ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ЛАБОРАТОРНОЙ РАБОТЕ №3

По дисциплине «Языки программирования»

ВАРИАНТ 10

Выполнил: ст. гр. ТКИ-142

Теканов Фёдор Андреевич

Проверил: к.т.н., доц. Васильева М. А.

(Проверил: к.т.н, доц. Балакина Е. П.)

Москва 2024

**Оглавление**

[1 ЗАДАНИЕ 3.1 4](#_Toc185391098)

[1.1 Формулировка задания 4](#_Toc185391099)

[1.2 Блок-схема основного алгоритма 5](#_Toc185391100)

[1.3 Код программы на языке С 7](#_Toc185391101)

[1.4 Результат выполнения программы 9](#_Toc185391102)

[1.5 Выполнение тестовых примеров 10](#_Toc185391103)

[1.6 Отметка о выполнение 11](#_Toc185391104)

[2 ЗАДАНИЕ 3.2 12](#_Toc185391105)

[2.1 Формулировка задания 12](#_Toc185391106)

[2.2 Блок-схема основного алгоритма 13](#_Toc185391107)

[2.3 Текст программы на языке С 18](#_Toc185391108)

[2.4 Результат выполнения программы 21](#_Toc185391109)

[2.5 Выполнение тестовых примеров 22](#_Toc185391110)

[2.6 Отметка о выполнение задания 23](#_Toc185391111)

1 задание 3.1

* 1. Формулировка задания

Протабулировать заданную в таблице функцию (Таблица 1). Использовать данные в таблице значения шага и интервала в качестве ввода пользователя для решения тестового примера. При невозможности расчёта функции в конкретной точке выводить её значение и надпись, означающую отсутствие решения.

Таблица 1 – Исходные данные

|  |  |  |
| --- | --- | --- |
| **Вариант** | **Функция** | **Константы** |
| **10** |  |  |

* 1. Блок-схема основного алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций расчета (Рисунок 2, Рисунок 3).

