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 2

Выполнил: ст. гр. ТКИ - 142

Бахров Кирилл Сергеевич

Проверил: к.т.н., доц. Васильева М. А Проверил: к.т.н., доц. Балакина Е. П.

Москва 2023

**Оглавление**

**Решение задачи 3.14**

Формулировка задания4

Блок-схема алгоритма5-6

Текст программы на языке C7-9

Результат выполнения программы10

Выполнение тестовых примеров11

Отметка о выполнении задания в веб-хостинге системы контроля версий12

**Решение задачи 3.1 доп13**

Формулировка задания13

Блок-схема алгоритма14

Текст программы на языке C15

Результат выполнения программы16

Выполнение тестовых примеров17

Отметка о выполнении задания в веб-хостинге системы контроля версий18

**Решение задачи 3.2**1**9**

Формулировка задания19

Блок-схема алгоритма20-21

Текст программы на языке C22-25

Результат выполнения программы26

Выполнение тестовых примеров27-28

Отметка о выполнении задания в веб-хостинге системы контроля версий29

**Решение задачи 3.2 доп**30

Формулировка задания30

Блок-схема алгоритма31

Текст программы на языке C32-33

Результат выполнения программы34

Выполнение тестовых примеров35

Отметка о выполнении задания в веб-хостинге системы контроля версий36  
**Решение задачи 3.3**37

Формулировка задания37

Блок-схема алгоритма38-39

Текст программы на языке C40-43

Результат выполнения программы44

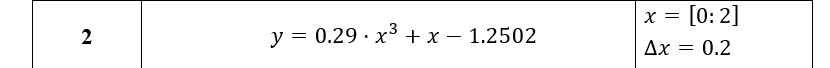
Выполнение тестовых примеров45

Отметка о выполнении задания в веб-хостинге системы контроля версий46

1. РЕШЕНИЕ ЗАДАЧИ 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. При решении данного задания в MatLab необходимо построить график!

Таблица  – Исходные данные



* 1. БЛОК-СХЕМА АЛГОРИТМА

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы используемых функций представлены ниже (Рисунки 2-5).

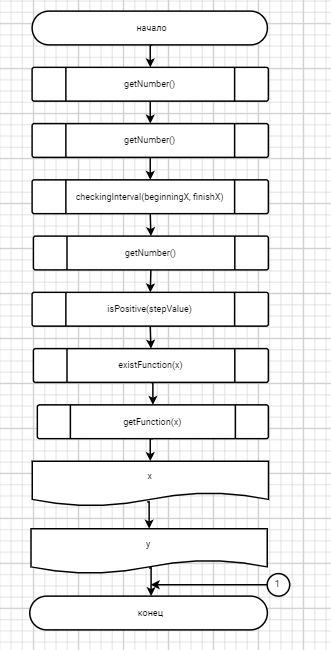
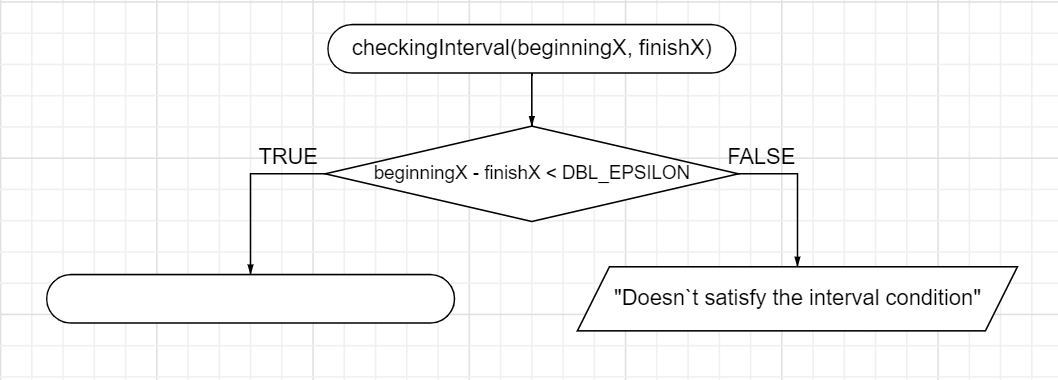
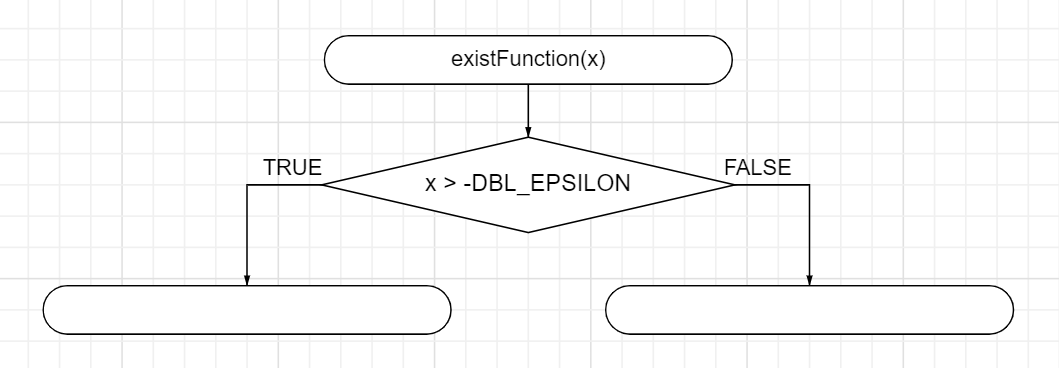
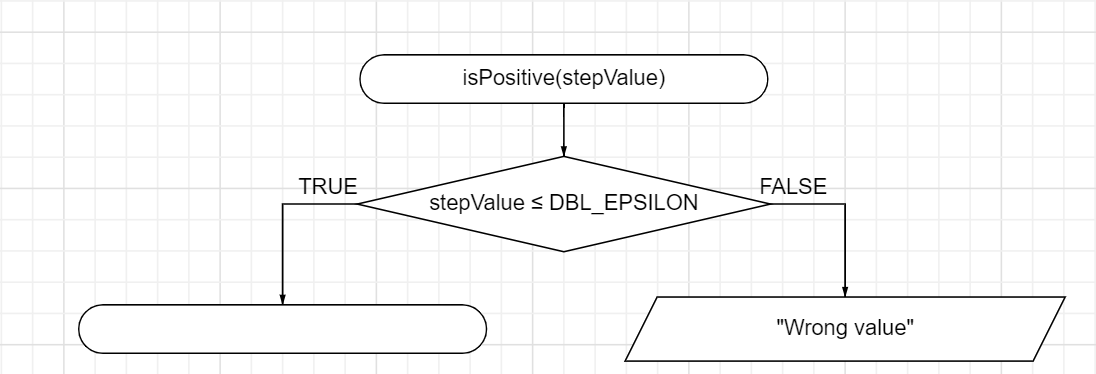
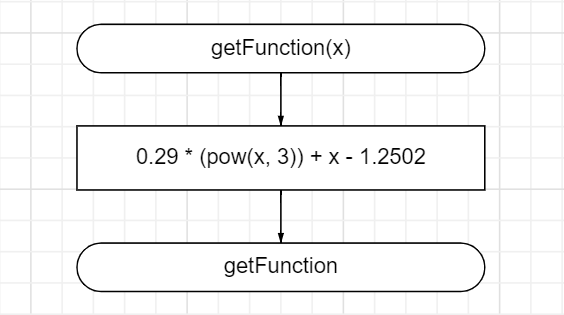


