|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально-экономических и компьютерных наук* |
|  |
| Ханжин Александр Евгеньевич  **ЛАБОРАТОРНАЯ РАБОТА №1**  студента образовательной программы «Разработка информационных систем для бизнеса» по направлению подготовки *06.10.23 Программная инженерия*   |  |  | | --- | --- | |  | Руководитель к.т.н., доцент,  кафедры ИТБ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  О. Л. Викентьева |   Пермь, 2023 |

# Задача 1

## Постановка задачи

Найти значения выражений:

1. m/--n++;
2. m/n<n—;
3. m+n++>n+m;
4. ;

## Анализ

Исходные данные:

* m, n, x – вещественные числа.

Выходные данные:

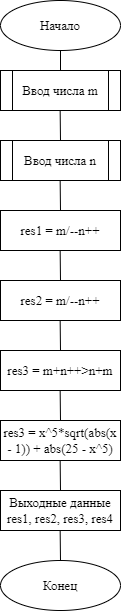
* res1, res4 – вещественные числа,
* res2, res3 – логические переменные.

***Табл. 1. Анализ классов исходных данных.***

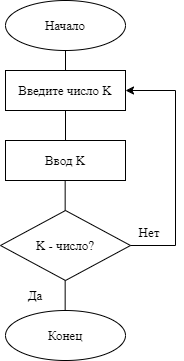
|  |  |  |  |
| --- | --- | --- | --- |
| **Задача** | **Исходные данные** | **Класс входных данных** | **Выходные данные** |
| 1) m/--n++ | M, n | Вещественные числа | Вещественное число |
| Строки (не числа) | Сообщение об ошибке ввода |
| 2) m/n<n—  3) m+n++>n+m | M, n | Вещественные числа | Логическое значение |
| Строки (не числа) | Сообщение об ошибке ввода |
| 4) | x | Вещественное число <>0 | Вещественное число |
| Строка (не число) | Сообщение об ошибке ввода |

## Алгоритм (блок-схема)

Для дальнейшей реализации проектирования необходимо составить блок схему, она представлена ниже на рисунках 1.3.1 и 1.3.2. Она состоит из двух частей: основная часть программы (рис. 1.3.1) и функции ввода числа (рис. 1.3.2)

******

***Рис. 1.3.1. Блок-схема 1 задания***

******

***Рис. 1.3.2. Блок-схема операции “Ввод числа”***

## Программа (листинг)

static void FirstTask()

{

double m, n;

bool isConvert;

Console.WriteLine("Задание 1. Вычисление значений");

// считывание чисел m и n

do

{

Console.WriteLine("Привет! Введи число m");

isConvert = double.TryParse(Console.ReadLine(), out m);

if (!isConvert)

Console.WriteLine($"Ошибка, вы ввели не число m. Повторите ввод");

} while (!isConvert);

do

{

Console.WriteLine("Привет! Введи число n");

isConvert = double.TryParse(Console.ReadLine(), out n);

if (!isConvert)

Console.WriteLine("Ошибка, вы ввели не число n. Повторите ввод");

} while (!isConvert);

// проверка возможности деления на n и вывод результатов первого выражения

if (n != 1)

{

n -= 1;

double res1 = m / n++;

Console.WriteLine($"Значение выражения: (m / --n++) равно {res1}. Значение m={m}, n={n}");

}

else { Console.WriteLine("Невозможно вычислить (m / --n++), деление на 0 невозможно"); };

// проверка возможности деления на n и вывод результатов второго выражения

if (n != 0)

{

bool res2 = m / n < n--;

Console.WriteLine($"Значение выражения: (m / n < n--) равно {res2}. Значение m={m}, n={n}");

}

else { Console.WriteLine("Невозможно вычислить (m / n < n--), деление на 0 невозможно"); }

// вычисление и вывод значения третьего выражения

bool res3 = m + n++ > n + m;

Console.WriteLine($"Значение выражения: (m + n++ > n + m) равно {res3}. Значение m={m}, n={n}");

double x;

// считывание переменной x

do

{

Console.WriteLine("Привет! Введи число x");

isConvert = double.TryParse(Console.ReadLine(), out x);

if (!isConvert)

Console.WriteLine("Ошибка, вы ввели не число x. Повторите ввод");