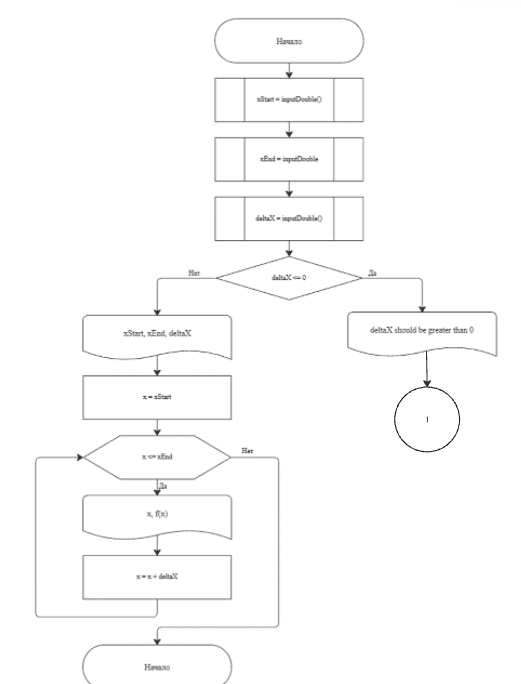


Рисунок 1 – Блок-схема основного алгоритма

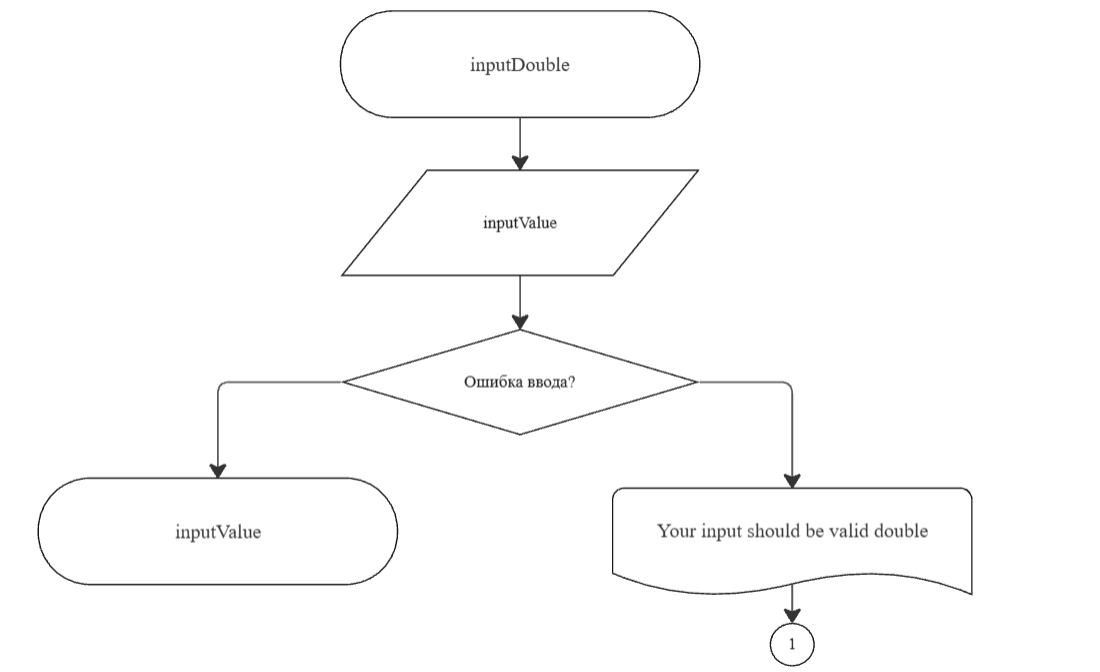


Рисунок 2 – Блок-схема функции inputDouble

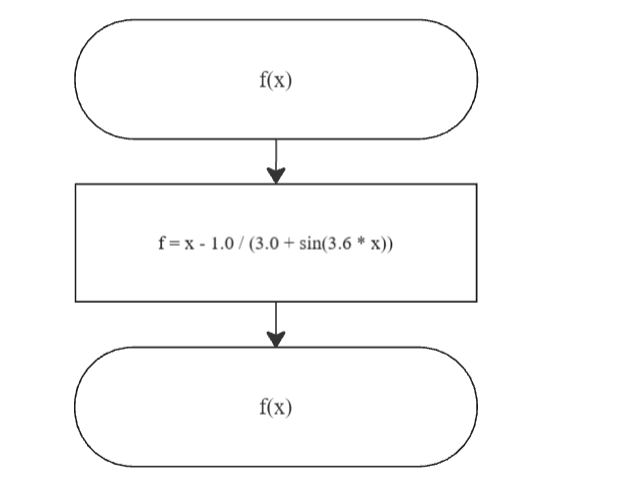


Рисунок 3 – Блок-схема функции F(x)

* 1. Код программы на языке С

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <errno.h>

/\*\*

\* @brief Функция для ввода значения типа double с проверкой корректности ввода.

\* @return Введенное значение типа double.

\* Если ввод некорректен, программа выводит сообщение об ошибке и завершает выполнение.

\*/

double inputDouble(void);

/\*\*

\* @brief вычисляет и возвращает значение функции f(x) = x - 1 / (3 + sin(3.6x))

\* в точке x.

\* Т.к. нет таких "x", при которых 3 + sin(3.6x) == 0,

\* значение может быть получено для произвольного "x" и нет необходимости

\* проверять возможность расчёта функции в точке.

\* @return значение функции в точке

\*/

double f(const double x);

/\*\*

\* @brief Точка входа в программу.

\* @return 0 в случае успеха.

\*/

int main(void) {

double xStart = inputDouble();

double xEnd = inputDouble();

double deltaX = inputDouble();

if (deltaX <= DBL\_EPSILON) {

errno = EIO;

perror("deltaX should be greater than 0");

return EXIT\_FAILURE;

}

printf("xStart = %lf, xEnd = %lf, deltaX = %lf\n\n", xStart, xEnd, deltaX);

for (double x = xStart; x <= xEnd + DBL\_EPSILON; x += deltaX) {

printf("f(%10.6lf) = %10.6lf\n", x, f(x));

}

return 0;

/\*\*

\* @brief Функция для ввода значения типа double.

\* @return Введенное значение типа double.

\* Проверяет корректность ввода. В случае ошибки устанавливает errno и завершает программу.

\*/

double inputDouble() {

double inputValue = 0;

int result = scanf("%lf", &inputValue);

if (result != 1) {

errno = EIO;

perror("Your input should be valid double");

exit(EXIT\_FAILURE);

}

return inputValue;

}

/\*\*

\* @brief Вычисляет значение функции f(x).

\* @param x Значение, в котором вычисляется функция.

\* @return Значение функции f(x) в точке x.

\*/

double f(const double x) {

return x - 1.0 / (3.0 + sin(3.6 \* x));

}

* 1. Результат выполнения программы

Ниже представлены результаты выполнения программы (Рисунок 4, Рисунок 5, Рисунок 6).

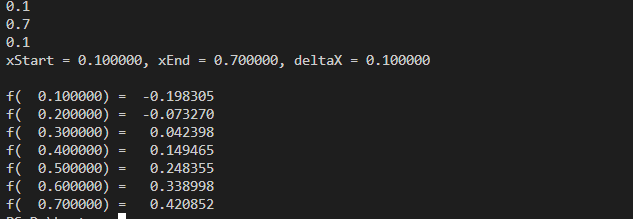


Рисунок 4 – Первый результат выполнения программы

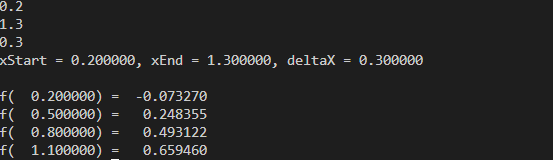


Рисунок 5 – Второй результат выполнения программы



Рисунок 6 – Проверка функции inputDouble()

* 1. Выполнение тестовых примеров

Ниже представлены результаты тестовых примеров на языке Python (Рисунок 7, Рисунок 8).