Рисунок  ­ Блок-схема основного алгоритма  
  
  
  
Рисунок  – Блок-схема используемых функций  
Рисунок 3 – Блок-схема используемых функций  
Рисунок 4 – Блок-схема используемых функций  
   
Рисунок 5 – Блок-схема используемых функций

* 1. ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <stdbool.h>

#include <float.h>

/\*\*

@brief считывает и проверияет ввод переменной

@return возвращает значение переменной

\*/

double getNumber();

/\*\*

@brief проверяет условие, чтобы начальное значение интервала было меньше конечного

@param beginningX - значение начала интервала

@param finishX - значение rjywf интервала

@return возвращает 1 если условие выполнено

\*/

double checkingInterval(double beginningX, double finishX);

/\*\*

@brief проверка значение на положительное число

@param stepValue - значение переменной stepValue

@return возвращает значение переменной

\*/

void isPositive(double stepValue);

/\*\*

@brief проверяет существует ли в функция в заданной точке

@param x - значение переменной x

@return возвращает true если функция существует

\*/

bool existFunction(double x);

/\*\*

@brief рассчитывает значение функции

@param x - значение переменной x

@return возвращает посчитанное значение

\*/

double getFunction(double x);

/\*\*

@brief точка входа в программу

@return возвращает 0, если программа выполнена верно

\*/

int main()

{

printf("Enter the beginning of the interval: ");

double beginningX = getNumber();

printf("Enter the end of the interval: ");

double finishX = getNumber();

checkingInterval(beginningX, finishX);

printf("Enter the step value: ");

double stepValue = getNumber();

isPositive(stepValue);

double x = beginningX;

while(fabs(x) - fabs(finishX) > - DBL\_EPSILON)

{

if (existFunction(x))

{

double y = getFunction(x);

printf(" x = %lf", x);

printf(" y = %lf\n", y);

}

else

{

printf("The function does not exist, when x = %lf\n", x);

}

x = x + stepValue;

}

return 0;

}

double getNumber()

{

double entered\_number;

if (scanf("%lf", &entered\_number) != 1 )

{

printf("Wrong value");

abort();

}

return entered\_number;

}

void checkingInterval(double beginningX, double finishX)

{

if(beginningX - finishX < DBL\_EPSILON)

{

printf("Does not satisfy the interval condition!");

abort;

}

}

void isPositive(double stepValue)

{

if(stepValue <= DBL\_EPSILON){

puts("Wrong value");

abort;

}

}

bool existFunction(double x)

{

return x > - DBL\_EPSILON;

}

double getFunction(double x)

{

return( (0.29 \* ( pow(x, 3))) + x - 1.2502 );

}

* 1. РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Результаты выполнения программы представлены ниже (Рисунок 6-7).

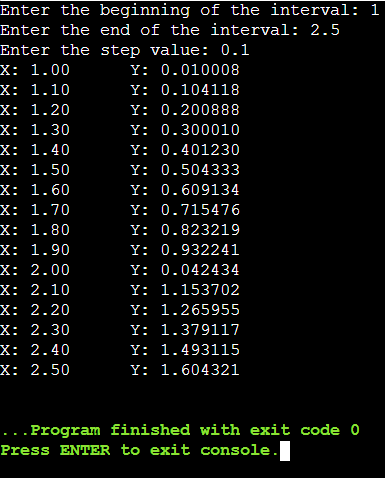
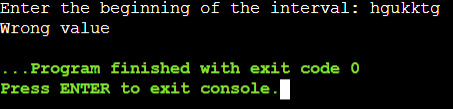
  
Рисунок 6 – Результаты выполнения программы  


Рисунок 7 – Результаты выполнения программы

* 1. ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 8, Рисунок 9).

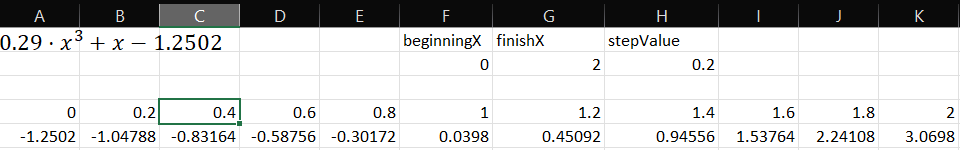


Рисунок 8 – Результат расчета значений функции

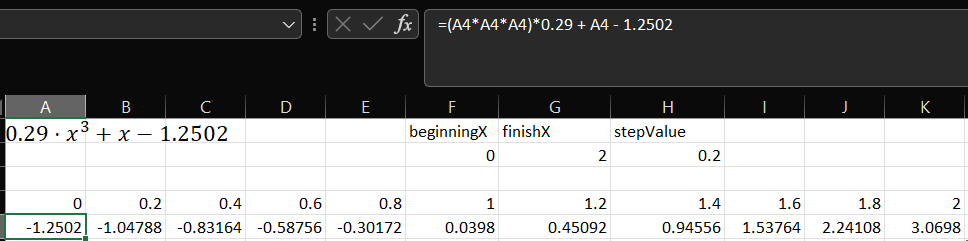


Рисунок 9 – Результат расчета значений функции

* 1. ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ



Рисунок – 10 Отметка о выполнении задания в веб-хостинге системы контроля версий

1. РЕШЕНИЕ ЗАДАЧИ 3.1 доп

2.1 ФОРМУЛИРОВКА ЗАДАНИЯ

При решении данных задач не забывайте дополнять свой отчёт пояснениями, почему выбран тот или иной метод решения. (Таблица 2)

Таблица 2 – Исходные данные



2.2 БЛОК-СХЕМА АЛГОРИТМА

Блок-схема основного алгоритма представлена ниже (Рисунок 11).