} while (!isConvert);

// вычисление и вывод значения четвертого выражения

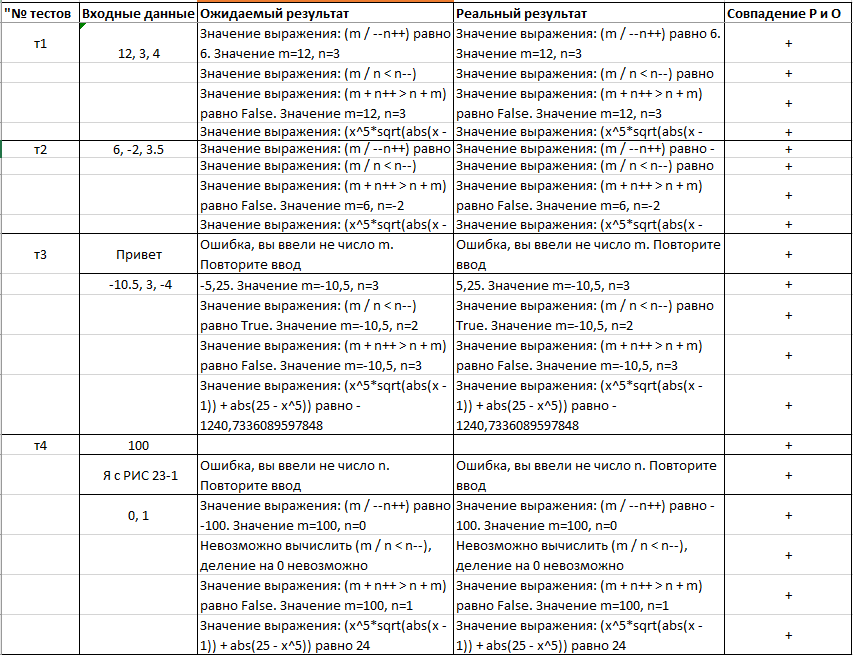
double res4 = Math.Pow(x, 5) \* Math.Sqrt(Math.Abs(x - 1)) + Math.Abs(25 - Math.Pow(x, 5));

Console.WriteLine($"Значение выражения: (x^5\*sqrt(abs(x - 1)) + abs(25 - x^5)) равно {res4}");

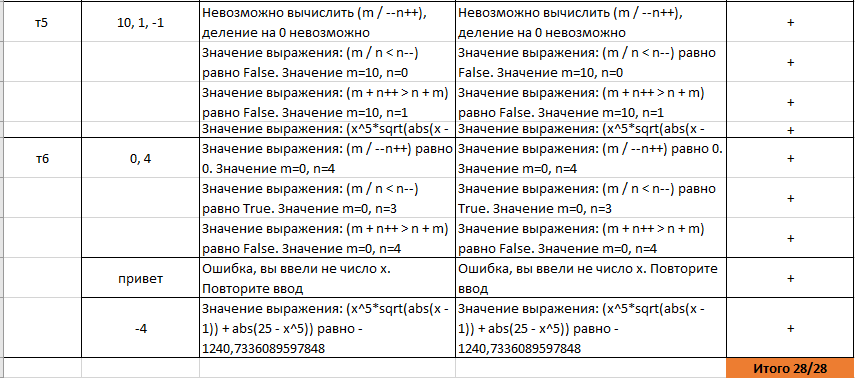
}

## Тестирование

Результаты тестирования представлены на рисунках 1.5.1 и 1.5.2 На рисунках видно, что все ожидаемые результаты совпадают с реальными. В ходе этого тестирования программа была протестирована с использованием различных входных данных. Тесты составлялись таким образом, чтобы покрыть все возможные варианты ввода данных пользователем.



