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 12

Выполнил: ст. гр. ТКИ-142

Наседкин Тимофей Филиппович

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

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

Москва 2023

**ОГЛАВЛЕНИЕ**

**1.Решение задачи 3-1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.3**

**1.1. Формулировка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.3**

**1.2. Блок-схема алгоритма\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.4-5**

**1.3. Код программы на языке C\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.6-7**

**1.4. Результаты выполнения программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.8**

**1.5. Выполнение тестовых примеров\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.9**

**1.6. Отметка о выполнении задания в веб-хостинге системы контроля версий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.10**

**2.Решение задачи 3-2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.11**

**2.1. Формулировка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.11**

**2.2. Блок-схема алгоритма\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.12-13**

**2.3. Код программы на языке C\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.14-15**

**2.4. Результаты выполнения программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.16**

**2.5. Выполнение тестовых примеров \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.17**

**2.6. Отметка о выполнении задания в веб-хостинге системы контроля версий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стр.18**

* 1. **Решение задачи 3-1**
  2. **Формулировка задачи**

Протабулировать заданную в таблице функцию. 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.

При решении данного задания в MatLab необходимо построить график!

Таблица 1 – Условие задачи 2-1

| Вариант | Функция | Константы |
| --- | --- | --- |

| 12 |  |  |
| --- | --- | --- |

# 

* 1. **Блок-схема алгоритма**

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

(Рисунок 4), (Рисунок 5), (Рисунок 6).

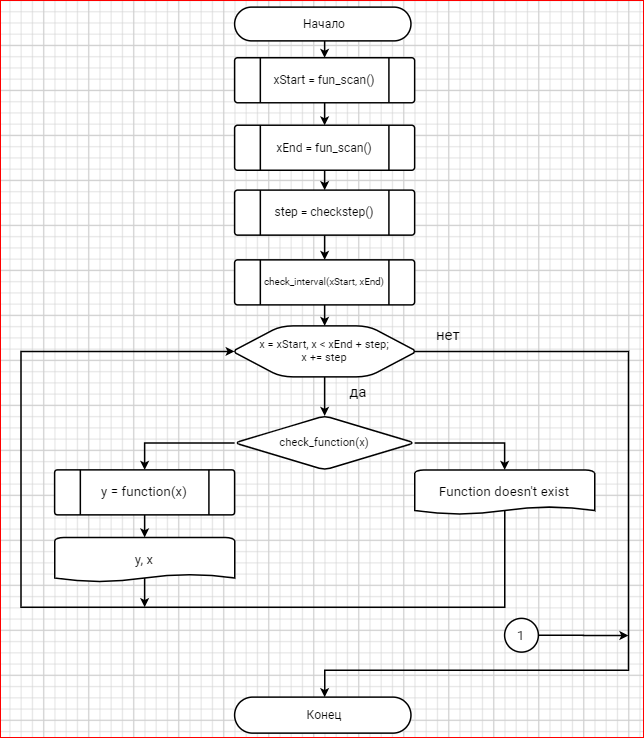


Рисунок 1 - Блок-схема основного алгоритма.

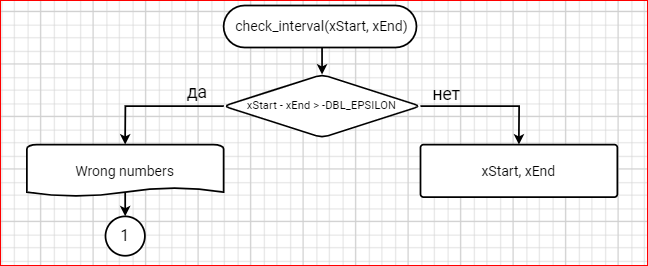


Рисунок 2 – Блок-схема функции check\_interval().