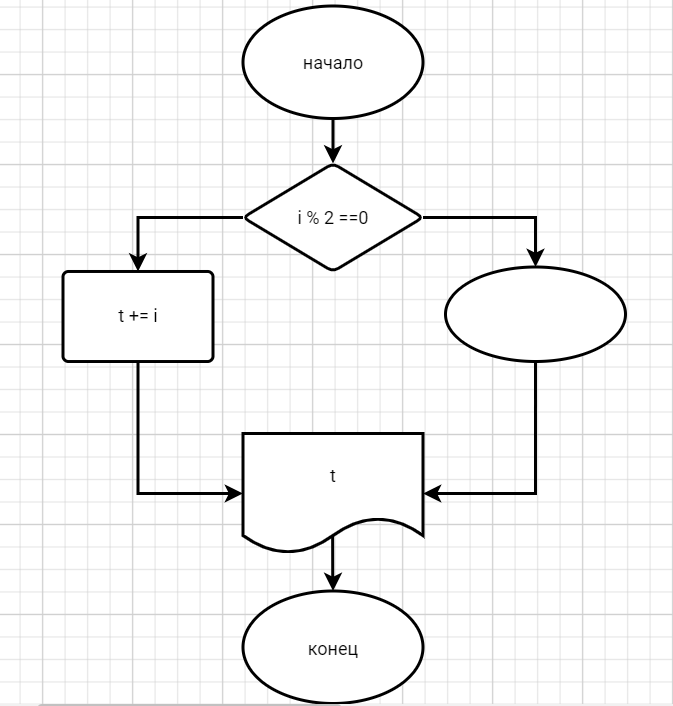


Рисунок 11 ­ Блок-схема основного алгоритма

2.3 Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

/\*\*

@brief считает сумму всех четных чисел от 1 до 100

@return возвращает посчитанную сумму

\*/

int sumEvenNumbers();

/\*\*

@brief точка входа в программу

@return возвращает 0, если программа выполнена верно

\*/

int main()

{

int t = 0;

for(int i = 0; i <= 100; i++){

if(i % 2 == 0){

t += i;

}

}

printf("%d", t);

return 0;

}

2.4 Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 12).

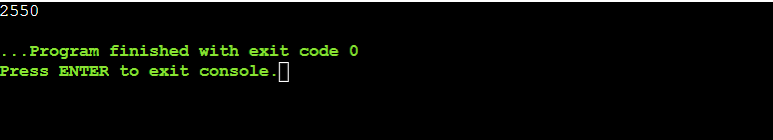


Рисунок 12– Результаты выполнения программы

2.5 Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 13).

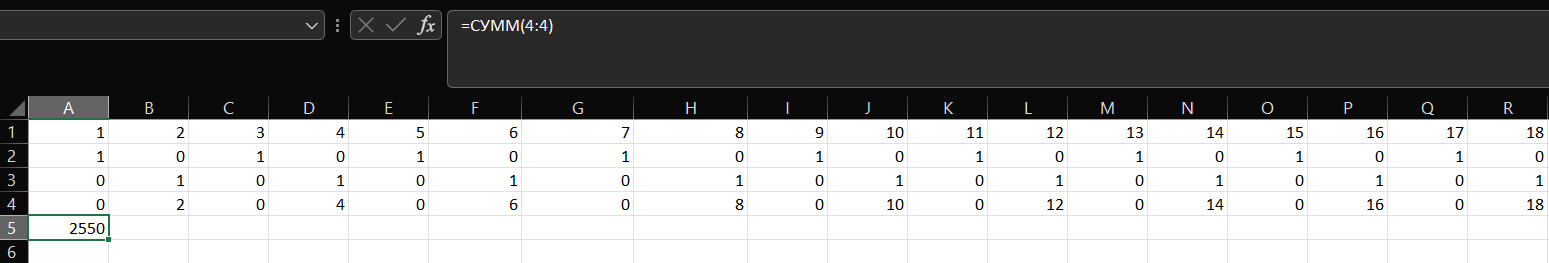


Рисунок 13 – Результат выполнения программы

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



Рисунок 14 – Результат выполнения задания в веб хостинге контроля версий

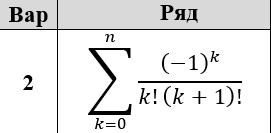
РЕШЕНИЕ ЗАДАЧИ 3.2  
3.1 Формулировка задания

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.

Таблица 3 – Исходные данные



3.2 Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 15). Блок-схемы используемых функций представлены ниже (Рисунок 16).

