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 9

Выполнил: ст. гр. ТКИ-141

Ткачев Вадим Евгеньевич

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

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

Москва 2024

Оглавление

[1 Решение задачи 3–1 3](#_Toc182520479)

[1.1 Формулировка задачи 3–1 3](#_Toc182520480)

[1.2 Блок-схема алгоритма задание 3–1 4](#_Toc182520481)

[1.3 Текст программы на языке С задание 3–1 7](#_Toc182520482)

[1.4 Результаты выполнения программы 3–1 9](#_Toc182520487)

[1.5 Выполнение тестовых примеров задание 3–1 9](#_Toc182520488)

[1.6 Отметка о выполнении задания в веб-хостинге системы контроля версий задание 3–1 10](#_Toc182520489)

[2 Решение задачи 3–2 12](#_Toc182520490)

[2.1 Формулировка задачи 3–2 12](#_Toc182520491)

[2.2 Блок-схема алгоритма задание 3–2 13](#_Toc182520492)

[2.3 Текст программы на языке С задание 3–2 20](#_Toc182520493)

[2.4 Результаты выполнения программы 3–2 23](#_Toc182520494)

[2.5 Выполнение тестовых примеров задание 3–2 23](#_Toc182520495)

[2.6 Отметка о выполнении задания в веб-хостинге системы контроля версий задание 3–2 24](#_Toc182520496)

[3 Решение задачи 3–3 26](#_Toc182520497)

[3.1 Формулировка задачи 2–3 26](#_Toc182520498)

[3.2 Блок-схема алгоритма задание 3–3 27](#_Toc182520499)

[3.3 Текст программы на языке С задание 3–3 32](#_Toc182520500)

[3.4 Результаты выполнения программы 3–3 34](#_Toc182520501)

[3.5 Выполнение тестовых примеров задание 3–3 35](#_Toc182520502)

[3.6 Отметка о выполнении задания в веб-хостинге системы контроля версий задание 3–3 36](#_Toc182520503)

1. Решение задачи 3–1
   1. Формулировка задачи 3–1

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.

Таблица  – Исходные данные

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

|  |  |  |
| --- | --- | --- |
| 9 |  |  |

* 1. Блок-схема алгоритма задание 3–1

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций get\_equation (Рисунок 2), get\_check\_x (Рисунок 3), input (Рисунок 4), check\_interval (Рисунок 5), check\_step (Рисунок 6), представлены ниже.