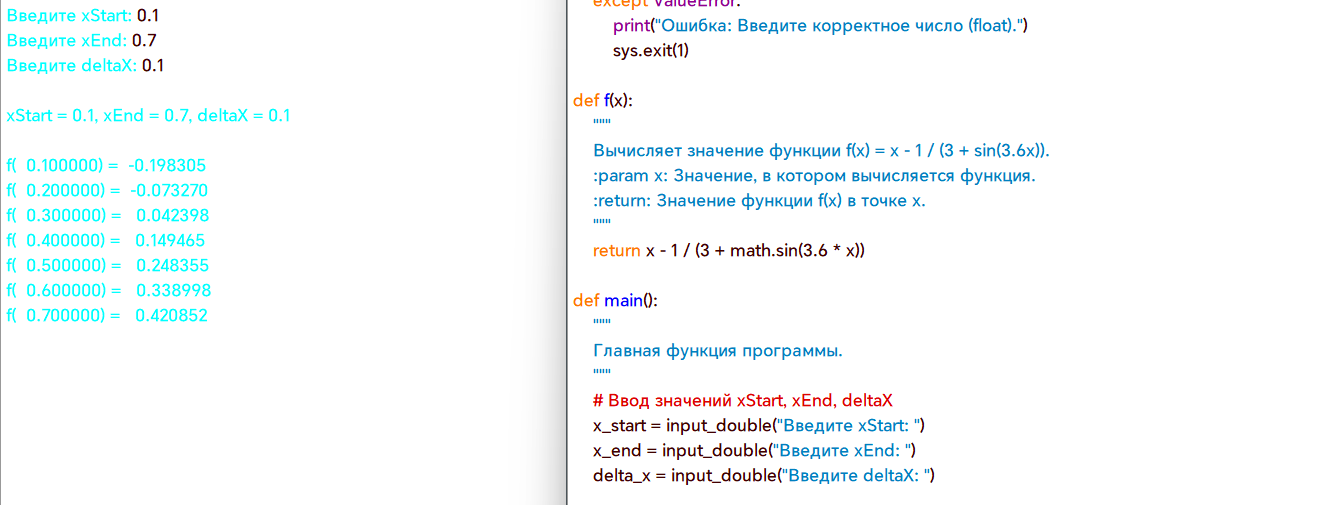


Рисунок 7 – Результат выполнения первого тестового примера

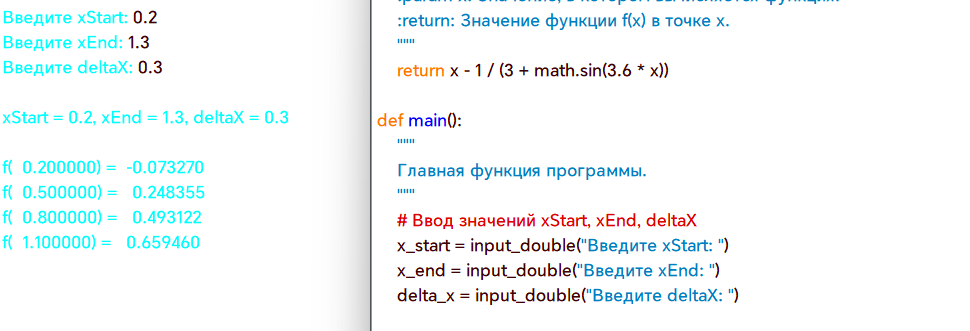


Рисунок 8 – Второй результат выполнения программы

* 1. Отметка о выполнение

Отметка о выполнение задания в GitHub (Рисунок 9).

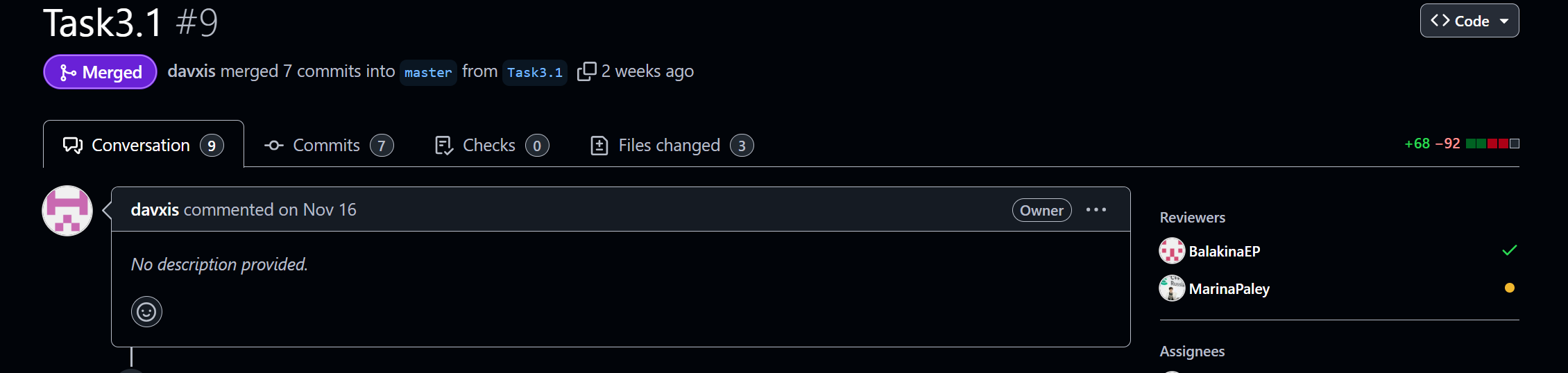


Рисунок 9 – Отметка о выполнение задания

1. задание 3.2
   1. Формулировка задания

Составьте две программы:

a) вычислить сумму первых n членов последовательности (k = 1, 2, 3 ..., n).

b) вычислить сумму всех членов последовательности, не меньших заданного числа e.