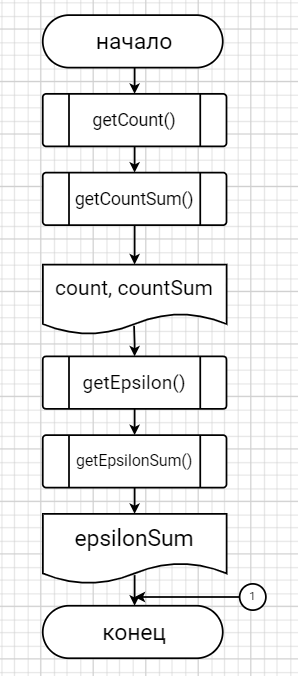


Рисунок 5 ­ Блок-схема основного алгоритма

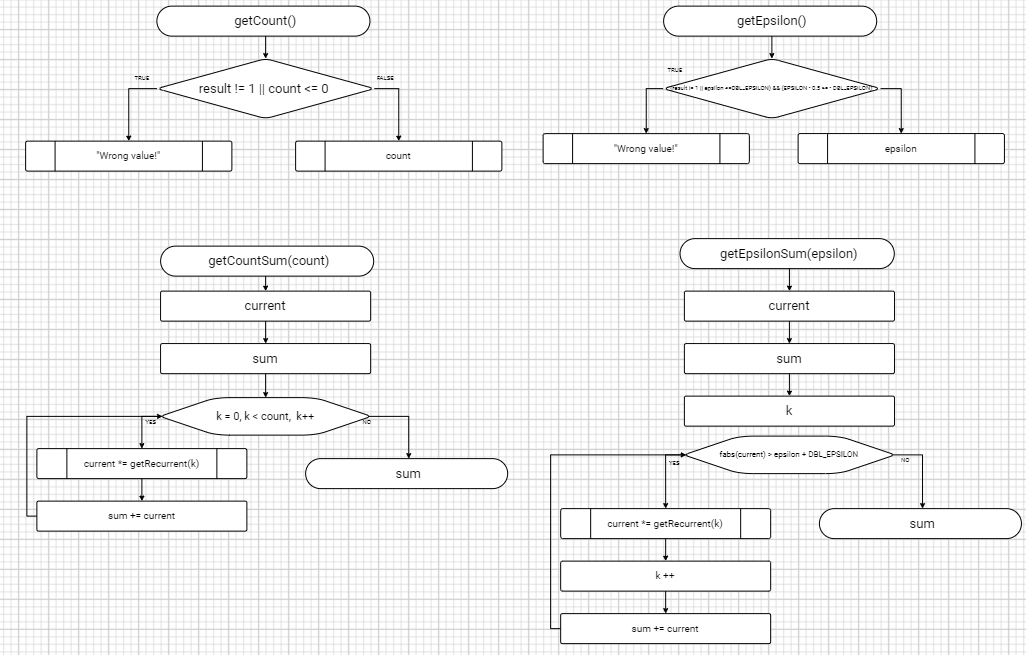


Рисунок 16 – Блок-схема используемых функций

3.3 Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <float.h>

#include <errno.h>

#include <locale.h>

/\*\*

\* @brief проверяет на положительное число

\* @remarks Экстренное завершение программы, в случае неправильного ввода

\* @return возвращает Количество членов последовательности

\*/

int getCount();

/\*\*

\* @brief проверяет на функцию, считывающую точность вычисления суммы последовательности

\* @remarks Экстренное завершение программы, в случае неправильного ввода

\* @return возвращает точность вычисления суммы последовательности

\*/

double getEpsilon();

/\*\*

\* @brief рассчитывает функцию сумму членов последовательности

\* @param count - количество членов последовательности

\* @return возвращает сумму последовательности

\*/

double getCountSum(int count);

/\*\*

\* @brief рассчитывает сумму членов последовательности

\* @param epsilon - точность вычисления суммы последовательности

\* @return возвращает посчитанную сумму последовательности

\*/

double getEpsilonSum(double epsilon);

/\*\*

\* @brief расчитывает рекурентный член последовательности

\* @param k - номер члена последовательности

\* @return возвращает значение рекурентного члена последовательности

\*/

double getRecurrent(int k);

/\*\*

\*@brief Точка входа в программу

\*@return Возврящает 0, если программа работает верно, иначе 1

\*/

int main()

{

setlocale(LC\_ALL, "RU");

printf("Введите количество членов последовательности: ");

int count = getCount();

double countSum = getCountSum(count);

printf("Суммы %d членов последовательности равны: %.20lf \n", count, countSum);

printf("Введите точность вычисления суммы последовательности: ");

double epsilon = getEpsilon();

double epsilonSum = getEpsilonSum(epsilon);

printf("Сумма последовательности с заданной точностью равна: %.20lf", epsilonSum);

return EXIT\_SUCCESS;

}

int getCount()

{

int count;

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

if(result != 1 || count <= 0)

{

errno = EIO;

perror("Wrong value");

abort;

}

return count;

}

double getEpsilon()

{

double epsilon;

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

if((result !=1 || epsilon <= DBL\_EPSILON) && (epsilon - 0.5 >= -DBL\_EPSILON))

{

errno = EIO;

perror("Wrong value");

abort;

}

return epsilon;

}

double getCountSum(int count)

{

double current = 1.0; //

double sum = 0; // \*\*\*

for(int k = 0; k < count + 1; k++)

{

current \*= getRecurrent(k);

sum += current;

}

return sum;

}

double getEpsilonSum(double epsilon)

{

double current = 1.0; ///

double sum = 0;

int k = 0;

while (fabs(current) > epsilon + DBL\_EPSILON) /// &&&&

{

current \*= getRecurrent(k);

k++;

sum += current;

}

return sum;

}

double getRecurrent(int k)

{

return((-1)/((k+2)\*(k+1)));

}

3.4 Результаты выполнения программы

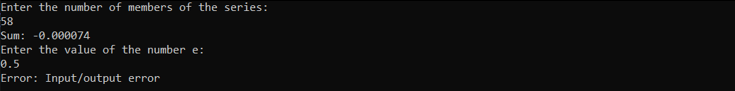
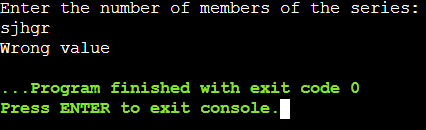
  
Рисунок 17 - Результаты выполнения программы  


Рисунок 18 - Результаты выполнения программы

3.5 Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 19 – Рисунок 24).