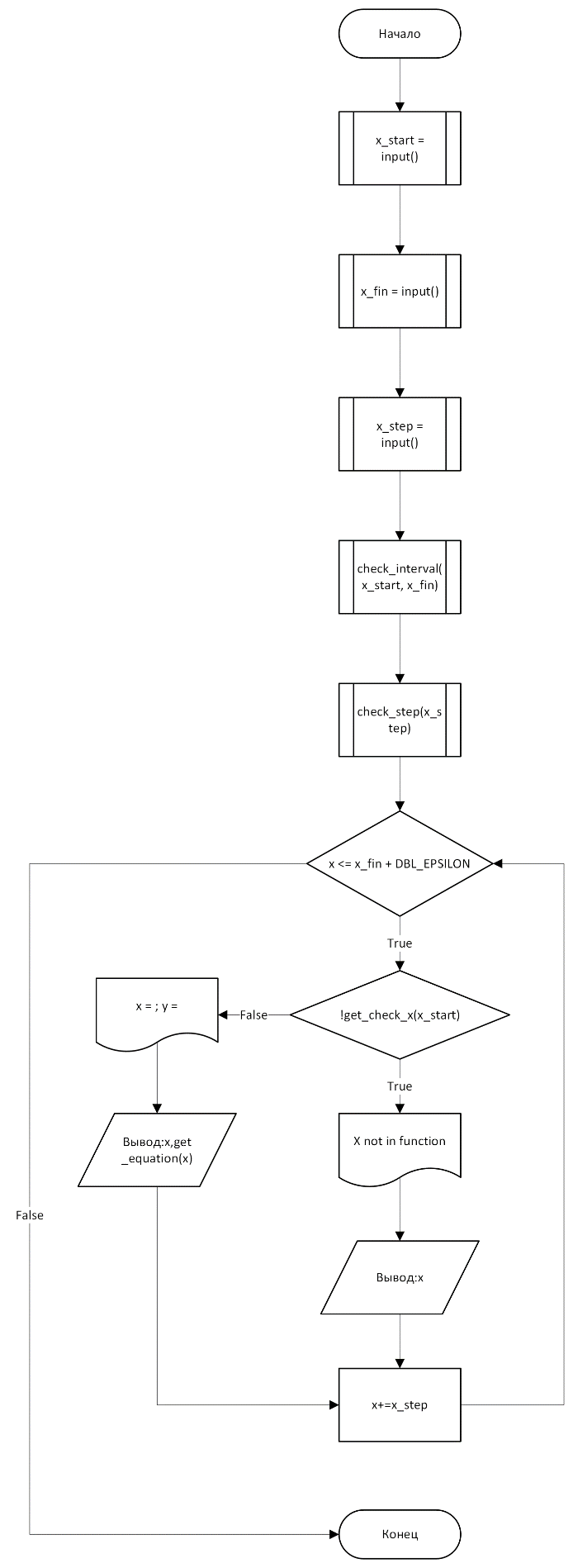


Рисунок  –­ Блок-схема основного алгоритма

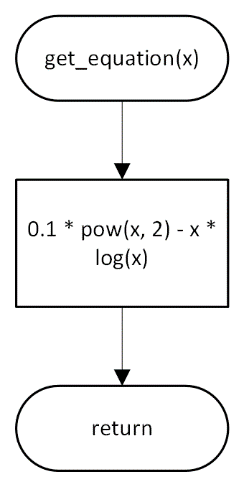


Рисунок  – Блок-схема функции счета

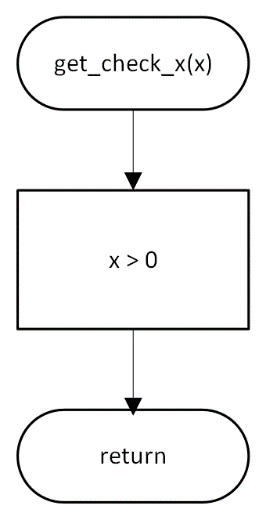


Рисунок – Блок-схема проверки X

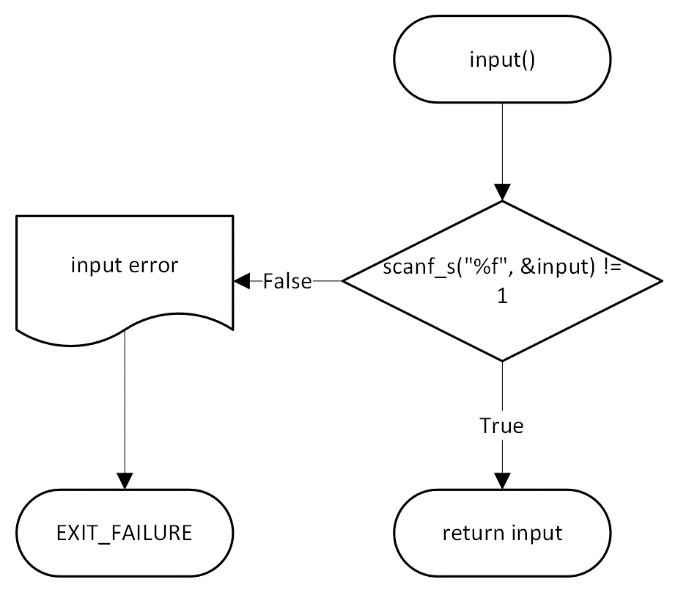


Рисунок – Блок-схема проверки ввода

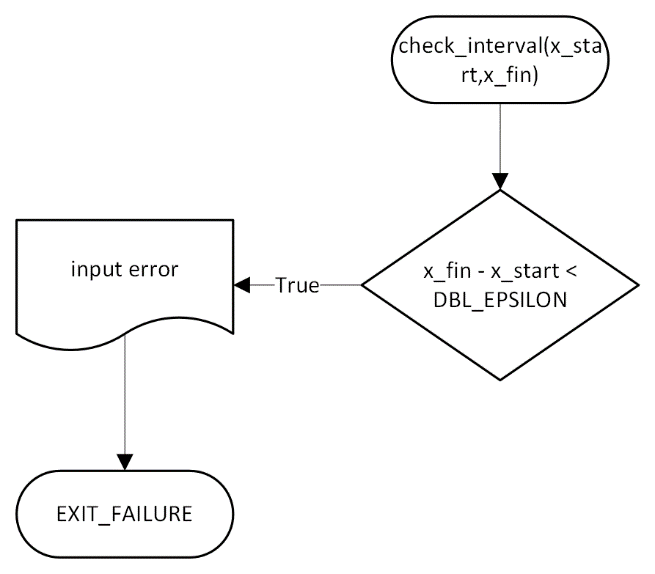


Рисунок – Блок-схема функции проверки интервала

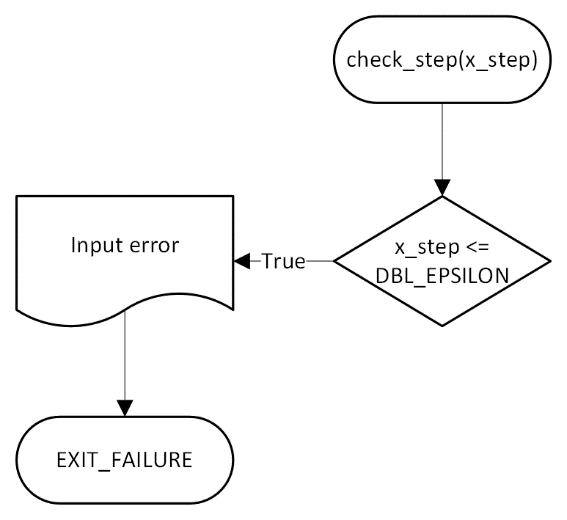


Рисунок – Блок-схема функции проверки шага

* 1. Текст программы на языке С задание 3–1

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <float.h>

#include <stdbool.h>

/\*\*

\* @brief Функция считает нашу функцию

\* @param x - значение аргумента

\* @return Возвращает значения функции

\*/

float get\_equation(float x);

/\*\*

\* @brief Проверяет, можно ли вычислить функцию для заданного значения x.

\* @param x Значение x, которое требуется проверить.

\* @return true, если функцию можно вычислить для данного значения x, false в противном случае.

\*/

bool get\_check\_x(float x);

/\*\*

\* @brief Проверка на введенное значение

\* @return возрващает значение, если верное, иначе ошибку

\*/

float input(void);

/\*\*

\* @brief Проверяет корректность интервала.

\* @param x\_start Начальное значение интервала.

\* @param x\_fin Конечное значение интервала.

\* @return Возвращет ошибку если интервао задан неверно

\*/

void check\_interval(const float x\_start, const float x\_fin);

/\*\*

\* @brief Проверяет коррекность шага