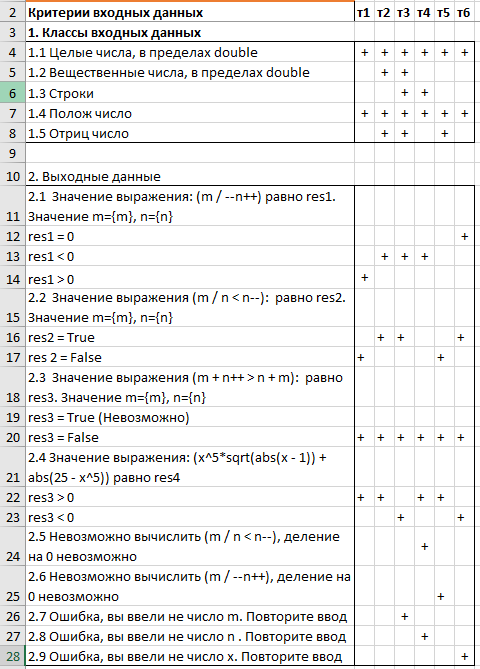
***Рисунок 1.5.1. – Таблица тестов для задания 1***

******

***Рисунок 1.5.2. – Таблица тестов для задания 1***

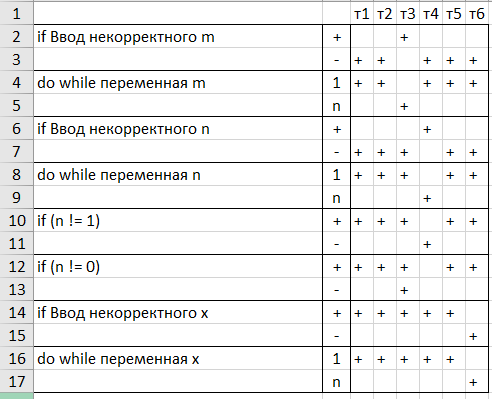
Таблица тестов составлялась с помощью дополнительных таблиц по критериям черного и белого ящиков. Они представлены ниже.

Рисунок 1.5.3. содержит таблицу с критериями черного ящика. Она состоит из классов входных и выходных данных, рассмотренных в анализе.



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

Также тестирование проводилось по критериям белого ящика. Так как циклы, отвечающие за проверку правильного ввода, не дадут это сделать. Таблица критериев белого ящика представлена на рисунке 1.5.4 (см. ниже).



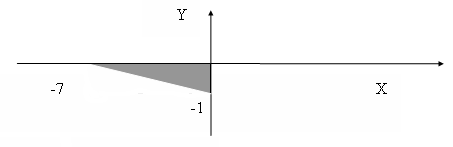
***Рисунок 1.5.4. – Таблица критериев белого ящика задания 1***

Ознакомиться с полным списком можно в формате excel, который приложен к лабораторной работе.

# Задача 2

## Постановка задачи

Определить принадлежность точки к заданной области.



***Рис.2.1.1. Постановка задачи 2***

## Анализ

Исходные данные:

* x, y – вещественные числа

Выходные данные:

* res – логическая переменная

***Табл. 1. Анализ классов исходных данных.***

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Классы** | **Ожидаемый результат** |
| x, y | Строка  Символ | Ошибка ввода, повторный ввод |
| Число | Логическое значение |

Классы входных данных:

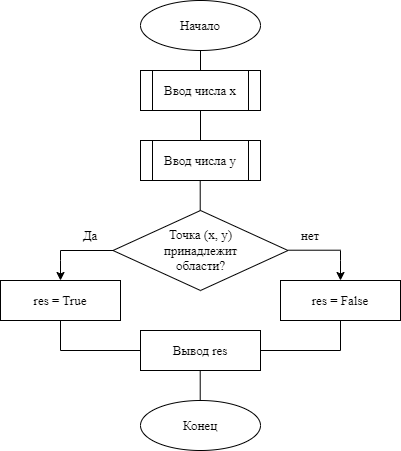
1. X и Y — текст (не преобразуемый в число).
2. X и Y — числа.

Классы выходных данных:

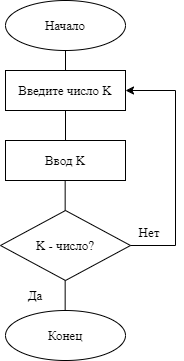
1. Ошибка ввода.
2. Ответ: True
3. Ответ: False.

## Блок-схема

Для дальнейшей реализации проектирования необходимо составить блок схему. Она состоит из двух частей: основная часть программы (рис. 2.3.1) и функции ввода числа (рис. 2.3.2).