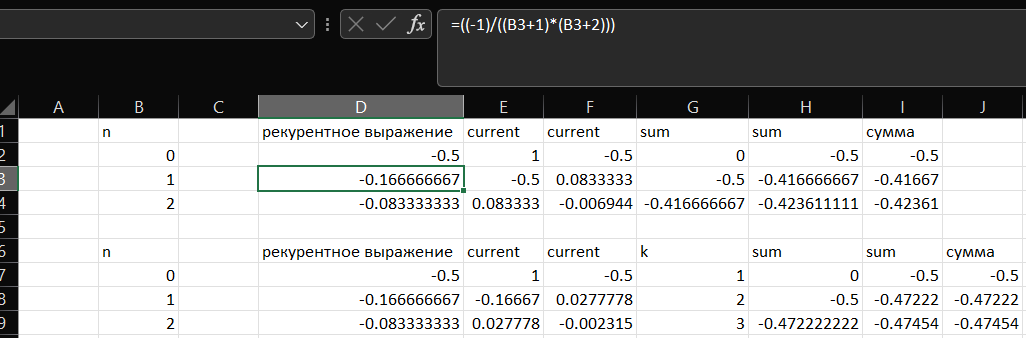
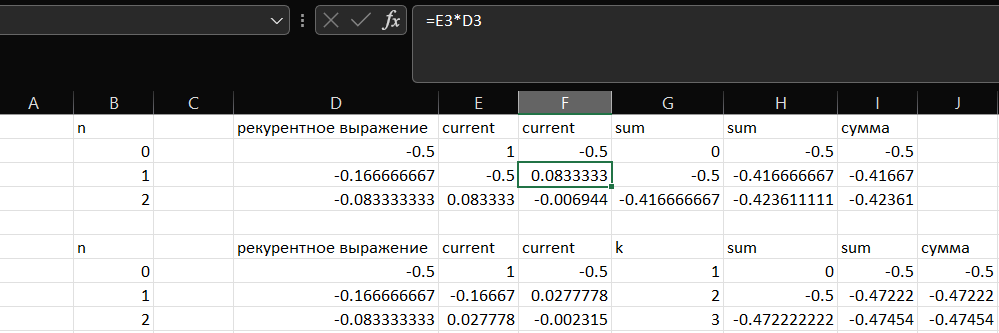
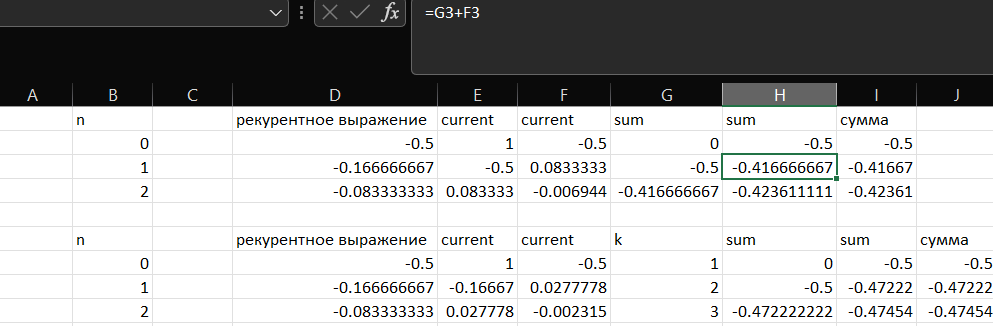
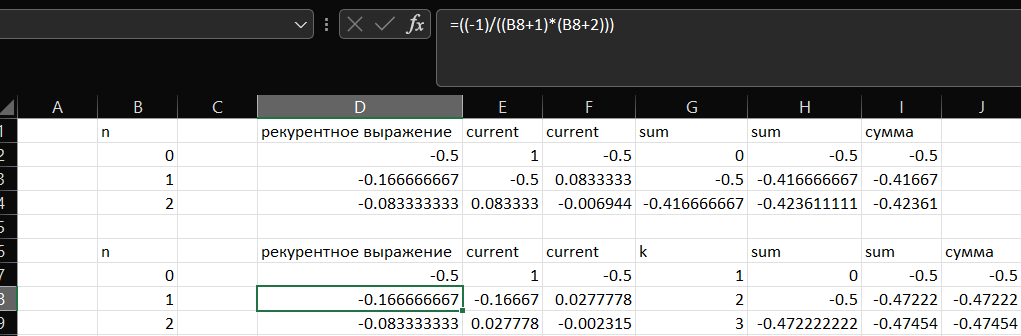
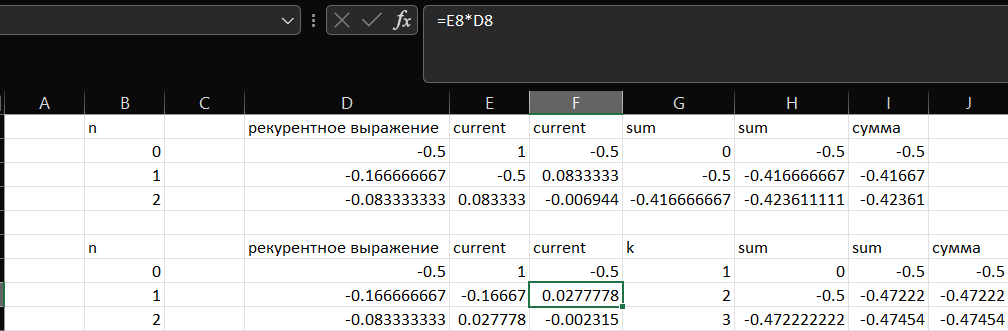


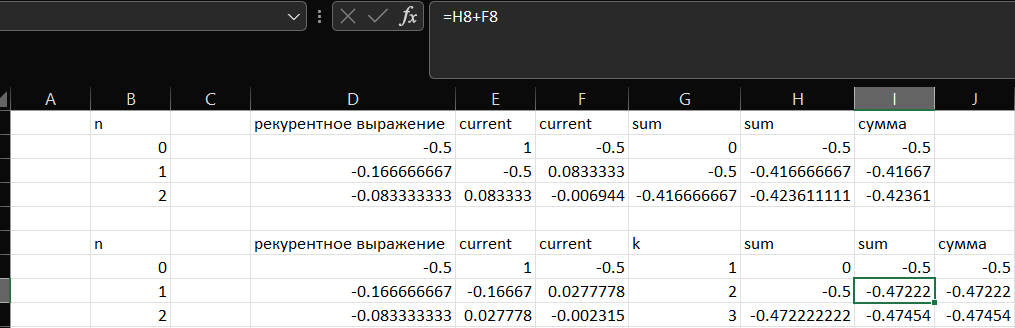
Рисунок 19 – Результат расчета функции

****Рисунок 20 – Результат расчета функции

****Рисунок 21 – Результат расчета функции

****Рисунок 22 – Результат расчета функции

****Рисунок 23 – Результат расчета функции

****Рисунок 24 – Результат расчета функции

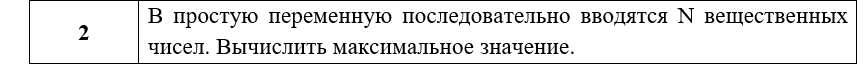
3.6 Отметка о выполнении задания в веб-хостинге системы контроля версий

Рисунок 25 – Результат выполнения задания в веб хостинге

РЕШЕНИЕ ЗАДАЧИ 3.2 доп  
4.1 Формулировка задания

При решении данных задач не забывайте дополнять свой отчёт пояснениями, почему выбран тот или иной метод решения.