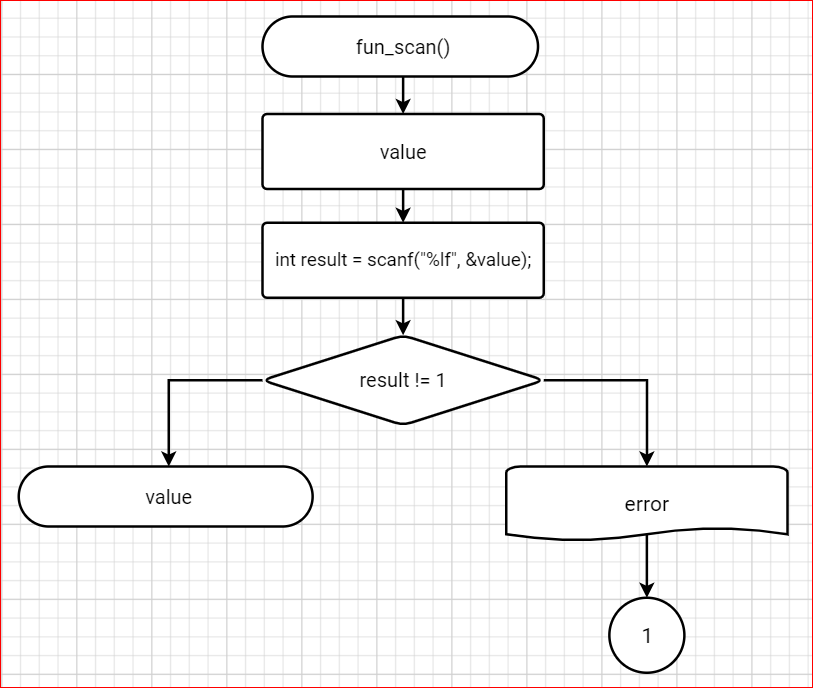


Рисунок 3 - Блок-схема функции fun\_scan().

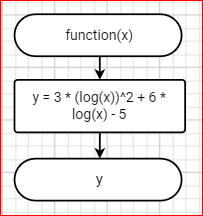


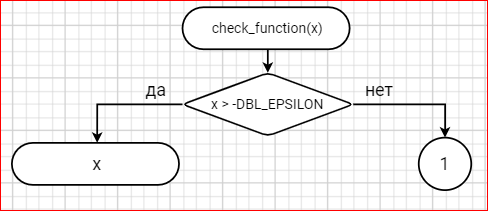
Рисунок 4 - Блок-схема функции function().

Рисунок 5 - Блок-схема функции check\_function().

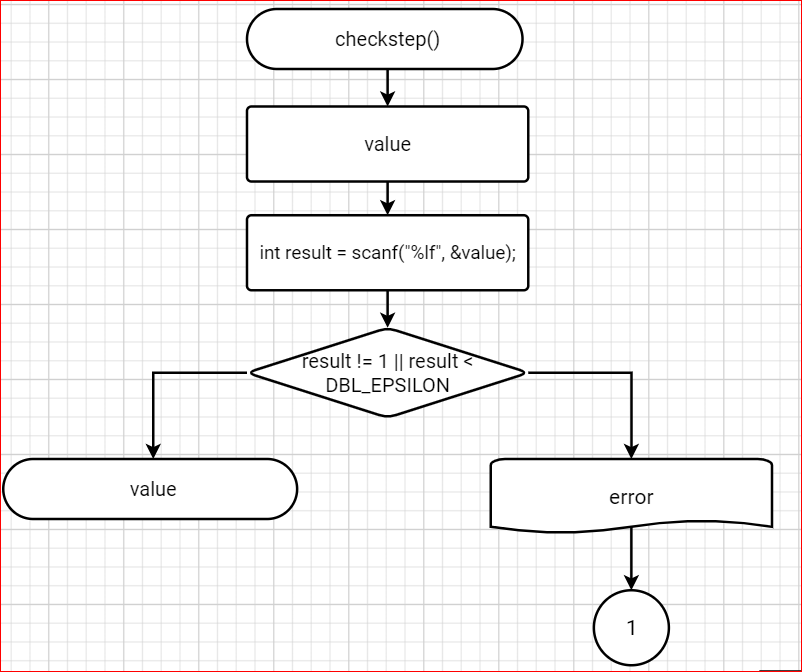


Рисунок 6 - Блок-схема функции checkstep().

