|  |
| --- |
| 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 №2**  студента образовательной программы «Разработка информационных систем для бизнеса» по направлению подготовки *09.03.04 Программная инженерия*   |  |  | | --- | --- | |  | Руководитель к.т.н., доцент кафедры ИТБ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  О. Л. Викентьева |   Пермь, 2023 |

# Задача 1

## Постановка задачи

Дана последовательность из n целых чисел. Найти максимальный элемент в этой последовательности.

Для полноты понимания результата работы программы так же реализуем ввод количества максимумов.

## Анализ

Исходные данные:

‒ n – целое положительное число длины последовательности;

‒ number – целые числа;

‒ maxNumber, countMax – целые числа числа;

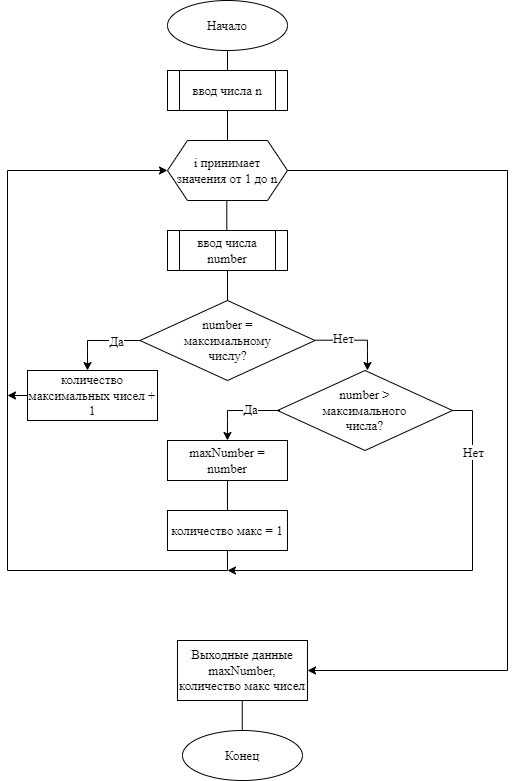
Входные и выходные данные каждой функции представлены в таблице 1.

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

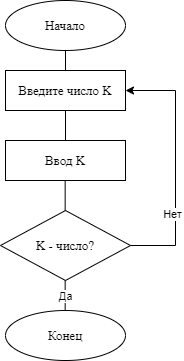
|  |  |  |  |
| --- | --- | --- | --- |
| **Описание функции** | **Исходные данные** | **Класс входных данных** | **Выходные данные** |
| 1) Поиск макс. элемента последовательности | n, numbers | Вещественные числа | Вещественное число |
| Строки (не числа) | Сообщение об ошибке ввода |
| 2) Подсчет количества максимальных элем-ов | n, numbers | Вещественные числа | Вещественное число |

## Алгоритм (блок-схема)

Для дальнейшей реализации проектирования необходимо составить блок схему. 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()

{

int n;

bool isConvert;

Console.WriteLine("Задание 1. Дана последовательность из n целых чисел. Найти максимальный элемент в этой последовательности.");

Console.WriteLine("Привет! Введи длину последовательности.");

// считывание числа натурального числа n

do

{

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

if (!isConvert)

{

Console.WriteLine("Длина последовательности должна быть целым числом.");

Console.WriteLine("Введи длину последовательности.");

}

else if (n < 0)

{

Console.WriteLine("Длина последовательности должна быть положительной.");

Console.WriteLine("Введи длину последовательности.");

isConvert = false;

}

else if (n == 0)

{

Console.WriteLine("Длина последовательности не может быть равна нулю.");

Console.WriteLine("Введи длину последовательности.");

isConvert = false;

};

} while (!isConvert);

// переменные для вычисления максимального числа из последовательности и их кол-ва

int number, maxNumber, countMax;

maxNumber = -1000000000;

countMax = 1;

