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(3-1, 3-2)

По дисциплине «Языки программирования»

ВАРИАНТ 1

Выполнил: ст. гр. ТКИ - 141

Абузина Елизавета Юрьевна

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

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

Москва 2024

ОГЛАВЛЕНИЕ

[1. ЗАДАНИЕ 3-1 3](#_Toc185682363)

[1.1 Формулировка задания 3](#_Toc185682364)

[1.2 Блок-схема алгоритма 4](#_Toc185682365)

[1.3 Текст программы на языке C 5](#_Toc185682366)

[1.4 Результаты выполнения программы 7](#_Toc185682367)

[1.5 Выполнение тестовых примеров 8](#_Toc185682368)

[1.6 Отметка о выполнении задания в веб-хостинге системы контроля версий 9](#_Toc185682369)

[2. ЗАДАНИЕ 3-2 10](#_Toc185682370)

[2.1 Формулировка задания 10](#_Toc185682371)

[2.2 Блок схема алгоритма 11](#_Toc185682372)

[2.3 Текст программы на языке C 13](#_Toc185682373)

[2.4 Результат выполнения программы 16](#_Toc185682374)

[2.5 Выполнение тестовых примеров 17](#_Toc185682375)

[2.6 Отметка о выполнении задания в веб-хостинге системы контроля версий 18](#_Toc185682376)