Таблица 4 – Исходные данные



4.2 Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 26). Блок-схемы используемых функций представлены ниже (Рисунок 27).

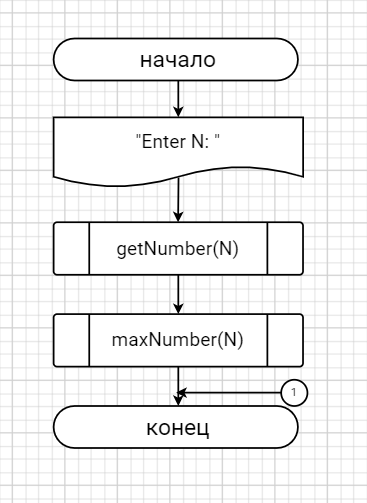


Рисунок 26 ­ Блок-схема основного алгоритма

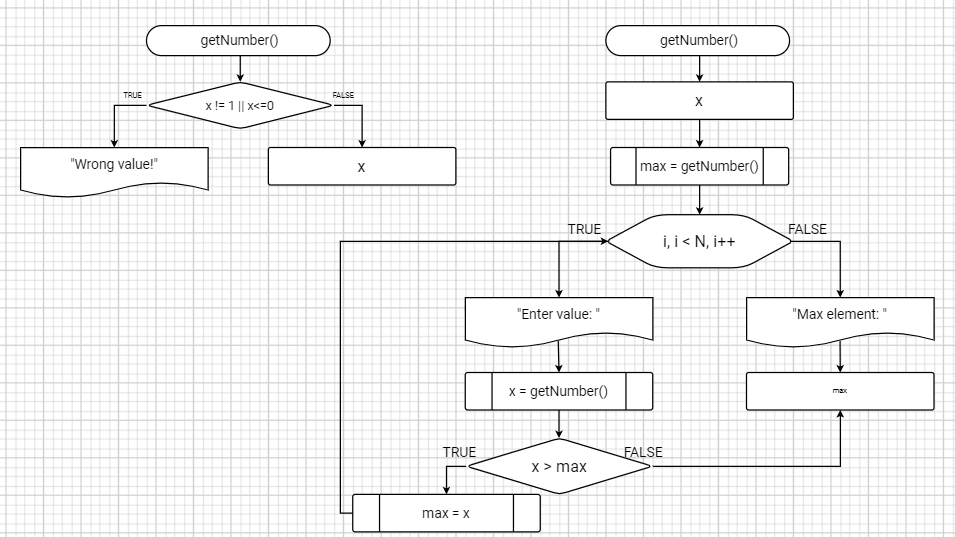
****

Рисунок 27 ­ Блок-схема функций

4.3 Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

/\*\*

\* @brief считывает и проверяет ввод переменной типа double

\* @return возвращает считанное значение

\*/

double getNumber();

/\*\*

@brief считает максимальное число из N введенных чисел

@return возвращает максимальное число из N введенных

\*/

double maxNumber(int N);

/\*\*

@brief точка входа в программу

@return возвращает 0, если программа выполнена верно

\*/

int main()

{

int N;

printf("Enter N: ");

getNumber(N);

maxNumber(N);

return 0;

}

double getNumber()

{

double x;

if (scanf("%lf", &x) !=1 || x <= 0)

{

puts("Wrong value");

abort();

}

return x;

}

double maxNumber(int N)

{

double x;

double max = getNumber();

for (int i = 1; i < N; i++)

{

printf("Enter value: ");

x = getNumber();

if (x > max)

{

max = x;

}

}

printf("Max element: %lf\n", max);

}

4.4 Результаты выполнения программы

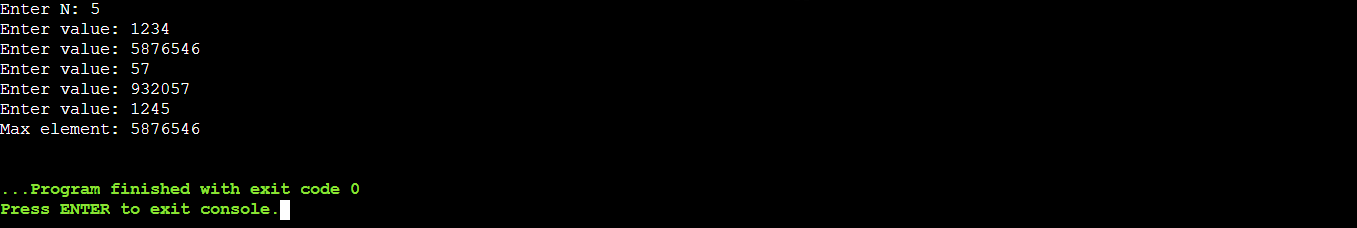


Рисунок 28 - Результаты выполнения программы

4.5 Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 29).