// последовательное считывание числа number

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

{

Console.WriteLine($"Введи целое число. Уже введено {i} чисел из {n}.");

do

{

isConvert = int.TryParse(Console.ReadLine(), out number);

if (!isConvert)

Console.WriteLine("Ошибка. Нужно ввести целое число.");

} while (!isConvert);

// перезапись максимального числа из предыдущих введеных и текущего и количества введеных макс. чисел

// по заданию нужно вывести только макс. из последовательности, но я доработал программу и подсчитываю еще их количество

if (maxNumber == number)

{

countMax += 1;

}

else if (number > maxNumber)

{

maxNumber = number;

countMax = 1;

}

}

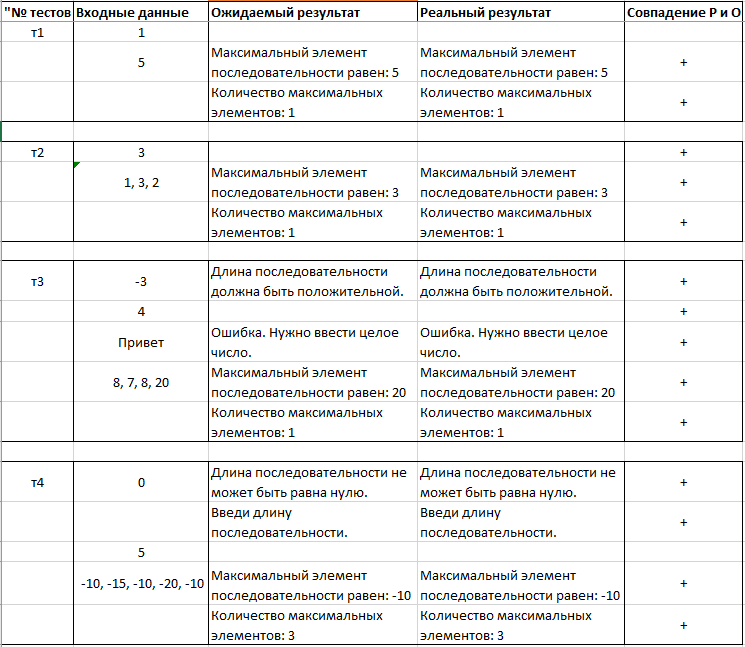
Console.WriteLine($"Максимальный элемент последовательности равен: {maxNumber}");

Console.WriteLine($"Количество максимальных элементов: {countMax}");

