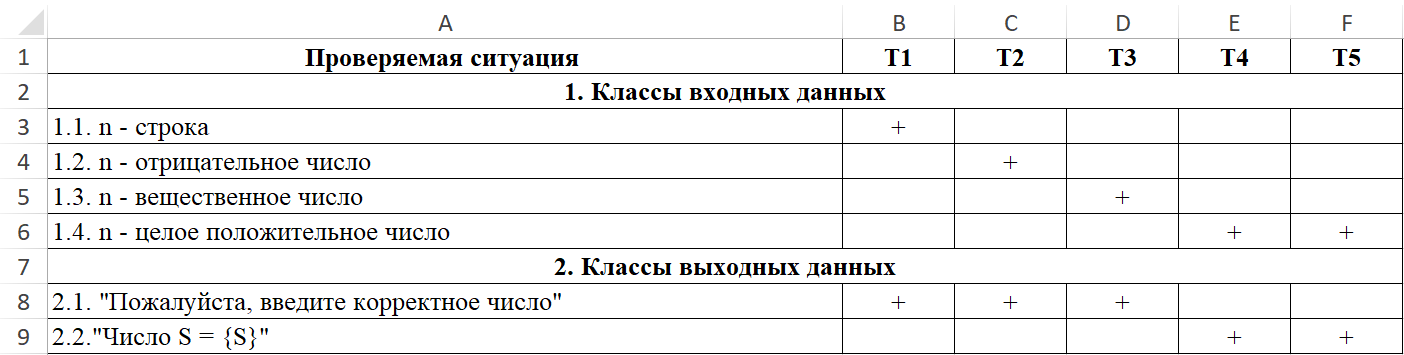
Результаты тестирования представлены на рисунке 3.5.1. По рисунку видно, что все ожидаемые результаты совпадают с реальными.



***Рисунок 3.5.1. – Тесты задачи 3***

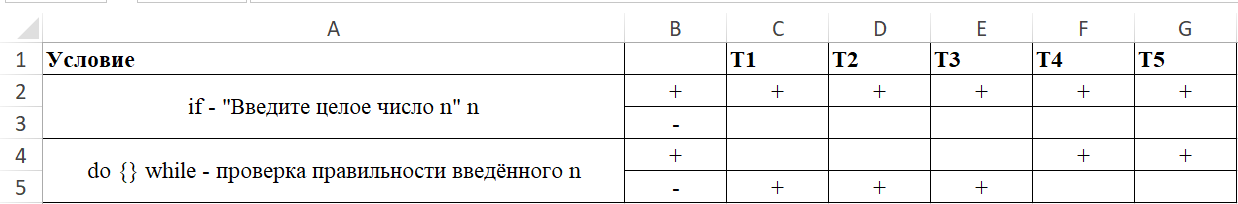
Ниже приведены критерии черного ящика, рассмотренные по классам входных и выходных данный.

На рисунке 3.5.2. представлено тестирование по критериям черного ящика.



***Рисунок 3.5.2. – Чёрный ящик 3***

Также тестирование проводилось по критериям белого ящика. Таблица критериев белого ящика представлена на рисунке 3.5.3. (см. ниже).



***Рисунок 3.5.3 –Белый ящик 3***