1. ЗАДАНИЕ 3-1
   1. Формулировка задания

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  – Исходные данные

Снимок экрана 2024-10-13 154809.png

* 1. Блок-схема алгоритма

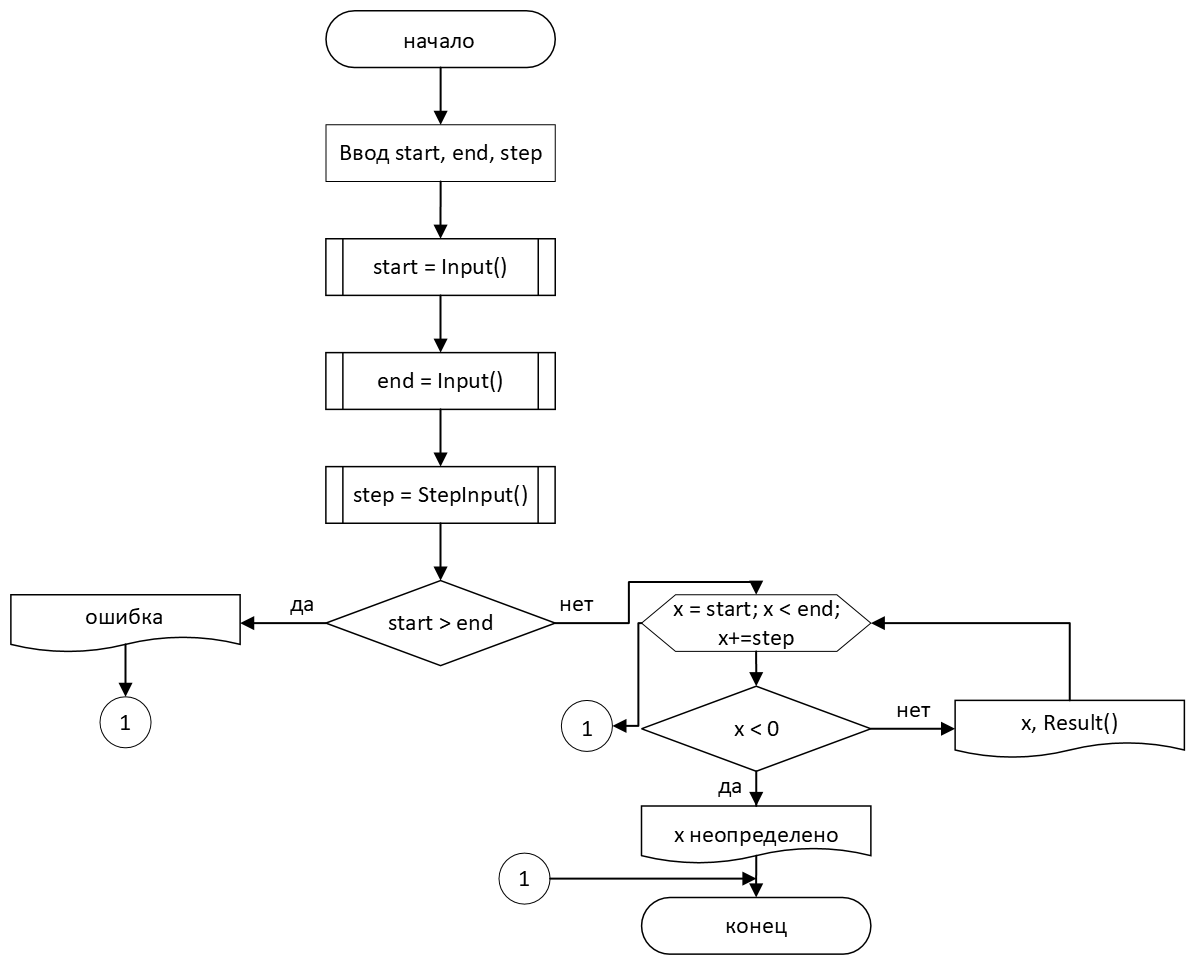


Рисунок  ­ Блок-схема основного алгоритма

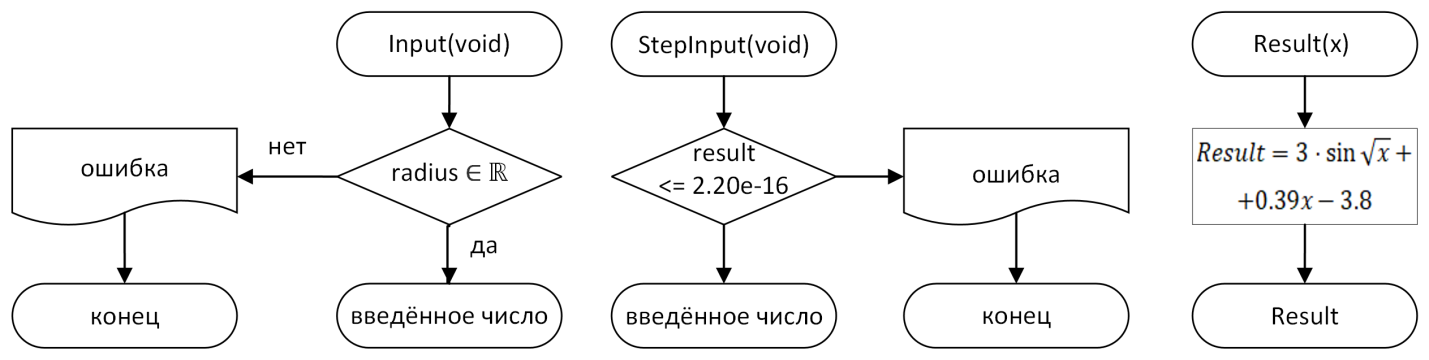


Рисунок  – Блок-схема используемых функций

* 1. Текст программы на языке C

#include <stdio.h>

#include <locale.h>

#include <math.h>

#include <errno.h>

#include <stdlib.h>

#include <float.h>

/\*\*

\* @brief Считывает вещественное число

\* @remarks При неправильном вводе программа завершает выполнение.

\* @return Вещественное число

\*/

double Input(void);

/\*\*

\* @brief Считывает вещественное число

\* @remarks При неправильном вводе программа завершает выполнение.

\* @return Вещественное число.

\*/

double StepInput(void);

/\*\*

\* @brief Рассчитывает значение переменной у.

\* @param х - значение переменной х.

\* @return Значение переменной у.