Помните о проверке пользовательского ввода. Все результаты вывести на экран. Отчёт дополнить блок-схемой. При вычислении факториалов рекомендуется отказаться от использования рекурсивных методов. ()

Таблица 2 – Исходные данные

|  |  |
| --- | --- |
| **Вариант** | **Ряд** |
| **10** |  |

* 1. Блок-схема основного алгоритма

Ниже представлена блок-схема основного алгоритма (Рисунок 10). Блок-схемы функций расчета (Рисунок 11, Рисунок 12, Рисунок 13, Рисунок 14).

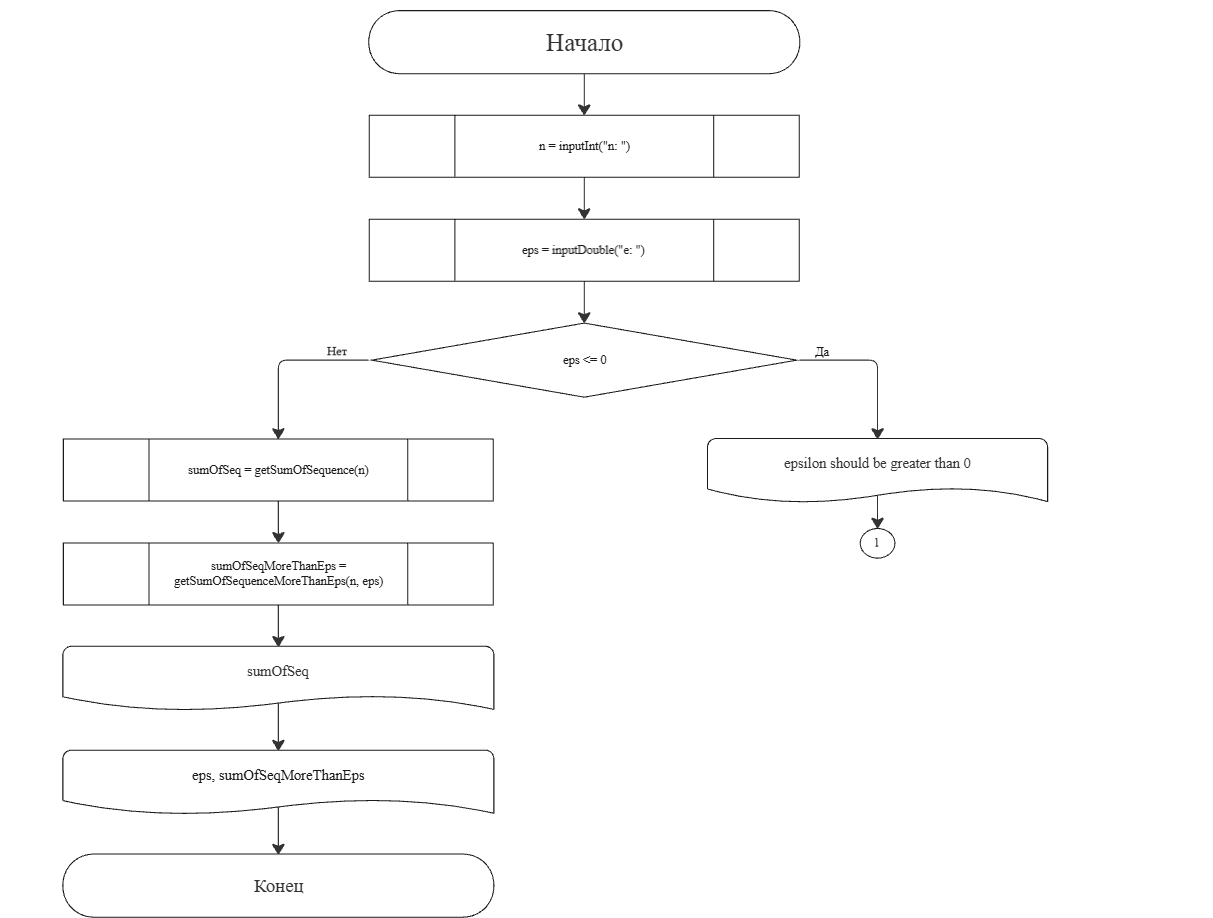


Рисунок 10 – Блок-схема основного алгоритма

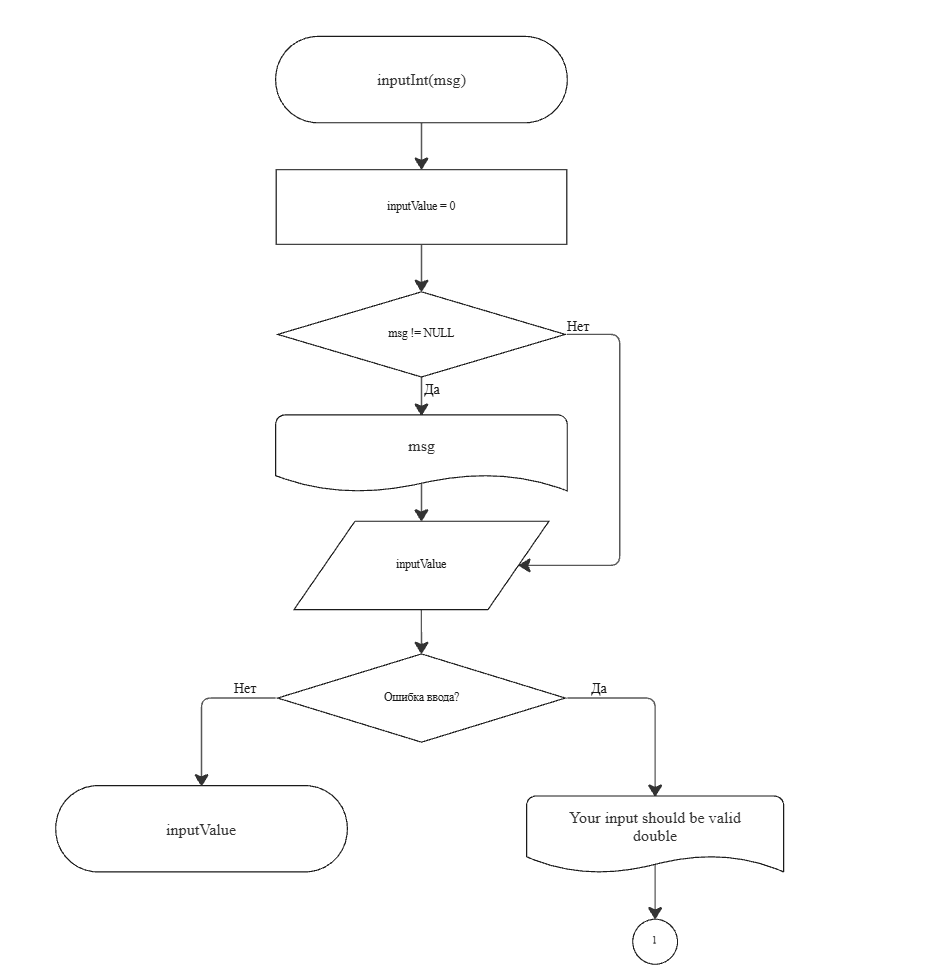


Рисунок 11 – Блок-схема функции inputInt

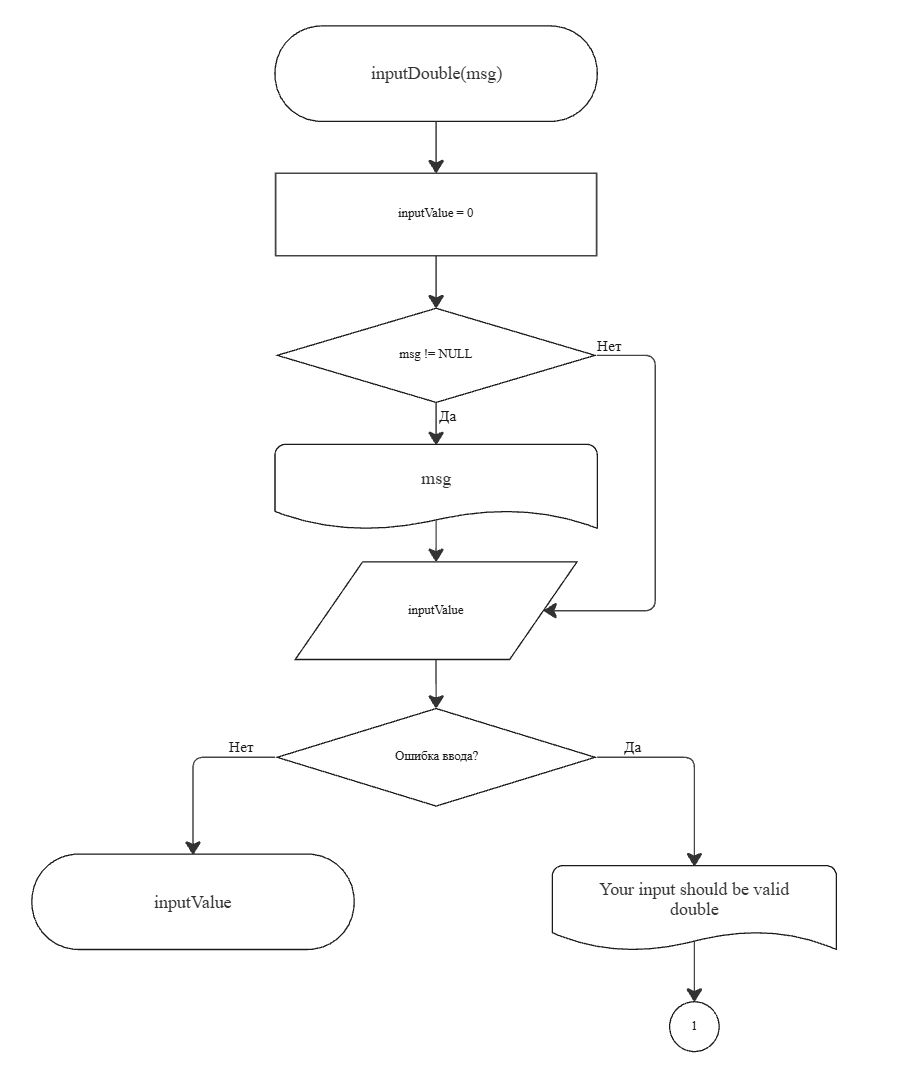


Рисунок 12 – Блок-схема функции inputDouble