***Рис. 2.3.1. Блок-схема 2 задания.***



***Рис. 2.3.2. Блок-схема операции “Ввод числа”***

## Программирование

static bool SecondTask()

{

double x, y;

bool isConvert;

Console.WriteLine("Задание 2. Определение положения точки");

// считывание координат (x y)

do

{

Console.WriteLine("Привет! Введи координату x");

isConvert = double.TryParse(Console.ReadLine(), out x);

if (!isConvert)

Console.WriteLine("Введи координату x, вещественное число");

} while (!isConvert);

do

{

Console.WriteLine("Привет! Введи координату y");

isConvert = double.TryParse(Console.ReadLine(), out y);

if (!isConvert)

Console.WriteLine("Введи координату y, вещественное число");

} while (!isConvert);

// вычисление положения точки заданной области

bool res;

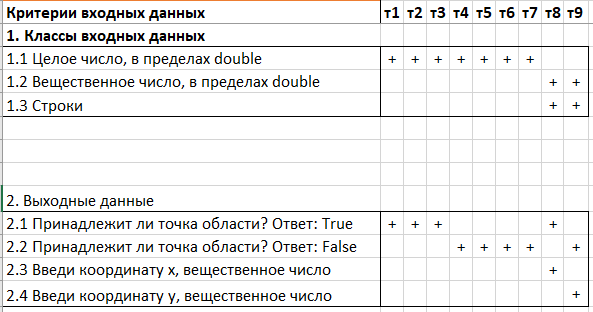
res = ((y >= -1 / 7.0 \* x - 1) & (x <= 0) & (y <= 0));

return res;

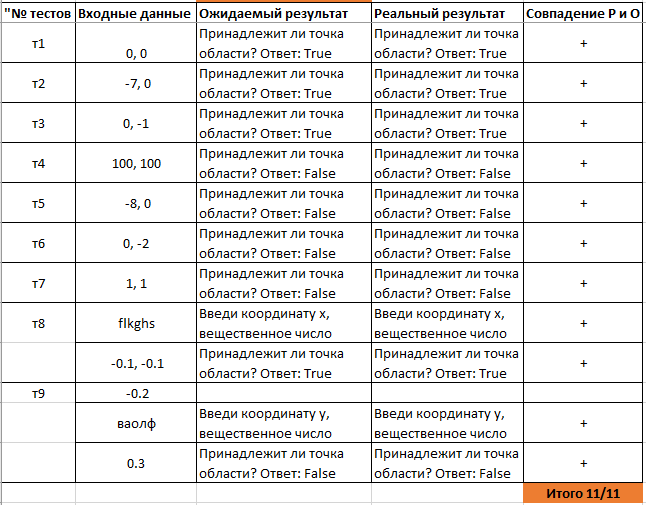
}

## Тестирование

Чтобы проверить все возможные варианты, нужно рассмотреть точки, лежащие в каждой из предложенных областей, также проверить точки на границах области. Для этого была составлена таблица критериев Черного ящика. Она представлена на рисунке 2.5.1.

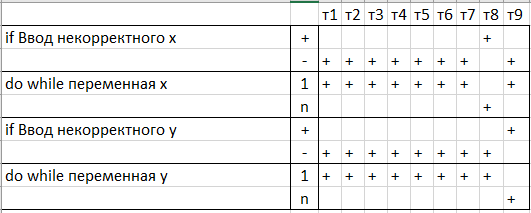
*** Рис. 2.5.1 Таблица критериев черного ящика задания 2.***

К ней были составлены тесты, представление на рисунке 2.5.2. Ожидаемые и реальные результаты совпали на 100%.



***Рисунок 2.5.2. – Таблица тестов для задания 2***

Таблица с тестированием по критериям белого ящика представлена на рисунке 2.5.3.

*** Рисунок 2.5.3. – Таблица критериев белого ящика задания 2***

Ознакомиться с полным списком можно в формате excel, который приложен к лабораторной работе.

# Задача №3

## Постановка задачи

Для задачи 3 вычислить значение выражения, используя различные вещественные типы данных (float и double):

Исходные данные:

* a = 1000,
* b = 0.0001.

## Анализ

Классы входных данных:

* aDouble, bDoube – константы double,
* aFloat, bFloat – константы float.

Классы выходных данных:

* resDouble – тип double,
* resfloat – тип float.