\*/

double Result(const double x);

/\*\*

\* @brief Точка входа в программу

\* @return 0, в случае успеха

\*/

int main(void)

{

setlocale(LC\_ALL, "Russian");

puts("Введите начало интервала:");

double start = Input();

puts("Введите конец интервала:");

double end = Input();

puts("Введите шаг:");

double step = StepInput();

if (start > end)

{

errno = EIO;

perror("Начало интервала не может быть больше конца");

exit(EXIT\_FAILURE);

}

for (double x = start; x < end + step; x += step)

{

if (x < DBL\_EPSILON)

printf("%lf Не определено\n", x);

else

printf("Результаты вычислений при x = %lf, y = %lf\n", x, Result(x));

}

return 0;

}

double Input(void)

{

double value = 0;

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

if (result != 1)

{

errno = EIO;

perror("Не удалось считать число");

exit(EXIT\_FAILURE);

}

return value;

}

double StepInput(void)

{

double result = Input();

if (result <= DBL\_EPSILON)

{

errno = EIO;

perror("Шаг не может быть отрицательным");

exit(EXIT\_FAILURE);

}

return result;

}

double Result(const double x)

{

return 3 \* sin(sqrt(x)) + 0.39 \* x - 3.8;

}

* 1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 3).

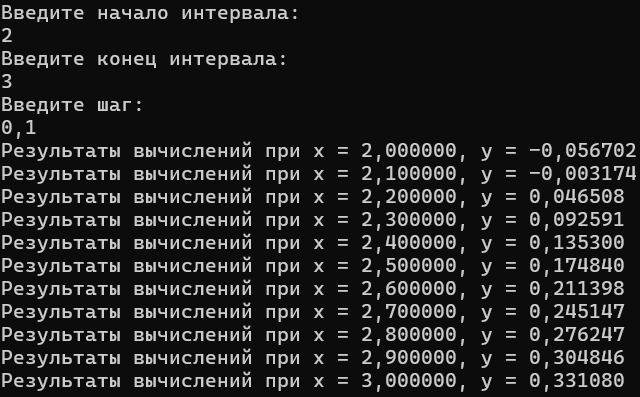


Рисунок  – Результаты выполнения программы

* 1. Выполнение тестовых примеров

В программе IDLE выполнен тестовый пример. Результаты его выполнения представлен ниже (Рисунок 4).

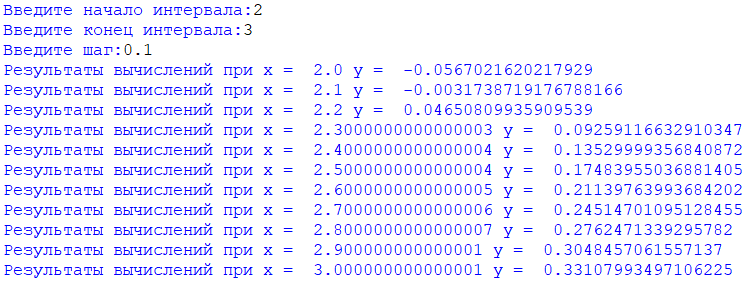
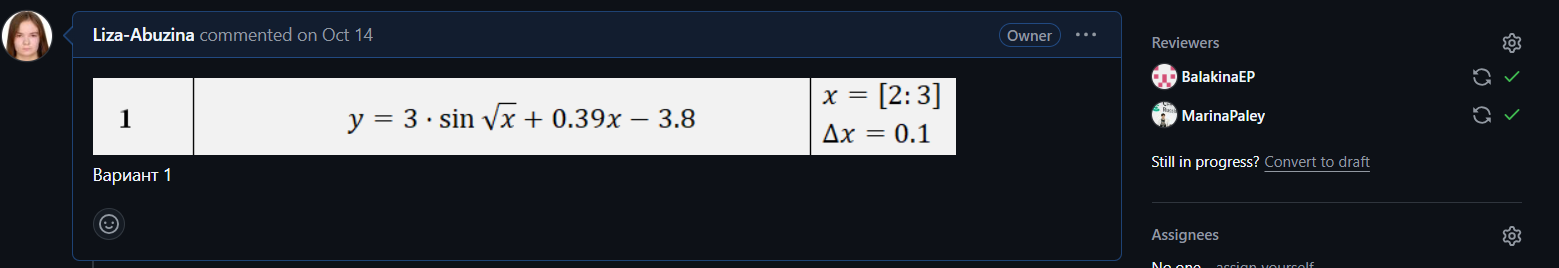


Рисунок  – Результат работы программы

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



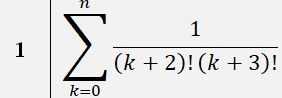
1. Задание 3-2
   1. Формулировка задания