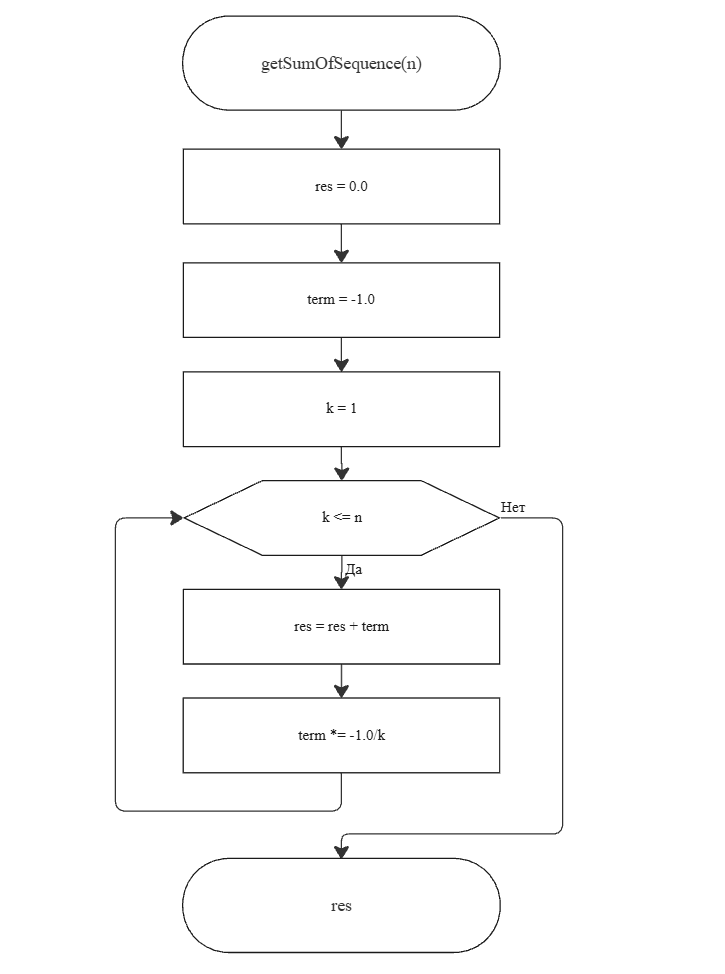


Рисунок 13 – Блок-схема функции getSumOfSequence(n)

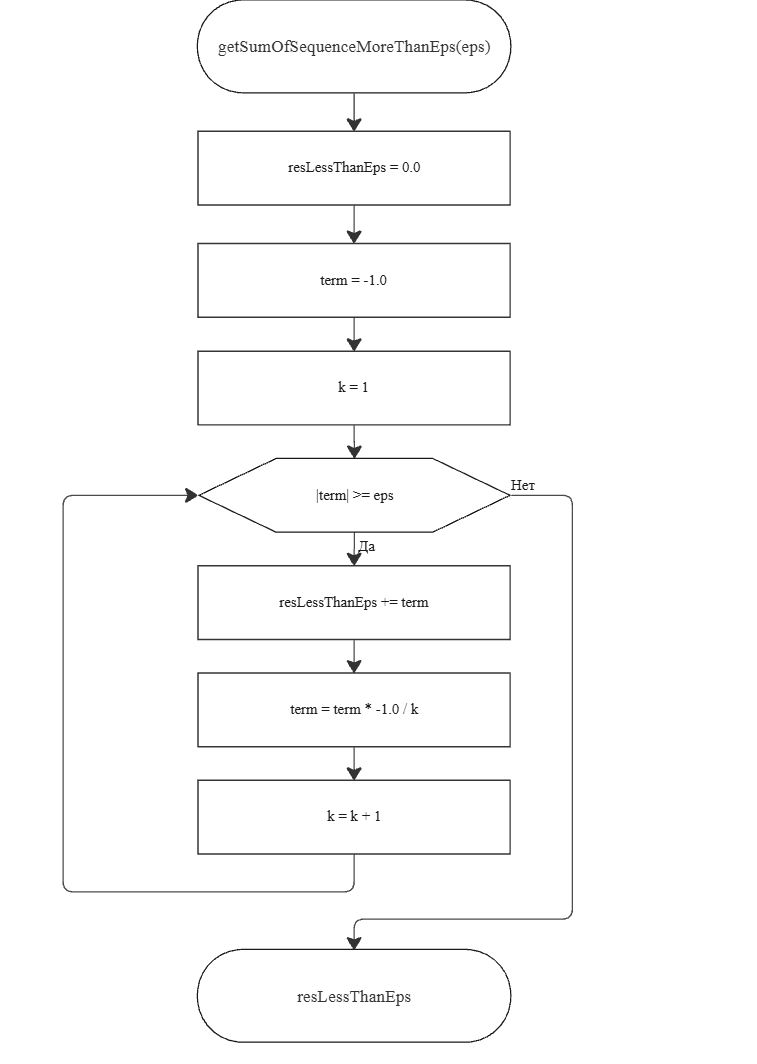


Рисунок 14 – Блок-схема функции getSumOfSequenceMoreThanEps(eps)

* 1. Текст программы на языке С

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

/\*\*

\* @brief Функция для ввода значения типа int с сообщением.