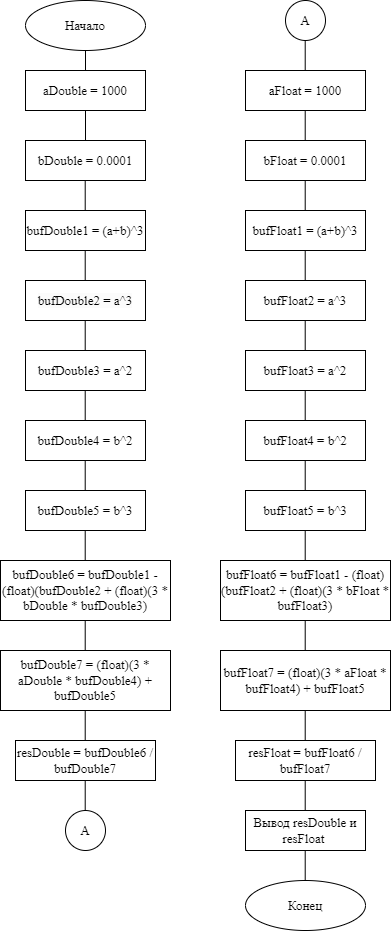
***Табл. 1. Анализ классов исходных данных.***

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Тип исходных данных** | **Ожидаемый результат** |
| a=1000 b=0.0001 | a, b – вещественные числа | Вещественное число |

## Блок-схема

Для дальнейшей реализации проектирования необходимо составить блок схему, она представлена ниже на рисунке 1. Вспомогательные переменные для bufDouble1-7, для типа double и bufFloat1-7 для типа float нужны, чтобы понять в каком месте значения в типе float отличаются от double.

Блок схема для данной задачи на рисунке 3.3.1.



***Рис. 3.3.1. Блок-схема 3 задания***

## Программа

static void ThirdTask()

{

Console.WriteLine("Задание 3. Вычисление примера с заданными данными");

double aDouble;

double bDouble;

aDouble = 1000;

bDouble = 0.0001;

// вычисление выражения в типе double

double bufDouble1 = Math.Pow(aDouble + bDouble, 3);

double bufDouble2 = Math.Pow(aDouble, 3);

double bufDouble3 = Math.Pow(aDouble, 2);

double bufDouble4 = Math.Pow(bDouble, 2);

double bufDouble5 = Math.Pow(bDouble, 3);

double bufDouble6 = (bufDouble1 - (double)(bufDouble2 + (double)(3 \* bufDouble3 \* bDouble)));

double bufDouble7 = ((double)(3 \* aDouble \* bufDouble4) + bufDouble5);

double resDouble = bufDouble6 / bufDouble7;

// вычисление выражения в типе float

float aFloat = 1000;

float bFloat = 0.0001f;

float bufFloat1 = (float)Math.Pow(aFloat + bFloat, 3);

float bufFloat2 = (float)Math.Pow(aFloat, 3);

float bufFloat3 = (float)Math.Pow(aFloat, 2);

float bufFloat4 = (float)Math.Pow(bFloat, 2);

float bufFloat5 = (float)Math.Pow(bFloat, 3);

float bufFloat6 = (float)(bufFloat1 - (float)(bufFloat2 + (float)(3 \* bFloat \* bufFloat3)));

float bufFloat7 = (float)((float)(3 \* aFloat \* bufFloat4) + bufFloat5);

float resFloat = bufFloat6 / bufFloat7;

Console.Write($"A - double, B - double. Ответ: ");

Console.WriteLine(resDouble);

Console.Write($"A - float, B - float. Ответ: ");

Console.WriteLine(resFloat);

}

## Объяснение выходных значений

A - double, B - double. Ответ: 0,9973843559953912

A - float, B - float. Ответ: 2133333,5

Результат double был получен из-за «накопления ошибок» неточного вычисления дробных значений, поэтому вместо нужной 1, был получен близкий к единице результат.

Результат float получился столь неточный, так как мы вычисляли значение степеней чисел с большой глубиной и значение было просто обрезано, из-за этого получился совершенно некорректных результат.

Хочу добавить, что с вычисление дробных чисел с большой глубиной является проблемой и в других языках программирования. В c# решить эту проблему можно попробовать с использованием математических переменных decimal и bigdecimal. Их использование поможет получить более точный результат.