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  – Исходные данные



* 1. Блок схема алгоритма

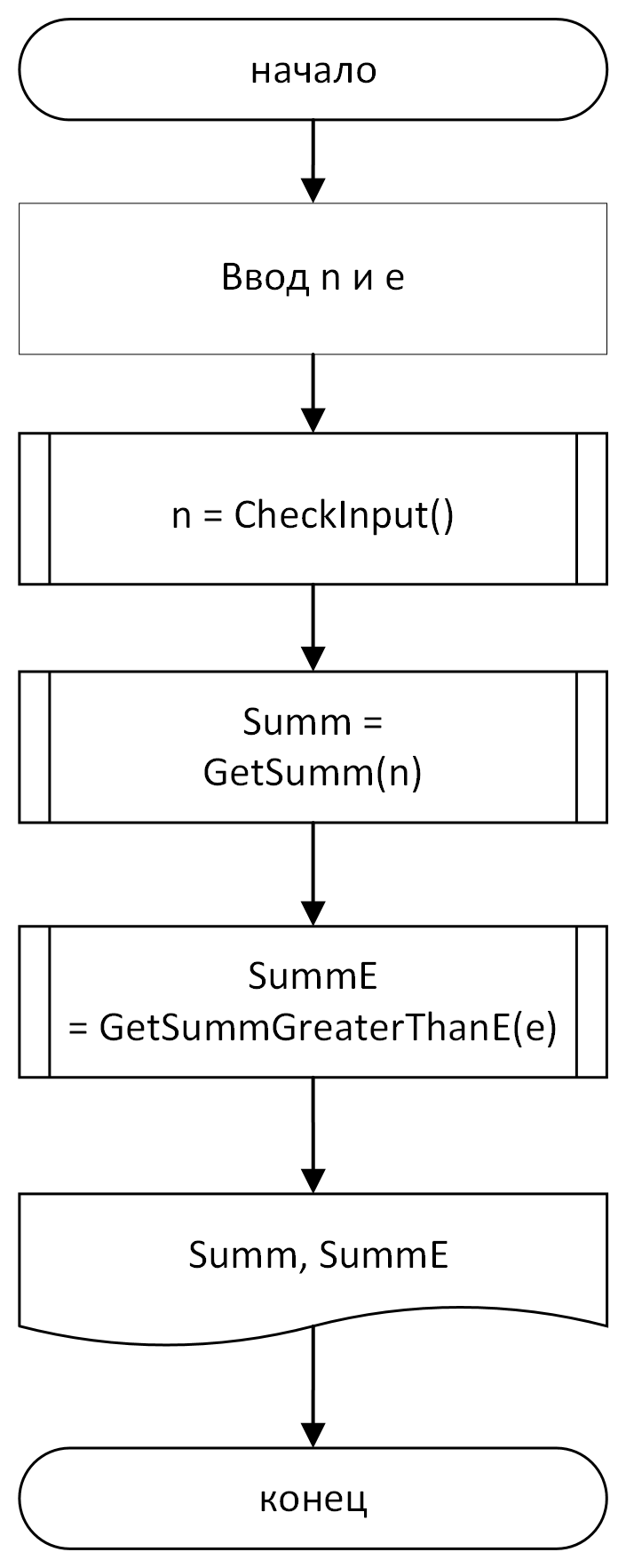


Рисунок  ­ Блок-схема основного алгоритма

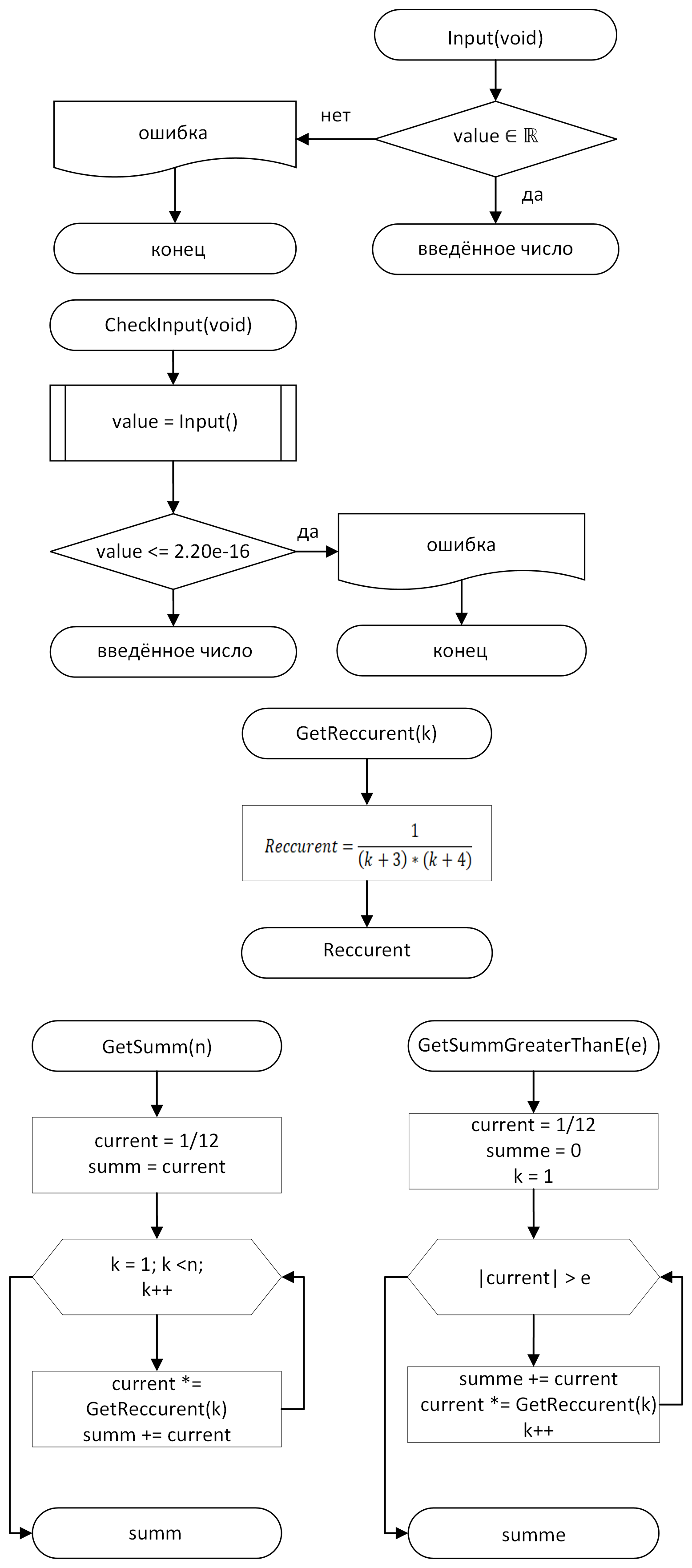


Рисунок  – Блок-схема используемых функций

* 1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <locale.h>

#include <float.h>

#include <math.h>

/\*\*

\* @brief Считывает вещественное число

\* @remarks При неправильном вводе программа завершает выполнение.

\* @return Вещественное число

\*/

double Input(void);

/\*\*

\* @brief Считывает вещественное число

\* @remarks При отрицательном значении ввода программа завершает выполнение

\* @return Вещественное число

\*/

double CheckInput(void);

/\*\*

\* @brief Получает на вход число k

\* @return Рекурентное число с заданным k

\*/

double GetReccurent(const double k);

/\*\*

\* @brief Считывает номер последнего члена последовательности

\* @return Сумма рекурентных чисел

\*/

double GetSumm(const double n);

/\*\*

\* @brief Вычисляет сумму членов последовательности, не меньших e

\* @param e Пороговое значение

\* @return Сумма членов последовательности, не меньших e

\*/

double GetSummGreaterThanE(const double e);

/\*\*

\* @brief Точка входа в программу

\* @return Код ошибки

\*/

int main(void)

{

setlocale(LC\_ALL, "Russian");

puts("Введите номер последнего члена последовательности");

double n = CheckInput();

printf("Сумма при n = %lf равна %.18lf\n", n, GetSumm(n));

puts("Введите пороговое число");

double e = CheckInput();

printf("Сумма членов последовательности, не меньших e = %lf равна %.18lf", e, GetSummGreaterThanE(e));

}

double Input(void)

{

double value;