\* @param x\_step Значение шага

\* @return Возвращает ошибку если шаг задан неверно

\*/

void check\_step(const float x\_step);

/\*\*

\* @brief Точка входа

\* @return Возвращает значение функции с заданным в цикле корнем

\*/

int main(void)

{

const float x\_start = input(), x\_fin = input(), x\_step = input();

check\_interval(x\_start, x\_fin);

check\_step(x\_step);

float x = x\_start;

while (x <= x\_fin + DBL\_EPSILON) {

if (!get\_check\_x(x\_start)) {

printf("X not in function %.2f\n", x);

}

else {

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

}

x += x\_step;

}

return 0;

}

bool get\_check\_x(float x)

{

return x > 0;

}

void check\_interval(const float x\_start, const float x\_fin)

{

if (x\_fin - x\_start < DBL\_EPSILON)

{

puts("Input error");

exit(EXIT\_FAILURE);

}

}

void check\_step(const float x\_step)

{

if (x\_step <= DBL\_EPSILON)

{

puts("Input error");

exit(EXIT\_FAILURE);

}

}

float input(void) {

float input;

if (scanf\_s("%f", &input) != 1)

{

puts("Input error");

exit(EXIT\_FAILURE);

}

return input;

}

float get\_equation(float x)

{

return 0.1 \* pow(x, 2) - x \* log(x);

}

1. 4. Результаты выполнения программы 3–1

Результаты выполнения программы в C представлена ниже (Рисунок 7).

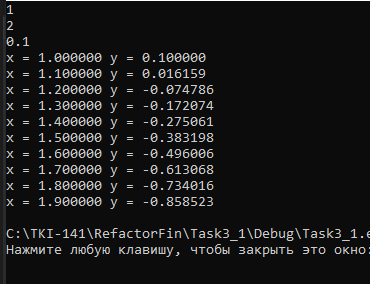


Рисунок – Результаты выполнения программы

* 1. Выполнение тестовых примеров задание 3–1

В программе Excelвыполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 8).