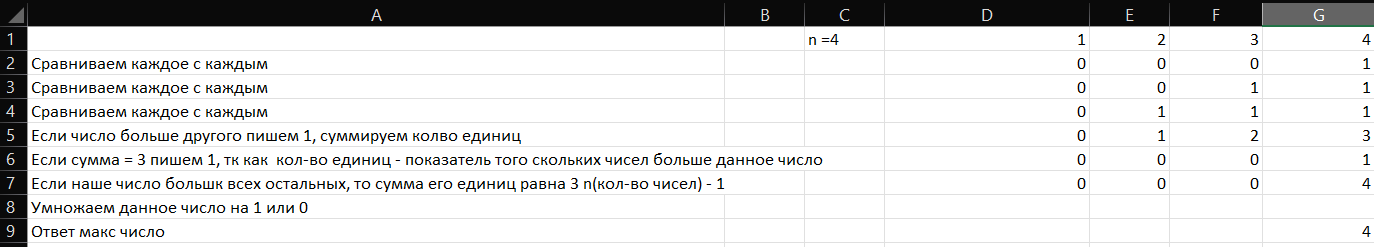


Рисунок 29 – Результат расчета функции

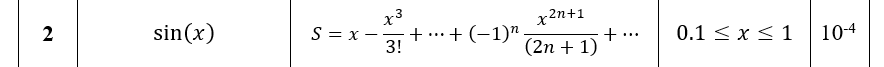
4.6 Отметка о выполнении задания в веб-хостинге системы контроля версий

Рисунок 30 – Результат выполнения задания в веб хостинге

РЕШЕНИЕ ЗАДАЧИ 3.3  
5.1 Формулировка задания

Протабулировать заданную функцию и сумму функционального ряда разложения этой функции на интервале [*a,b*] и с шагом *h* (шаг и интервал задается в константах). Функциональнй ряд вычисляется по соответствующей рекуррентной формуле с заданной точностью *ɛ*. В результате показать три столбца: значение аргумента, значение функции в данной точке и значение суммы ряда, вычисленное с заданной точностью в данной точке. Два последних столбца должны иметь близкие результаты.

Таблица 5 – Исходные данные



5.2 Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 31). Блок-схемы используемых функций представлены ниже (Рисунки 32-34).

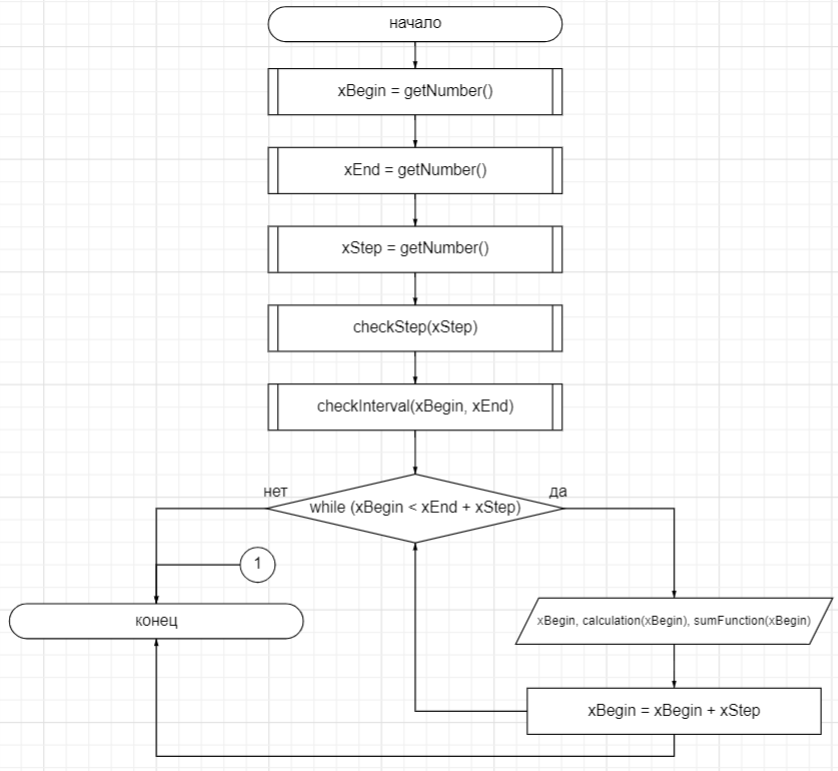


Рисунок 31 ­ Блок-схема основного алгоритма

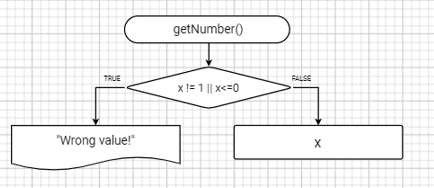
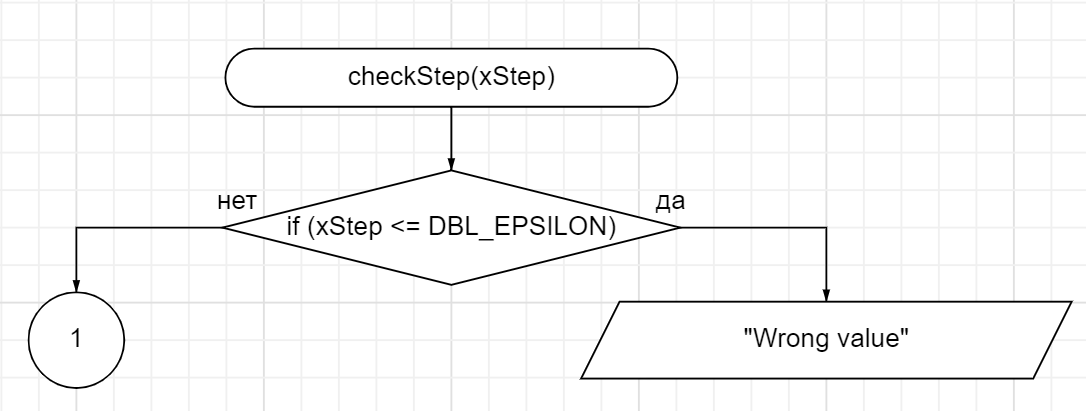
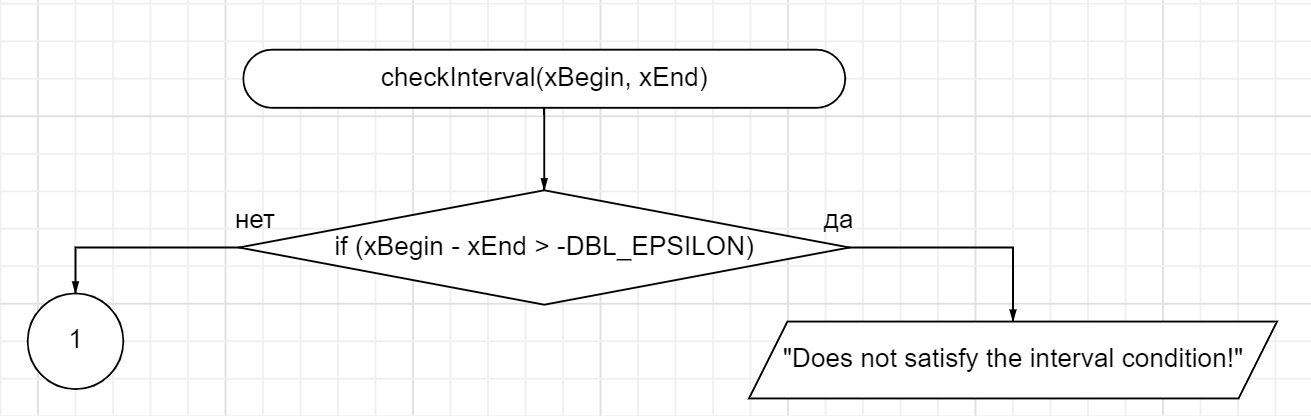
****