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

if (result != 1)

{

errno = EIO;

perror("Не удалось считать число");

exit(EXIT\_FAILURE);

}

return value;

}

double CheckInput(void)

{

double value = Input();

if (value <= DBL\_EPSILON)

{

errno = EDOM;

printf("Ошибка ввода числового значения");

exit(EXIT\_FAILURE);

}

return value;

}

double GetSumm(double n)

{

double current = 1.0 / 12;

double summ = current;

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

{

current \*= GetReccurent(k);

summ += current;

}

return summ;

}

double GetReccurent(double k)

{

return 1/((k + 3) \* (k + 4));

}

double GetSummGreaterThanE(double e)

{

double current = 1.0 / 12;

double summ = 0.0;

double k = 1.0;

while (fabs(current) >= e)

{

summ += current;

current \*= GetReccurent(k);

k++;

}

return summ;

}

* 1. Результат выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 3,Рисунок 4, Рисунок 5, Рисунок 6).

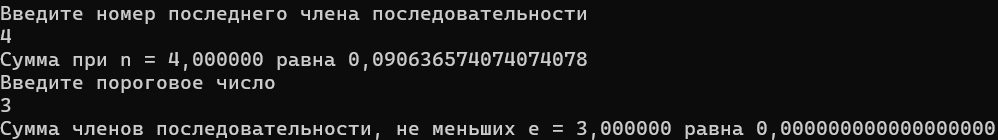


Рисунок  – Результаты выполнения программы

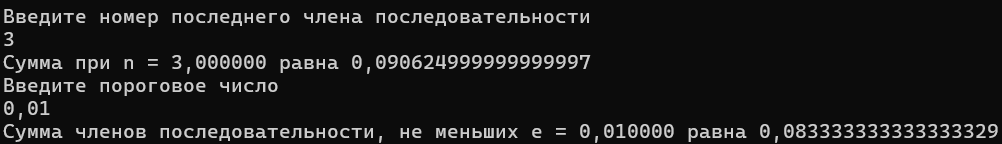


Рисунок 4 – Результаты выполнения программы при альтернативном вводе

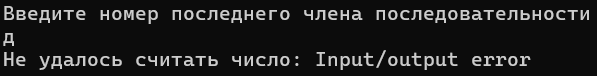


Рисунок 5 – Результаты выполнения программы при альтернативном вводе

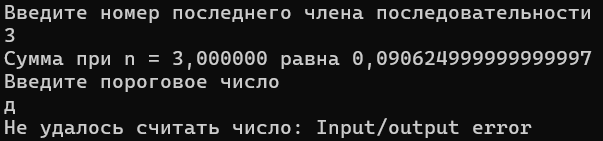


Рисунок 6 – Результаты выполнения программы при альтернативном вводе

* 1. Выполнение тестовых примеров

В программе IDLE выполнен тестовый пример. Результаты его выполнения представлен ниже (Рисунок 7, Рисунок 8, Рисунок 9, Рисунок 10).

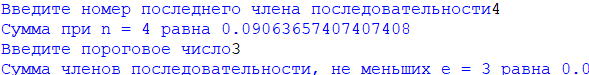


Рисунок 7 – Результат работы программы

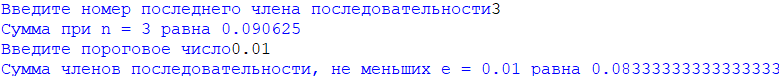


Рисунок 8 – Результат работы программы при альтернативном вводе

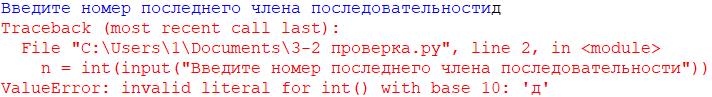


Рисунок 9 – Результат работы программы при альтернативном вводе

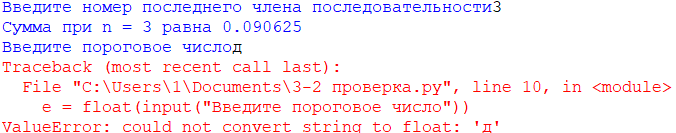


Рисунок 10 – Результат работы программы при альтернативном вводе

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