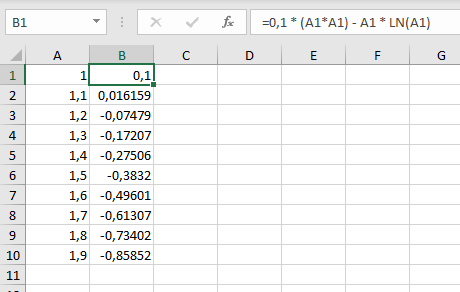


Рисунок – Результат выполнения тестового примера

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий задание 3–1

Ниже представлено доказательство того, что задание 3–1 было принято. (Рисунок 9)

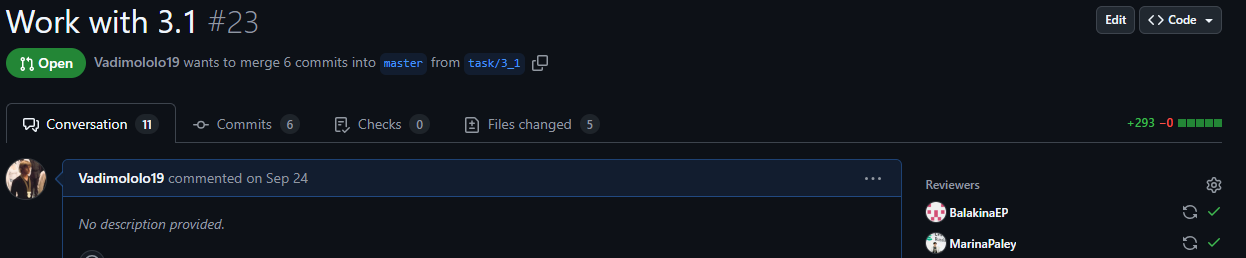


Рисунок – Approve задачи 3–1

1. Решение задачи 3–2
   1. Формулировка задачи 3–2

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

|  |  |
| --- | --- |
| Вариант | Ряд |
| 9 |  |

* 1. Блок-схема алгоритма задание 3–2

Блок-схема основного алгоритма представлена ниже (Рисунок 10). Блок-схемы функций get\_current (Рисунок 11), get\_sum (Рисунок 12), get\_count (Рисунок 13), check\_interval (Рисунок 14), check\_zero\_value (Рисунок 15), check\_e (Рисунок 16), get\_sum\_n (Рисунок 17), get\_count\_e (Рисунок 18), check\_zero\_value\_e (Рисунок 19), представлены ниже.