* 1. **Код программы на языке C**

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <float.h>

#include <stdbool.h>

/\*\*

\* @brief Функция расчета формулы

\* @param x значение параметра

\* @return Значение вычисления y

\*/

float function(float x);

/\*\*

\* @brief Функция проверки ввода на число

\* @return Результат проверки

\*/

double fun\_scan();

/\*\*

\* @brief Функция проверки ввода шага функции на положительное число

\* @return Результат проверки

\*/

double checkstep();

/\*\*

\* @brief Функция проверки ввода концов отрезка

\* @param xStart, xEnd значения вводимыъх параметров концов отрезка

\*/

void check\_interval(float xStart, float xEnd);

/\*\*

\* @brief Функция проверки существования функции

\* @param x значение вводимого параметра фунуции

\*/

bool check\_function(float x);

/\*\*

\* @brief Точка входа в программу

\* @return Вывод 0, если программа работает неверно. Иначе вывод 1

\*/

int main()

{

float xStart = fun\_scan(), xEnd = fun\_scan(), step = checkstep();

check\_interval(xStart, xEnd);

for (float x = xStart; x < xEnd + step; x += step)

{

if (check\_function(x))

{

printf ("y=%f\n x=%f\n", function(x), x);

}

else

{

puts("Function doesn't exist");

}

}

return 0;

}

float function(float x)

{

return (3 \* pow(log(x), 2) + 6 \* log(x) - 5);

}

void check\_interval(float xStart, float xEnd)

{

if (xStart - xEnd > -DBL\_EPSILON)

{

puts("Wrong numbers");

abort();

}

}

bool check\_function(float x)

{

return x > -DBL\_EPSILON;

}

double fun\_scan()

{

double value;

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

if (result != 1)

{

puts("error io");

abort();

}

return value;

}

double checkstep()

{

double value;

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

if ((result != 1)||(result < DBL\_EPSILON))

{

puts("error io");

abort();

}

return value;

}

* 1. **Результаты выполнения программы**

Результаты выполнения программы представлены ниже (Рисунок 7).

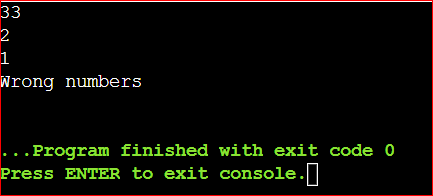


Рисунок 7 – Результаты выполнения программы

* 1. **Выполнение тестовых примеров**

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 8).

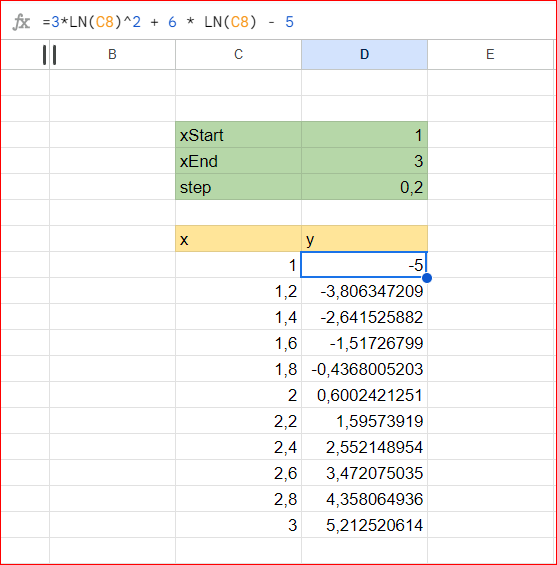


Рисунок 8 - Результат нахождения значения функции.

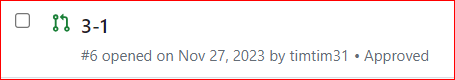
* 1. **Отметка о выполнении задания в веб-хостинге системы контроля версий** 

Рисунок 9 – Approve задачи 3-1.