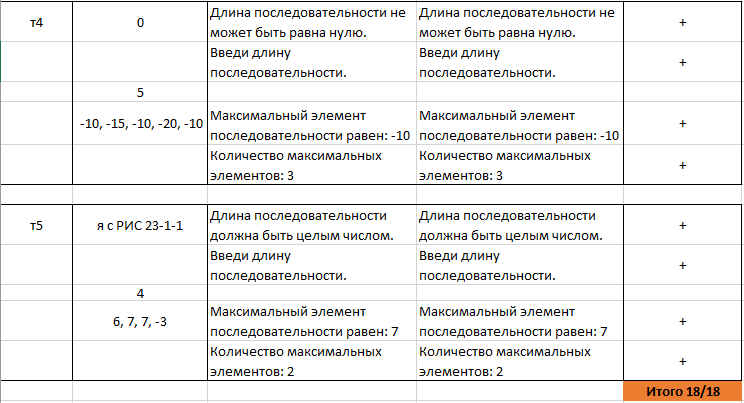
}

## 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.



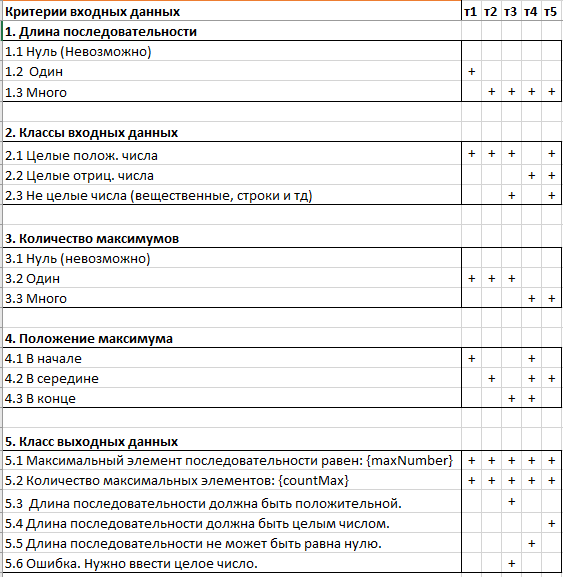
***Рисунок 1.5.1. – Таблица тестов для задания 1***

******

***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.

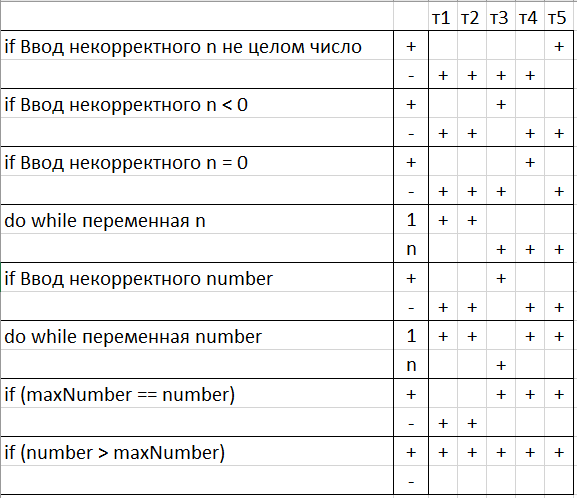
Рисунок 1.5.3. содержит таблицу с критериями черного ящика. Она состоит из классов входных и выходных данных, рассмотренных в анализе, а также других критериев.



***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).

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



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

# Задача 2

## Постановка задачи

Дана последовательность целых чисел, за которой следует 0. Найти максимальный элемент в этой последовательности.

Для полноты понимания результата работы программы так же реализуем ввод количества максимумов.

## Анализ

Исходные данные:

‒ number – целое положительное число длины последовательности;

Выходные данные:

‒ maxNumber, countMax – целые числа;

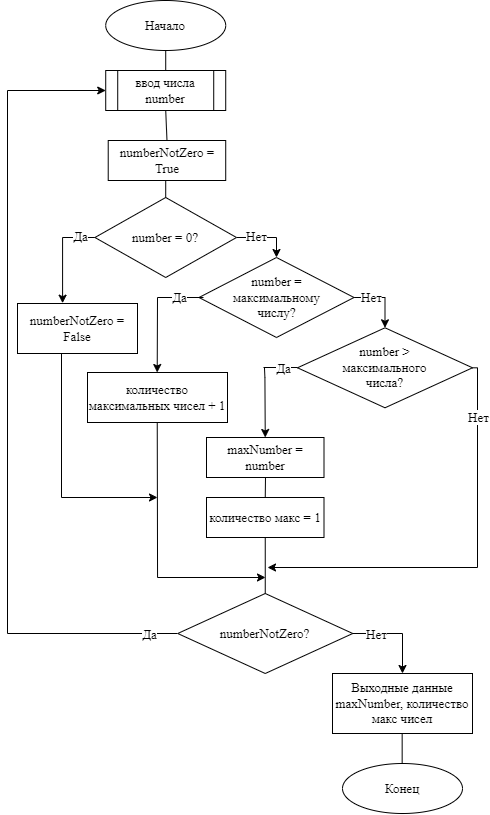
Входные и выходные данные каждой функции представлены в таблице 1.