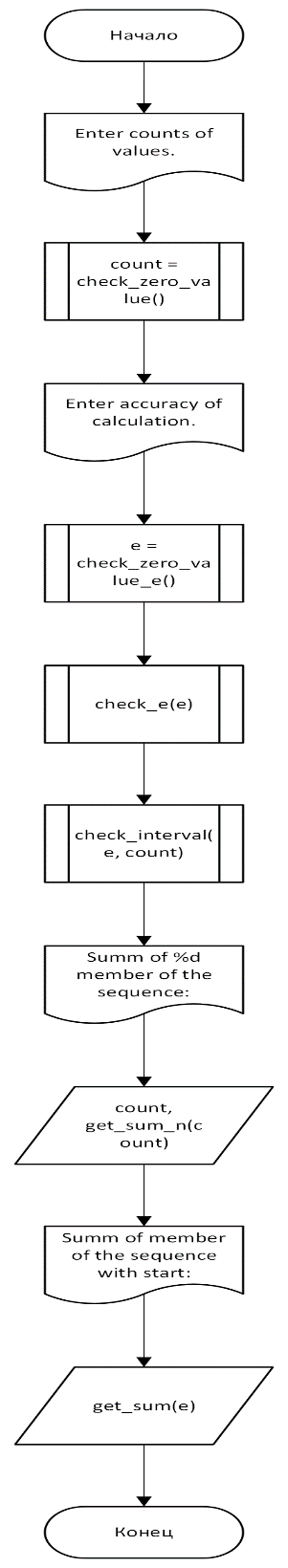


Рисунок  – Блок-схема основного алгоритма

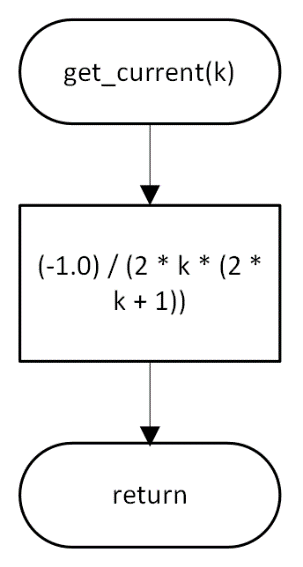


Рисунок – Блок-схема функции get\_current

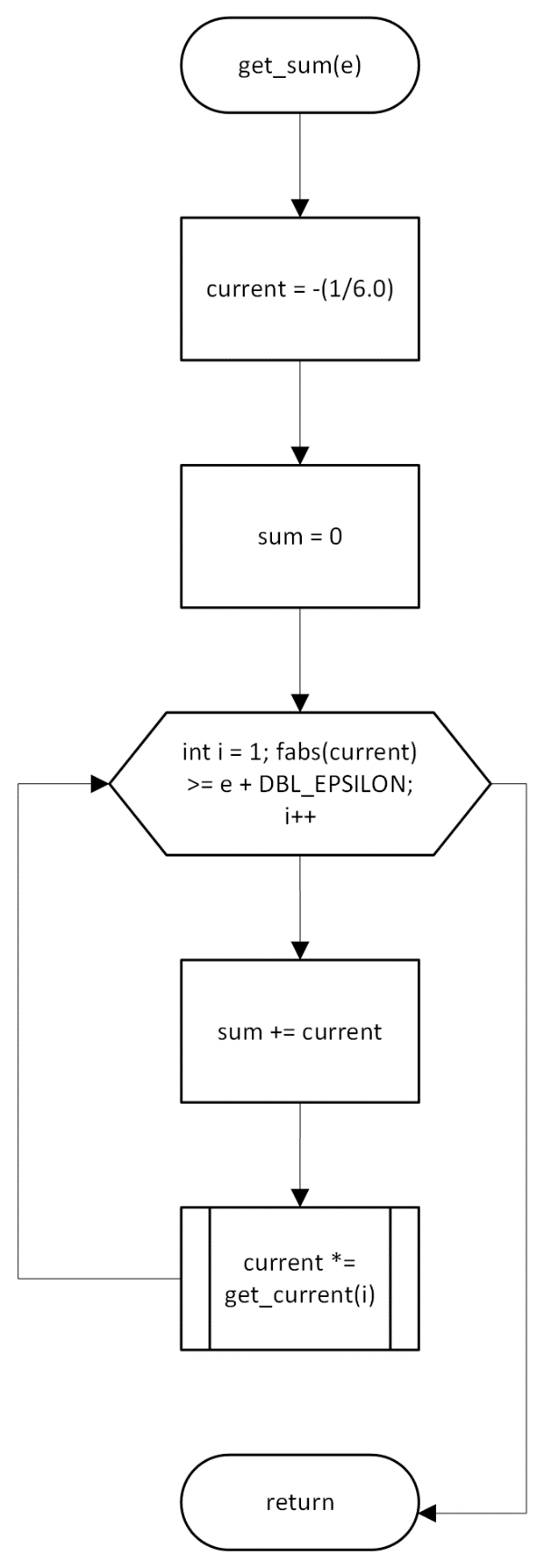


Рисунок – Блок-схема функции get\_sum

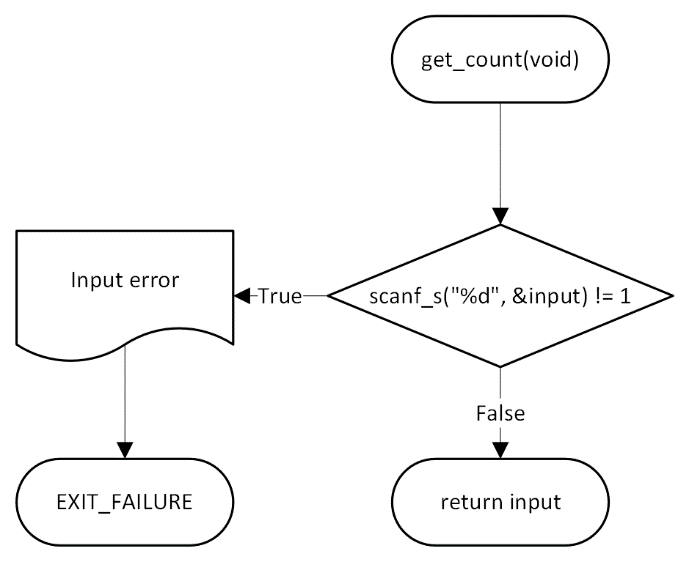


Рисунок – Блок-схема функции get\_count

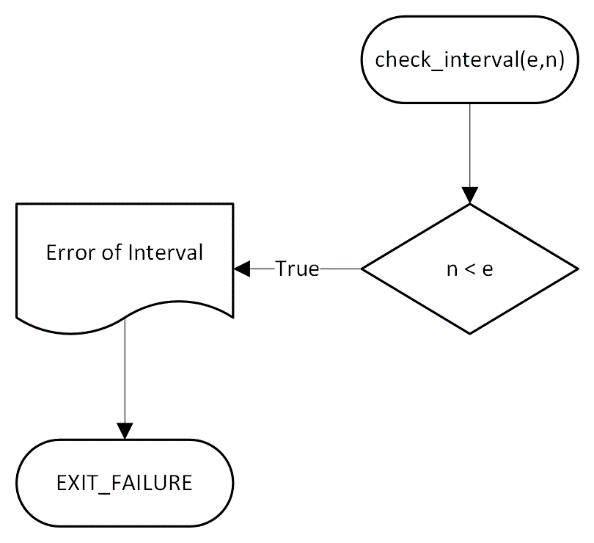


Рисунок – Блок-схема функции check\_interval

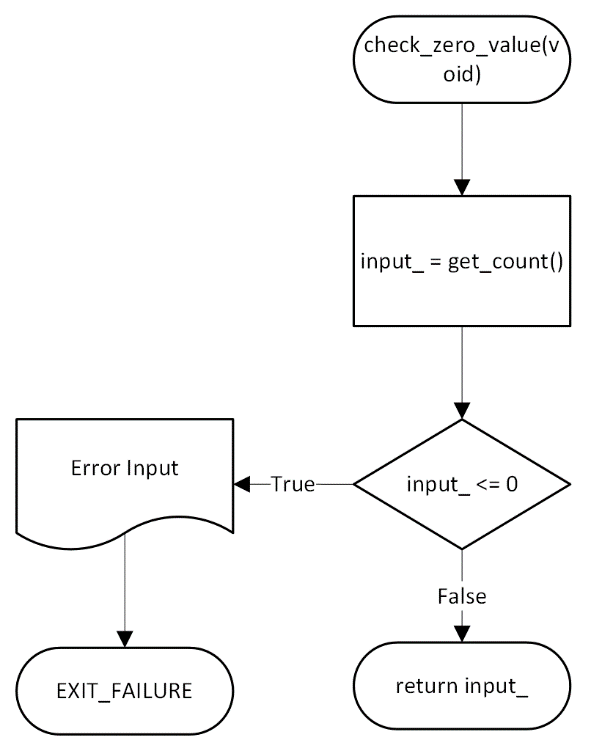


Рисунок – Блок-схема функции check\_zero\_value