1. **Решение задачи 3-2**
   1. **Формулировка задачи**

Составьте две программы:

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

Таблца 2 – Условие задачи 3-2

| Вар | Ряд |
| --- | --- |

| 12 |  |
| --- | --- |

1. **Блок-схема алгоритма**

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы дополнительных функций представлены ниже (Рисунок 2, Рисунок 3, Рисунок 4, Рисунок 5).

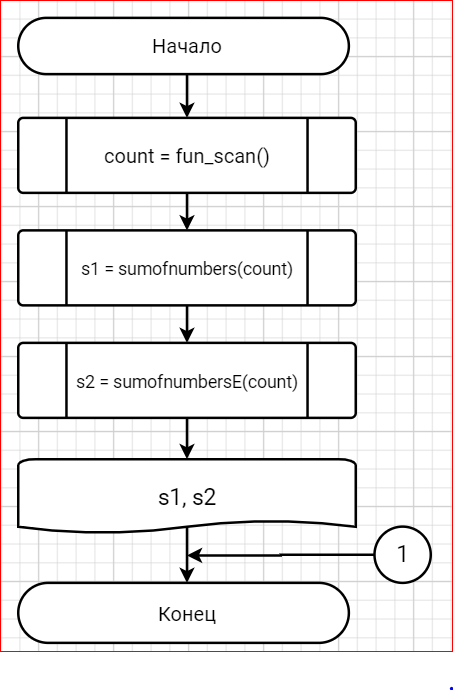


Рисунок 1 - Блок-схема основного алгоритма.

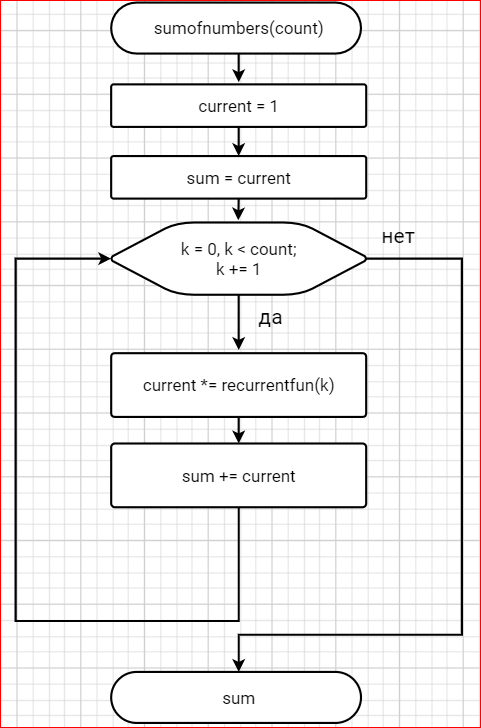
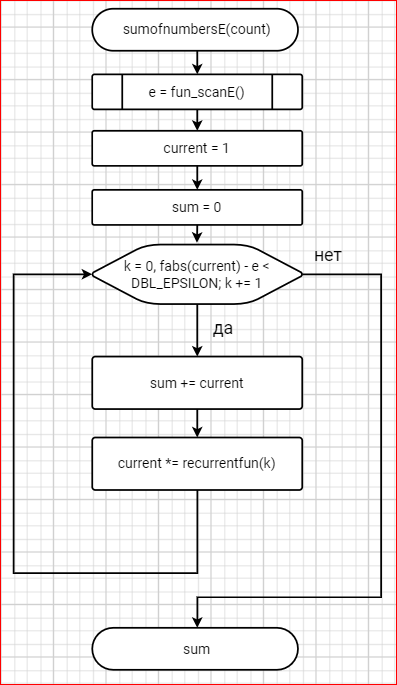


Рисунок 2 - Блок-схема функции sumofnumbers(), Блок-схема функции sumofnumbersE().