Рисунок 32 ­ Блок-схема функций

****Рисунок 33 ­ Блок-схема функций ****  
Рисунок 34 ­ Блок-схема функций

5.3 Текст программы на языке C

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <float.h>

#include <errno.h>

/\*\*

\* @brief Считывает и проверяет ввод переменной

\* @return Возвращает считанное значение

\*/

double getNumber();

/\*\*

\* @brief Проверяет, чтобы шаг был положительным

\* @return Возвращает 1, если условие верное

\*/

double checkStep(double xStep);

/\*\*

@brief Проверяет условие, чтобы начальное значение интервала было меньше конечного

@param xBegin - значение начала интервала

@param xEnd - значение конца интервала

@return Возвращает 1, если условие выполнено

\*/

double checkInterval(double xBegin, double xEnd);

/\*\*

\* @brief Рассчитывает заданную функцию в данной точке

\* @param x - аргумент функции

\* @return Возвращает значение функции в данной точке

\*/

double calculation(double x);

/\*\*

\* @brief Рассчитывает сумму функционального ряда

\* @param x - аргумент функции

\* @return Возвращает сумму функционального ряда

\*/

double sumFunction(double x);

/\*\*

\* @brief точка входа в программу

\* @return Возвращает 0, если программа работает верно

\*/

int main()

{

printf\_s("Enter the beginning value: \n");

double xBegin = getNumber();

printf\_s("Enter the end value: \n");

double xEnd = getNumber();

printf\_s("Enter the step value: \n");

double xStep = getNumber();

checkStep(xStep);

checkInterval(xBegin, xEnd);

while (xBegin < xEnd + xStep) //

{

printf\_s("X: %lf Y: %lf Sum(x): %lf\n", xBegin, calculation(xBegin), sumFunction(xBegin));

xBegin = xBegin + xStep;

}

return 0;

}

double getNumber()

{

double entered\_number;

if (scanf\_s("%lf", &entered\_number) != 1)

{

errno = EIO;

perror("Wrong value");

abort();

}

return entered\_number;

}

double checkStep(double xStep)

{

if (xStep <= DBL\_EPSILON) //

{

errno = EIO;

perror("Wrong value");

abort();

}

return 1;

}

double checkInterval(double xBegin, double xEnd)

{

if (xBegin - xEnd > -DBL\_EPSILON) //

{

printf("Does not satisfy the interval condition!");

abort();

}

return 1;

}

double calculation(double x)

{

return (sin(x));

}

double sumFunction(double x)

{

const double e = pow(10, -4); // точность

double n = 0;

double current\_x = 1.0; //

double sum = x;

while (calculation(x) - sum > e - DBL\_EPSILON) //

{

current\_x \*= (((pow(x, 2)) \* (((-2) \* n) - 1)) / ((2 \* n) + 3));

sum += current\_x;

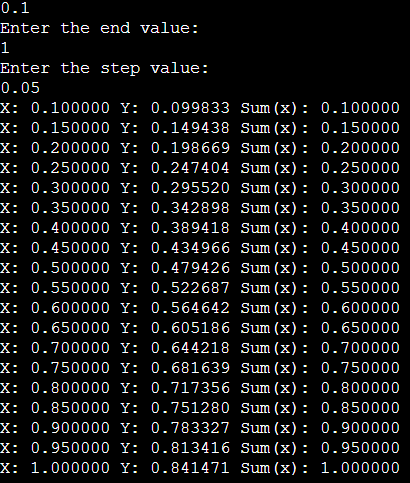
n = n + 1;

}

return sum;

}

5.4 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

  
Рисунок 35 – Результаты выполнения программы

5.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 36).

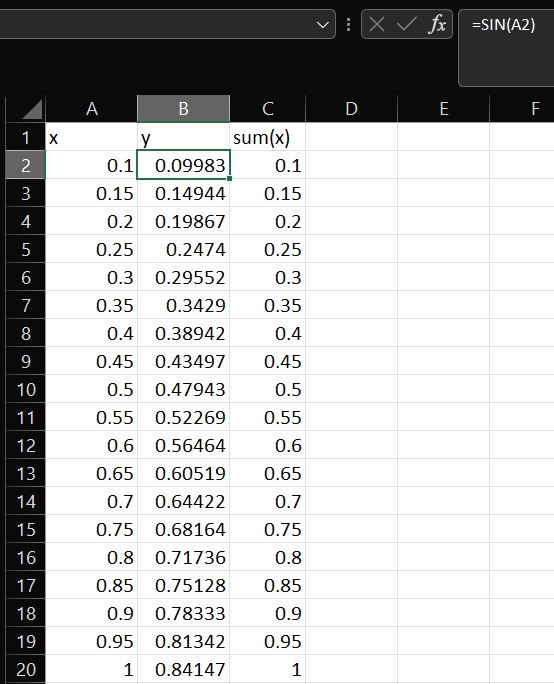


Рисунок 36 – Результат расчета функции

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

Рисунок 37 – Результат выполнения задания в веб хостинге