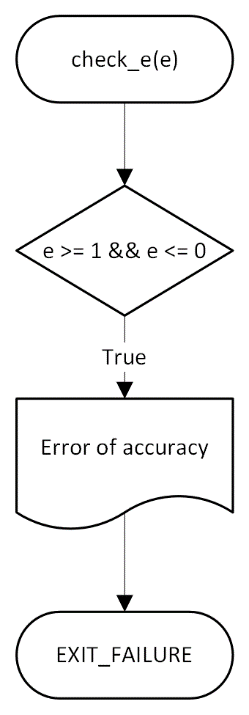


Рисунок – Блок-схема функции check\_e

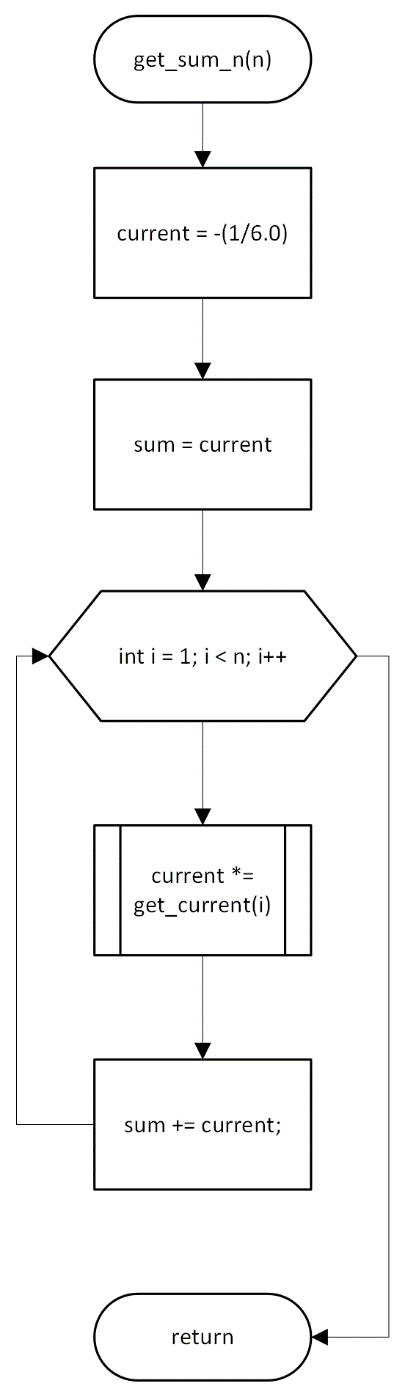


Рисунок – Блок-схема функции get\_sum\_n

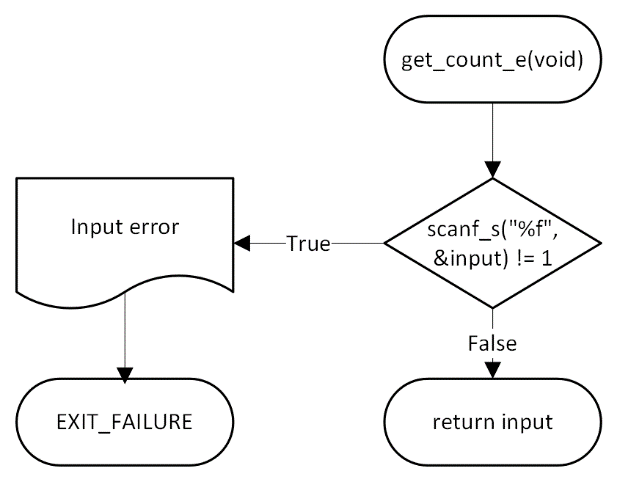


Рисунок – Блок-схема функции get\_count\_e

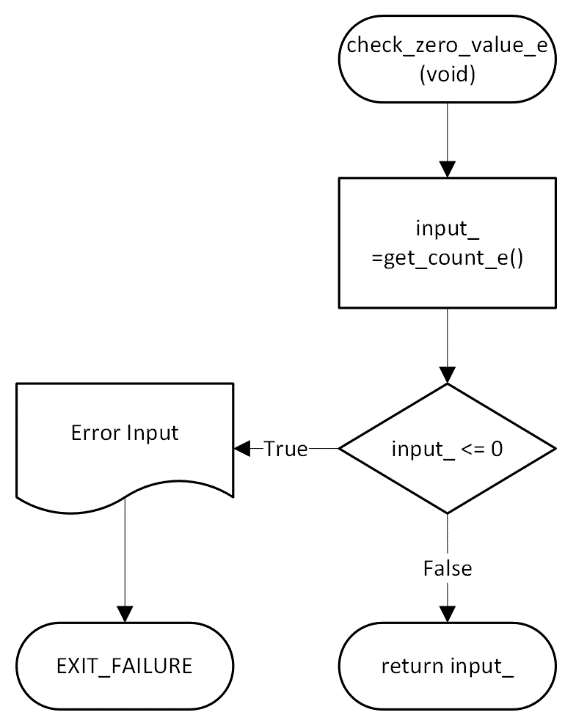


Рисунок – Блок-схема функции check\_zero\_value\_e

* 1. Текст программы на языке С задание 3–2

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <float.h>

/\*\*

\* @brief Функция счета реккурентной функции.

\* @param k - Параметр, заданный пользователем.

\* @return Возвращает посчитанную функцию при k.

\*/

float get\_current(const int k);

/\*\*

\* @brief Функция считает сумму с 1 по n-ный элемент.

\* @param n - значение задаваемое пользователем.

\* @return Возвращает значение суммы.

\*/

float get\_sum(const float e);

/\*\*

\* @brief Функция проверки ввода значения.

\* @remarks При неправильном вводе программа будет закрыта с кодом ошибки /c EXIT\_FAILURE.