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

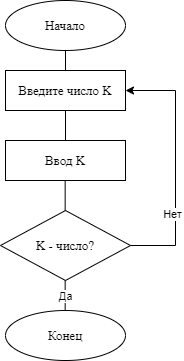
|  |  |  |  |
| --- | --- | --- | --- |
| **Описание функции** | **Исходные данные** | **Класс входных данных** | **Выходные данные** |
| 1) Поиск макс. элемента последовательности | n, numbers | Вещественные числа | Вещественное число |
| Строки (не числа) | Сообщение об ошибке ввода |
| 2) Подсчет количества максимальных элем-ов | n, numbers | Вещественные числа | Вещественное число |

## 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 void SecondTask()

{

Console.WriteLine("Задача 2. Дана последовательность целых чисел, за которой следует 0. Найти максимальный элемент в этой последовательности.");

// переменные для вычисления максимального числа из последовательности и их кол-ва

bool isConvert, numberNotZero;

int number, maxNumber, countMax, index;

maxNumber = -1000000000;

countMax = 1;

index = 0;

do

{

Console.WriteLine("Введите число. Если хотите завершить работу - введите 0.");

// последовательное считывание числа number

do

{

isConvert = int.TryParse(Console.ReadLine(), out number);

if (!isConvert)

Console.WriteLine("Ошибка. Нужно ввести целое число.");

} while (!isConvert);

if (number != 0)

{

numberNotZero = true;

// Количество введенных чисел

index += 1;

// перезапись максимального числа из предыдущих введеных и текущего и количества введеных макс. чисел

// по заданию нужно вывести только макс. из последовательности, но я доработал программу и подсчитываю еще их количество

if (maxNumber == number)

{

countMax += 1;

}

else if (number > maxNumber)

{

maxNumber = number;

countMax = 1;

}

}

else

{

numberNotZero = false;

}

} while (numberNotZero);

// Проверка длины последовательности

if (index == 0)

{

Console.WriteLine("Последовательность пуста.");

}

else

{

Console.WriteLine($"Максимальный элемент последовательности равен: {maxNumber}");

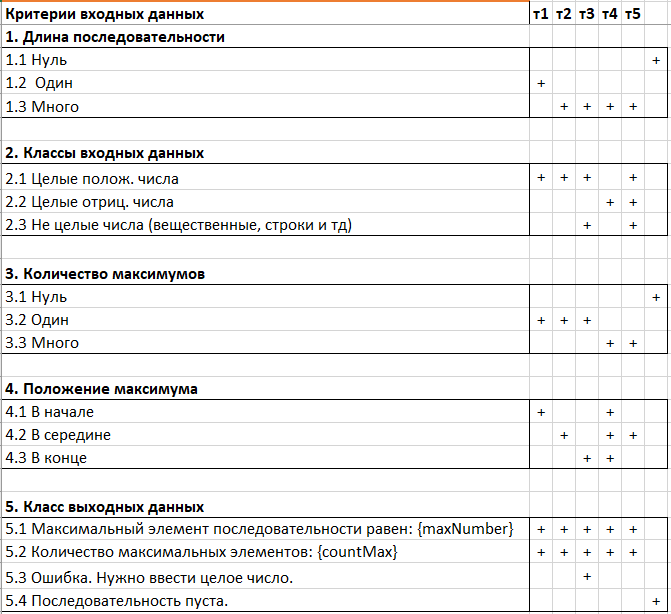
Console.WriteLine($"Количество максимальных элементов: {countMax}");

}

}

## Тестирование

Для тестирования программы была составлена таблица критериев Черного ящика. Она представлена на рисунке 2.5.1.