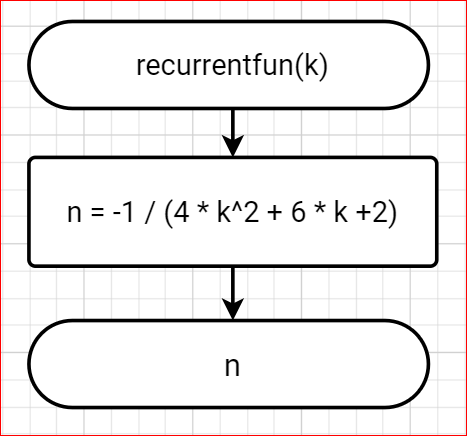


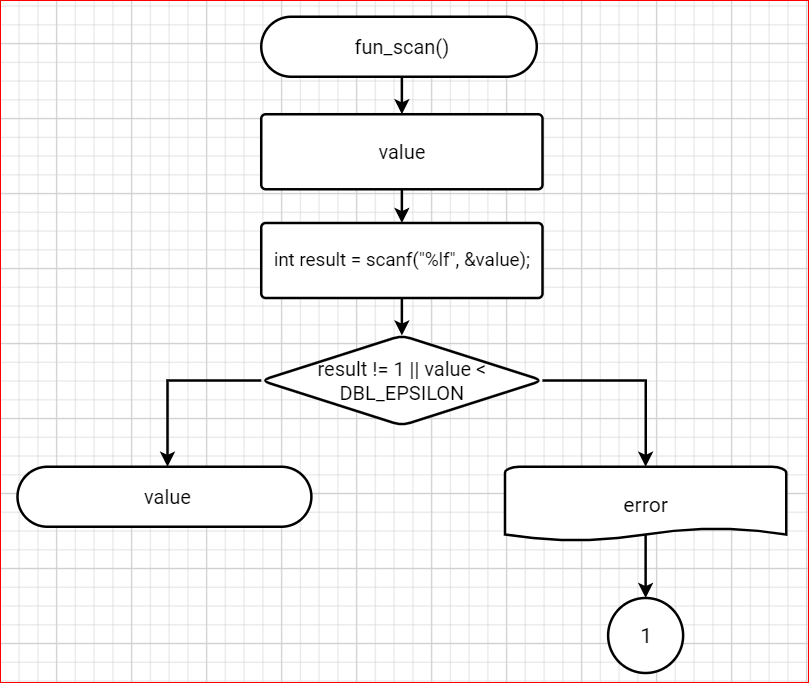
Рисунок 3 - Блок-схема функции recurrentfun().

Рисунок 4 - Блок-схема функции fun\_scan().

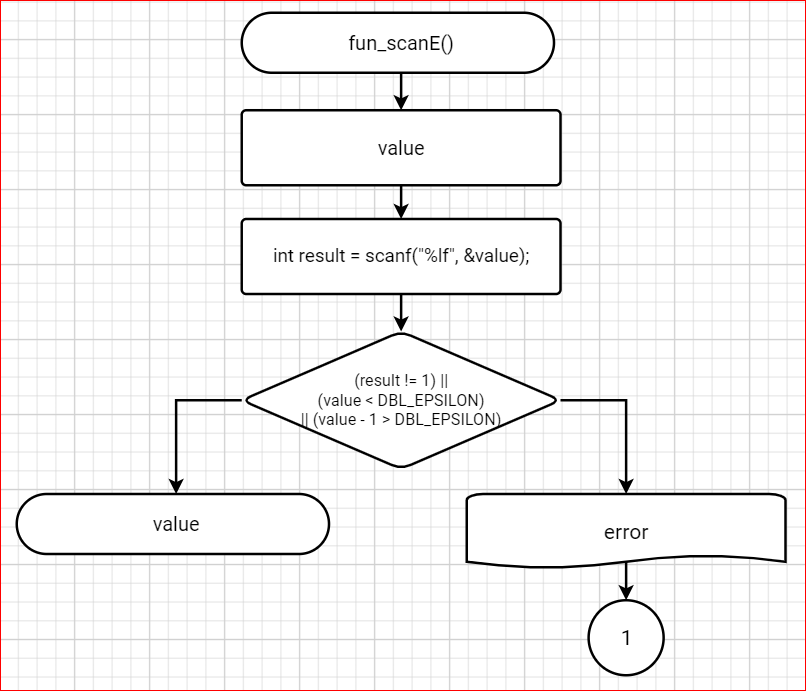


Рисунок 5 - Блок-схема функции fun\_scanE().

