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 8

Выполнил: ст. гр. ТКИ-141

Гулин Игорь Алексеевич

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

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

Москва 2024

Оглавление

[**1. Задание 3.2 3**](#_Toc185180120)

[1.1. Формулировка задания 3](#_Toc185180121)

[**1.2.** **Блок-схема алгоритма 4**](#_Toc185180122)

[1.3. Код задания 3.2 8](#_Toc185180123)

[**1.4. Решение тестового примера 10**](#_Toc185180124)

[**1.5. Решение тестовых примеров в Excel 11**](#_Toc185180125)

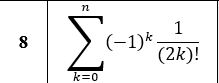
[1.6. Зачёт задания в GitHub 11](#_Toc185180126)

# Задание 3.2

* 1. Формулировка задания

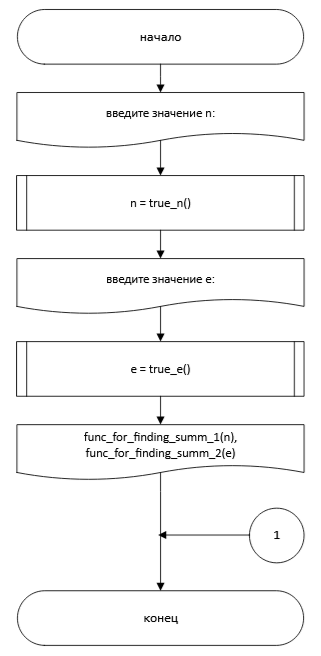
1. вычислить сумму первых *n* членов последовательности (*k* = 1, 2, 3 ..., *n*).
2. вычислить сумму всех членов последовательности, не меньших заданного числа *e*.

Таблица 2 – Формулировка задания 3.2

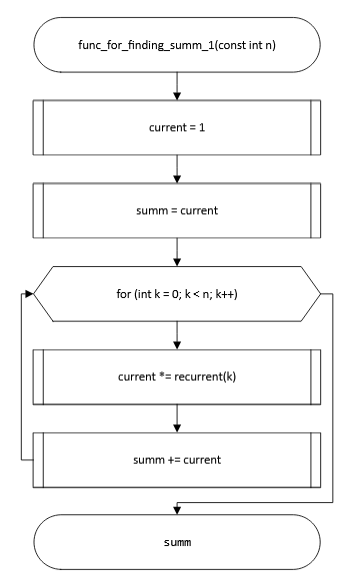


* 1. Блок-схема алгоритма

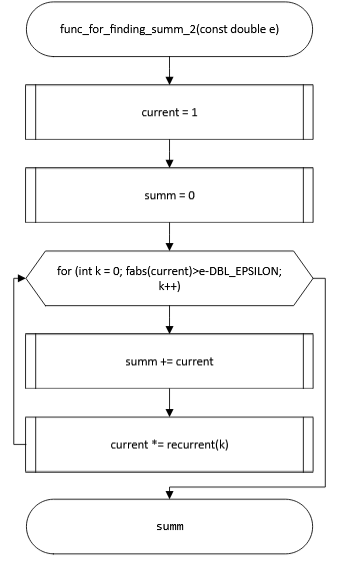
Блок-схема алгоритма представлена на рисунках (***Рисунок 1***, ***Рисунок 2***, ***Рисунок 3***, ***Рисунок 4***, ***Рисунок 5***, ***Рисунок 6***, ***Рисунок 7***)



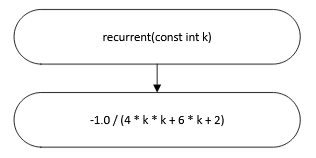
**Рисунок 1. Блок-схема функции main()**

**

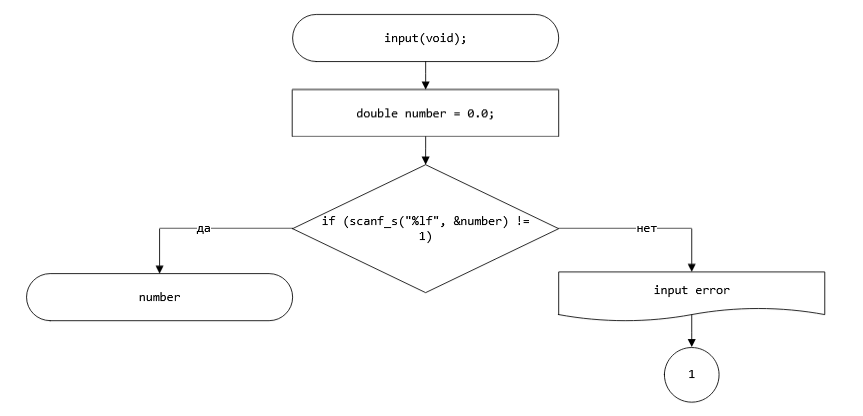
**Рисунок 2. Блок-схема функции func\_for\_finding\_summ\_1(const int n)**