\* @param msg Сообщение, которое будет выведено перед вводом значения.

\* @return Введенное значение типа int.

\* Если ввод некорректен, программа выводит сообщение об ошибке и завершает выполнение.

\*/

int inputInt(const char\* msg);

/\*\*

\* @brief Запрашивает у пользователя ввод числа с плавающей запятой (double).

\* Эта функция отображает сообщение, переданное в качестве аргумента,

\* и ожидает ввода числа с плавающей запятой. Если введенное значение

\* не является корректным числом, функция будет запрашивать ввод до

\* тех пор, пока не будет введено корректное значение.

\* @param msg Указатель на строку (const char\*), содержащую сообщение,

\* которое будет отображаться пользователю при запросе ввода.

\* @return Возвращает введенное пользователем число с плавающей запятой

\* типа double.

\*/

double inputDouble(const char\* msg);

/\*\*

\* @brief Вычисляет сумму членов последовательности.

\* @param n Число элементов последовательности.

\*/

double getSumOfSequence(const int n);

/\*\*

\* @brief Вычисляет сумму членов последовательности,

\* которые выше по модулю чем epsilon.

\* @param n Число элементов последовательности.

\* @param eps Значение epsilon.

\*/

double getSumOfSequenceMoreThanEps(const double eps);

/\*\*

\* @brief Точка входа в программу.

\* @return 0 в случае успеха.

\*/

int main(void) {

int n = inputInt("n: ");

double eps = inputDouble("e: ");

if (eps <= DBL\_EPSILON) {

errno = EIO;

perror("epsilon should be greater than 0");

return EXIT\_FAILURE;

}

double sumOfSeq = getSumOfSequence(n);

double sumOfSeqMoreThanEps = getSumOfSequenceMoreThanEps(n, eps);

printf("sum from k=0 to k=n is equal to %8.5lf\n", sumOfSeq);

printf("sum from k=0 to k=n where term greater or equal than %8.5lf is equal to %8.5lf\n",

eps, sumOfSeqMoreThanEps);

return 0;

}

/\*\*

\* @brief Функция для ввода значения типа int с сообщением.

\* @param msg Сообщение, которое будет выведено перед вводом значения.

\* @return Введенное значение типа int.

\* Если ввод некорректен, программа выводит сообщение об ошибке и завершает выполнение.

\*/

int inputInt(const char\* msg) {

int inputValue = 0;

if (msg != NULL) {

printf(msg);

}

int result = scanf("%d", &inputValue);

if (result != 1) {

errno = EIO;

perror("Your input should be valid int\n");

exit(EXIT\_FAILURE);

}

return inputValue;

}

double inputDouble(const char\* msg) {

double inputValue = 0;

if (msg != NULL) {

printf(msg);

}

int result = scanf("%lf", &inputValue);

if (result != 1) {

errno = EIO;

perror("Your input should be valid double");

exit(EXIT\_FAILURE);

}

return inputValue;

}

/\*\*

\* @brief Вычисляет сумму членов последовательности.

\* @param n Число элементов последовательности.

\*/

double getSumOfSequence(int n) {

double res = 0.0;

double term = -1.0;

for (int k = 1; k <= n; k++) {

res += term;

term \*= -1.0 / k;

}

return res;

}

/\*\*

\* @brief Вычисляет сумму членов последовательности,

\* которые выше по модулю чем epsilon.

\* @param n Число элементов последовательности.

\* @param eps Значение epsilon.

\*/

double getSumOfSequenceMoreThanEps(const double eps) {

double resLessThanEps = 0.0;

double term = -1.0;

for (int k = 1; abs(term) >= eps+DBL\_EPSILON; k++)

{

resLessThanEps += term;

term \*= -1.0 / k;

}

return resLessThanEps;

}

* 1. Результат выполнения программы

Ниже представлен результат выполнения программы (Рисунок 15, Рисунок 16, Рисунок 17, Рисунок 18).

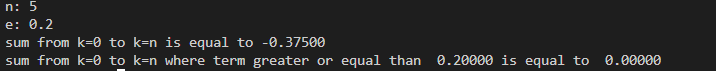


Рисунок 15 – Первый результат выполнения программы

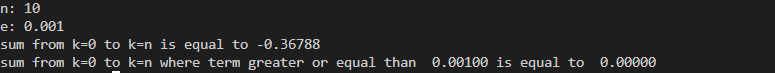


Рисунок 16 – Второй результат выполнения программы



Рисунок 17 – Проверка функции inputInt



Рисунок 18 – Проверка функции inputDouble

* 1. Выполнение тестовых примеров

Ниже представлены результаты выполнения тестовых примеров на языке Python (Рисунок 19, Рисунок 20).