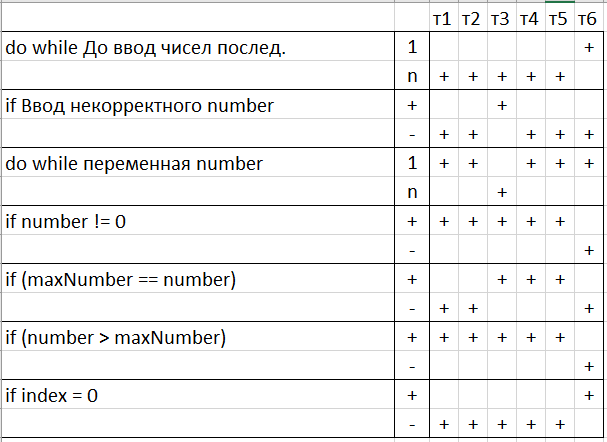
*** Рис. 2.5.1. Таблица критериев черного ящика задания 2.***

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 5. – Таблица критериев белого ящика задания 2***

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

# Задача 3

## Постановка задачи

Найти первое отрицательное число последовательности u=cos(ctg(n)), где n=1,2,3….;

Исходные данные:

‒ n – целое число 1,2,3…;

## Анализ

Классы выходных данных:

‒ n – целое число;

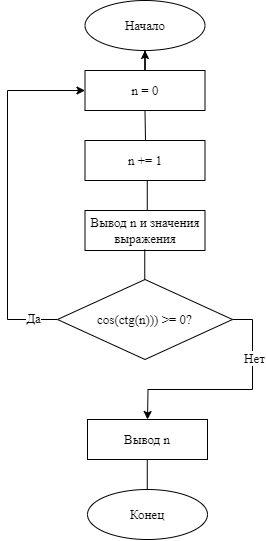
Входные и выходные данные каждой функции представлены в таблице 1.

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

|  |  |  |
| --- | --- | --- |
| **Описание функции** | **Исходные данные** | **Выходные данные** |
| 1) Поиск первого отриц. числа последовательности | n – целые натуральные числа | Вещественное число |

## Блок-схема

Для дальнейшей реализации проектирования необходимо составить блок схему (рисунок 3.3.1). По ходу выполнения программы пользователь получает информацию о каждом пройденном значении n и текущем результате выражения для отслеживания корректности итогового результата.

****

***Рис. 3.3.1. Блок-схема 1 задания***

## Программа

static void ThirdTask()

{

Console.WriteLine("Задание 3. Найти первое отрицательное число последовательности u=cos(ctg(n)), где n=1,2,3...");

int n = 0;

// перебор значений n и проверка условия

do

{

n += 1;

Console.Write($"При n = {n} принимает значение: ");

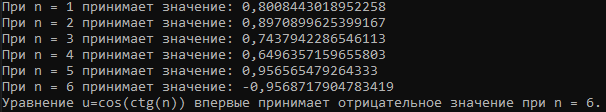
Console.WriteLine(Math.Cos((float)1 / Math.Tan(n)));

} while (Math.Cos((float)1 / Math.Tan(n)) >= 0);

Console.WriteLine($"Уравнение u=cos(ctg(n)) впервые принимает отрицательное значение при n = {n}.");

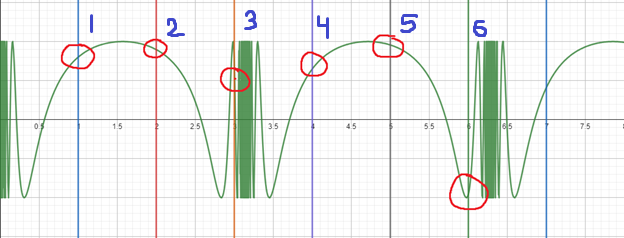
}

## Объяснение выходных значений

В результате работы программы получились следующие значения (рис3.5.1).

***Рис. 3.5.1. Блок-схема 1 задания***

Подтвердить корректность вычислений программы можно с помощью анализа функции. На рисунке 3.5.2 представлены пересечения графика функции с прямыми n = 1, 2, 3... Впервые такая прямая пересекает функцию ниже прямой y = 0 и принимает отрицательное значение при n = 6.

***Рис. 3.5.2. Блок-схема 1 задания***