**

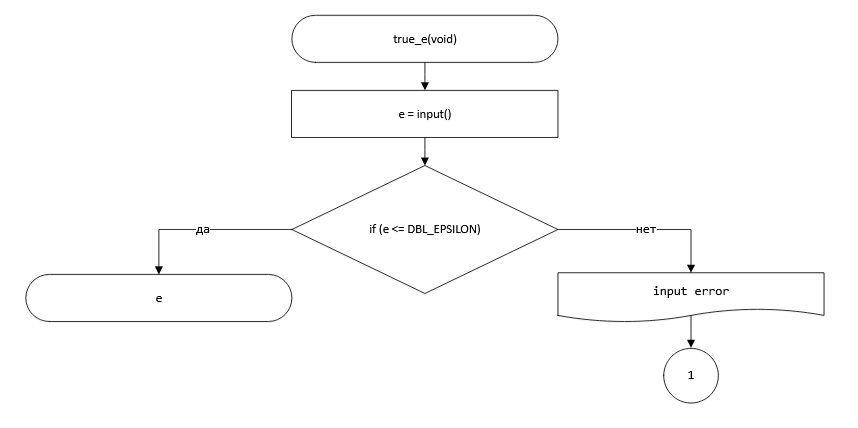
**Рисунок 3. Блок-схема функции func\_for\_finding\_summ\_2(const double e)**

**

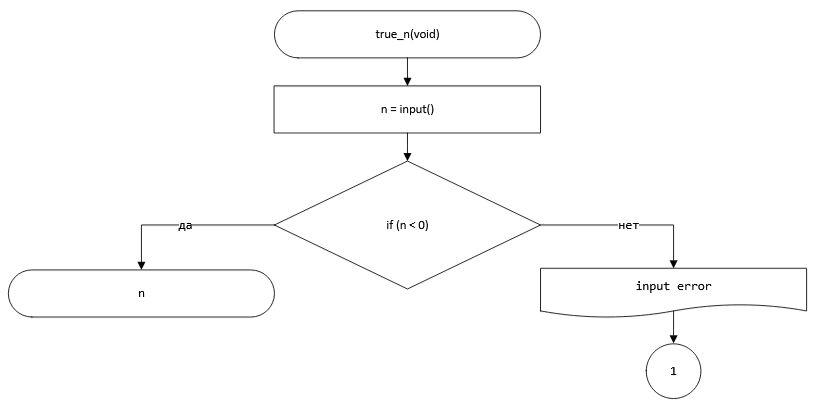
**Рисунок 4. Блок-схема функции recurrent(const int k)**

**

**Рисунок 5. Блок-схема функции input()**

**

**Рисунок 6. Блок-схема функции true\_e()**

**

**Рисунок 7. Блок-схема функции true\_n()**

* 1. Код задания 3.2

#include <stdio.h>

#include <stdlib.h>

#include <float.h>

#include <math.h>

/\*\*

\* @brief вычисляет сумму первых n членов последовательности (k = 1, 2, 3 ..., n).

\* @param n значение последнего члена последовательности

\* @return возвращает сумму первых n членов последовательности (k = 1, 2, 3 ..., n)

\*/

double func\_for\_finding\_summ\_1(const int n);

/\*\*

\* @brief вычисляет сумму всех членов последовательности, не меньших заданного числа e

\* @param e значение параметра e

\* @return возвращает сумму всех членов последовательности, не меньших заданного числа e

\*/

double func\_for\_finding\_summ\_2(const double e);

/\*\*

\* @brief вычисляет рекуррентное соотношение относительно данного значения элемента последовательности

\* @param k значение данного элемента последовательности

\* @return возвращает полученное значение

\*/

double recurrent(const int k);

/\*\*

@brief проверяет вводимое значение, является ли оно числом и больше ли нуля

@return Возвращает введённое значение, если оно является числом и больше нуля

\*/

double input(void);

/\*\*

@brief проверяет параметр e, больше ли нуля, и вводит его

@return возвращает значение e если оно больше нуля

\*/

double true\_e(void);

/\*\*

@brief проверяет параметр n, больше ли нуля, и вводит его

@return возвращает значение n если оно больше нуля

\*/

int true\_n(void);

/\*\*

\* @brief Точка входа в программу.

\* @return Возвращает в случае успеха.