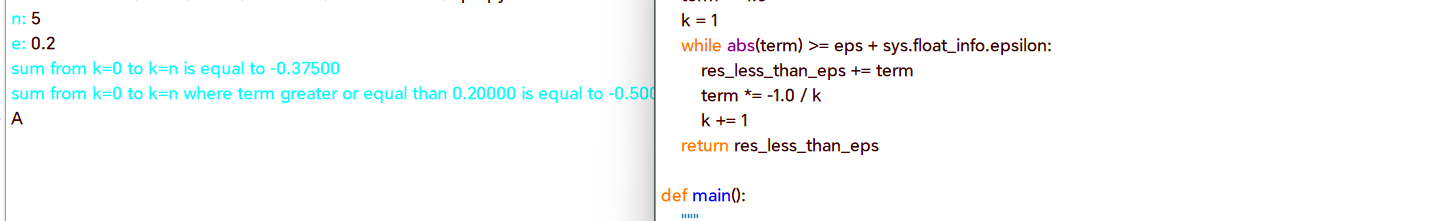


Рисунок 19 – Выполнение первого тестового примера

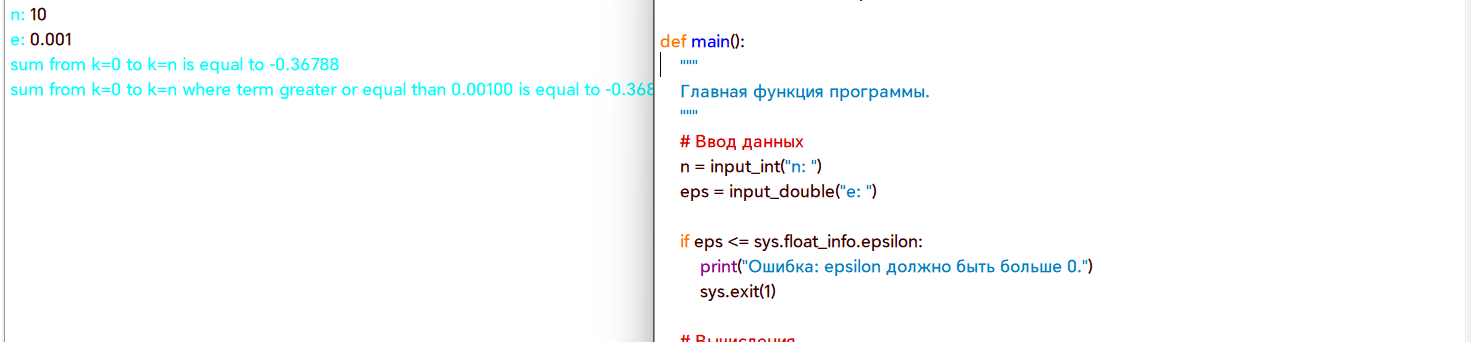


Рисунок 20 – Выполнение второго тестового примера

* 1. Отметка о выполнение задания

Отметка о выполнение задания в GitHub (Рисунок 21).

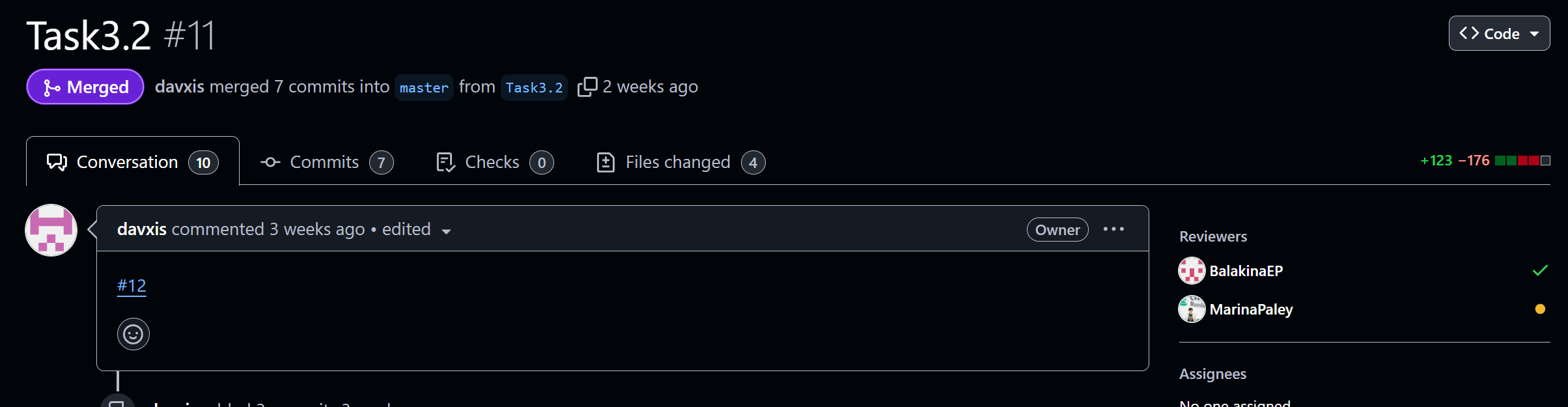


Рисунок 21 – Отметка о выполнение задания