1. **Код программы на языке C**

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <float.h>

/\*\*

\* @brief Функция проверки ввода на положительное число

\* @return Результат проверки

\*/

int fun\_scan();

/\*\*

\* @brief Функция проверки ввода на положительное число и меньшее модулю первого члена последовательности

\* @return Результат проверки

\*/

double fun\_scanE();

/\*\*

\* @brief Функция расчета суммы последовательности чисел

\* @param count вводимое значение количества чисел последовательности

\* @return sum значение вычисления суммы последовательности

\*/

double sumofnumbers(int count);

/\*\*

\* @brief Функция расчета суммы последовательности чисел, каждый элемент которой должен быть больше заданного е

\* @param count вводимое значение количества чисел последовательности

\* @param е вводимое значение минимального возможного числа последовательности

\* @return sum значение вычисления суммы последовательности

\*/

double sumofnumbersE(int count);

/\*\*

\* @brief Функция расчета рекурентного члена последовательности

\* @param k - номер члена последовательности

\* @return Значение рекурентного члена последовательности

\*/

double recurrentfun(int k);

/\*\*

\* @brief Точка входа в программу

\* @return Вывод 0, если программа работает неверно. Иначе вывод 1

\*/

int main()

{

int count = fun\_scan();

printf("%f\n", sumofnumbers(count));

printf("%f\n", sumofnumbersE(count));

return 0;

}

double sumofnumbers(int count)

{

double current = 1;

double sum = current;

for (int k = 0; k < count; k++)

{

current \*= recurrentfun(k);

sum += current;

}

return sum;

}

double sumofnumbersE(int count)

{

double e = fun\_scanE();

double current = 1;

double sum = 0;

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

{

sum += current;

current \*= recurrentfun(k);

}

return sum;

}

double recurrentfun(int k)

{

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

}

int fun\_scan()

{

int value;

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

if ((result != 1) || (value < DBL\_EPSILON))

{

puts("error");

abort();

}

return value;

}

double fun\_scanE()

{

double value;

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

if ((result != 1) || (value < DBL\_EPSILON) || (value - 1 > DBL\_EPSILON))

{

puts("error");

abort();

}

return value;

}

* 1. **Результаты выполнения программы**

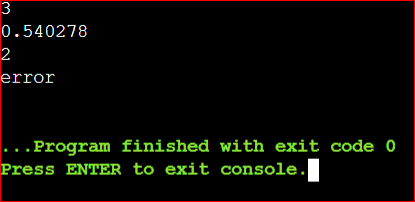
Результаты выполнения программы представлены ниже (Рисунок 6)

Рисунок 6 - результаты выполнения программы.

* 1. **Выполнение тестовых примеров**

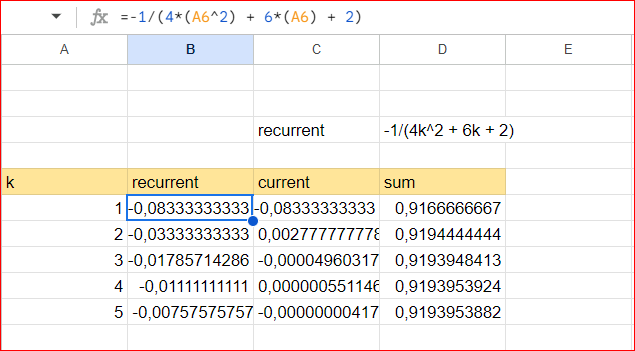
В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 7).

Рисунок 7 - Результат вычисления суммы последовательности.

**Отметка о выполнении задания в веб-хостинге системы контроля версий**

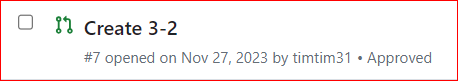


Рисунок 8 – Approve задачи 3-2