\*/

int main(void) {

printf("введите значение n:\t");

int n = true\_n();

printf("введите значение e:\t");

double e = true\_e();

printf("%lf\n%lf",func\_for\_finding\_summ\_1(n), func\_for\_finding\_summ\_2(e));

return 0;

}

double func\_for\_finding\_summ\_1(const int n) {

double current = 1;

double summ = current;

for (int k = 0; k < n; k++) {

current \*= recurrent(k);

summ += current;

}

return summ;

}

double func\_for\_finding\_summ\_2(const double e) {

double current = 1;

double summ = 0;

for (int k = 0; fabs(current)>e-DBL\_EPSILON; k++) {

summ += current;

current \*= recurrent(k);

}

return summ;

}

double recurrent(const int k) {

return -1.0 / (4 \* k \* k + 6 \* k + 2);

}

double input(void) {

double number = 0;

if (scanf\_s("%lf", &number) != 1) {

printf("input error");

exit(EXIT\_FAILURE);

}

return number;

}

double true\_e(void) {

double e = input();

if (e <= DBL\_EPSILON) {

printf("input error");

exit(EXIT\_FAILURE);

}

return e;

}

int true\_n(void) {

int n = input();

if (n < 0) {

printf("input error");

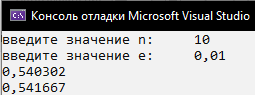
exit(EXIT\_FAILURE);

}

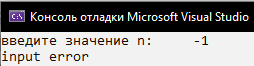
return n;

}

* 1. Решение тестового примера

****

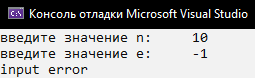
**Рисунок 8. Решение тестового примера**

****

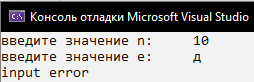
**Рисунок 9. Решение тестового примера, когда параметр n отрицателен**

****

**Рисунок 10. Решение тестового примера, когда параметр n буква**

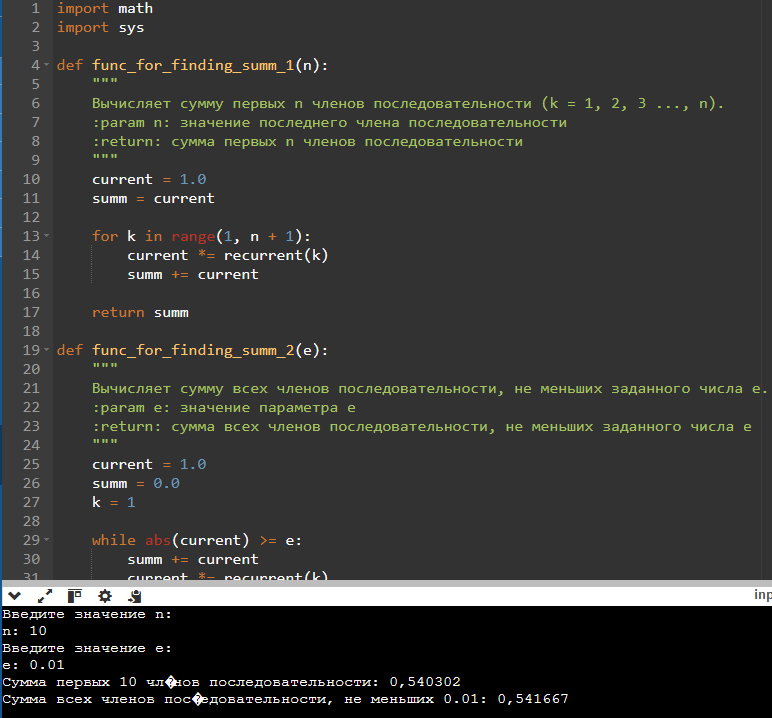
****

**Рисунок 11. Решение тестового примера, когда параметр e отрицателен**

****

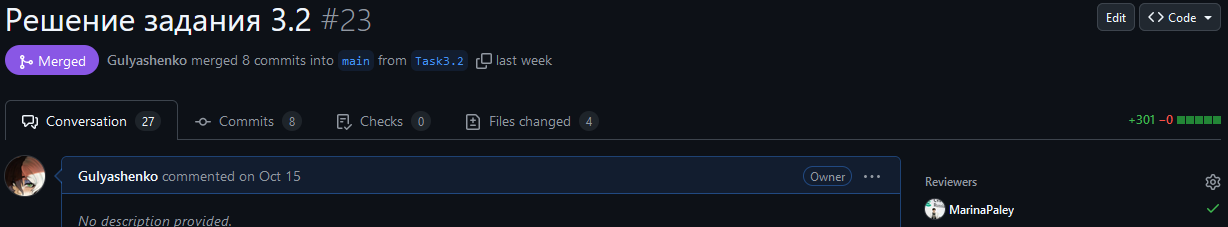
**Рисунок 12. Решение тестового примера, когда параметр e буква**

* 1. Решение тестовых примеров в Python



**Рисунок 13. Решение тестового примера в Python**

* 1. Зачёт задания в GitHub



**Рисунок 14. Зачёт задания в GitHub**