\* @return возвращает значение при успешном вводе.

\*/

int get\_count(void);

/\*\*

\* @brief Функция првоерки интервала для части б).

\* @param e - нижнее значение интервала.

\* @param n - верхнее значение интервала.

\* @remarks При неправильной проверке программа будет закрыта с кодом ошибки /c EXIT\_FAILURE.

\*/

void check\_interval(const float e, const int n);

/\*\*

\* @brief Функция проверки на ноль.

\* @remarks При неправильном вводе программа будет закрыта с кодом ошибки /c EXIT\_FAILURE.

\* @return Возрващает значение если без ошибки.

\*/

int check\_zero\_value(void);

/\*\*

\* @brief Функция проверка e

\* @param e - погрешность для проверки

\* @remarks При неправильном вводе программа будет закрыта с кодом ошибки /c EXIT\_FAILURE.

\* @return Возвращает e если без ошибки

\*/

void check\_e(const float e);

/\*\*

\* @brief Функция счета суммы по n-ому члену

\* @param n - переменна заданная пользователем

\* @return Фукнция возвращает значение суммы по n-ому члену

\*/

float get\_sum\_n(const int n);

/\*\*

\* @brief Фукнция счета суммы с погрешностью e

\* @param e - погрешность заданная пользователем

\* @return Возвращает значение посчитанной функции

\*/

float get\_count\_e(void);

/\*\*

\* @brief Функция проверки на ноль.

\* @remarks При неправильном вводе программа будет закрыта с кодом ошибки /c EXIT\_FAILURE.

\* @return Возрващает значение если без ошибки.

\*/

float check\_zero\_value\_e(void);

/\*\*

\* @brief Точка входа.

\* @return Возвращает значение функции с кодом ошибки 0.

\*/

int main(void)

{

puts("Enter counts of values.");

int count = check\_zero\_value();

puts("Enter accuracy of calculation.");

float e = check\_zero\_value\_e();

check\_e(e);

check\_interval(e, count);

printf("Summ of %d member of the sequence: %f\n", count, get\_sum\_n(count));

printf("Summ of member of the sequence with start: %f\n", get\_sum(e));

return 0;

}

float get\_current(int k)

{

return (-1.0) / (2 \* k \* (2 \* k + 1));

}

float get\_sum(float e)

{

float current = -(1/6.0);

float sum = 0;

for (int i = 1; fabs(current) >= e + DBL\_EPSILON; i++)

{

sum += current;

current \*= get\_current(i);

}

return sum;

}

float get\_sum\_n(int n)

{

float current = -(1 / 6.0);

float sum = current;

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

{

current \*= get\_current(i);

sum += current;

}

return sum;

}

int get\_count(void)

{

int input =0;

if (scanf\_s("%d", &input) != 1)

{

printf("Input error");

exit(EXIT\_FAILURE);

}

return input;

}

void check\_interval(float e, int n)

{

if (n < e)

{

puts("Error of Interval");

exit(EXIT\_FAILURE);

}

}

int check\_zero\_value(void)

{

int input\_ = get\_count();

if (input\_ <= 0)

{

puts("Error Input");

exit(EXIT\_FAILURE);

}

return input\_;

}

void check\_e(float e)

{

if (e >= 1 && e <= 0)

{

puts("Error of accuracy");

exit(EXIT\_FAILURE);

}

}

float get\_count\_e(void)

{

float input = 0.0;

if (scanf\_s("%f", &input) != 1)

{

printf("Input error");

exit(EXIT\_FAILURE);

}

return input;

}

float check\_zero\_value\_e(void)

{

float input\_ = get\_count\_e();

if (input\_ <= 0)

{

puts("Error Input");

exit(EXIT\_FAILURE);

}

return input\_;

}

* 1. Результаты выполнения программы 3–2

Результаты выполнения программы в C представлена ниже (Рисунок 20)

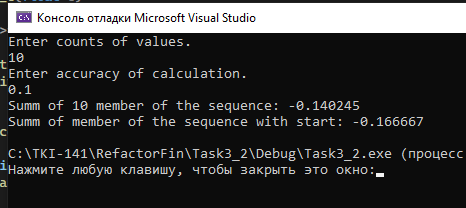


Рисунок – Результат выполнения программы

* 1. Выполнение тестовых примеров задание 3–2

В программе Pythonвыполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 21).

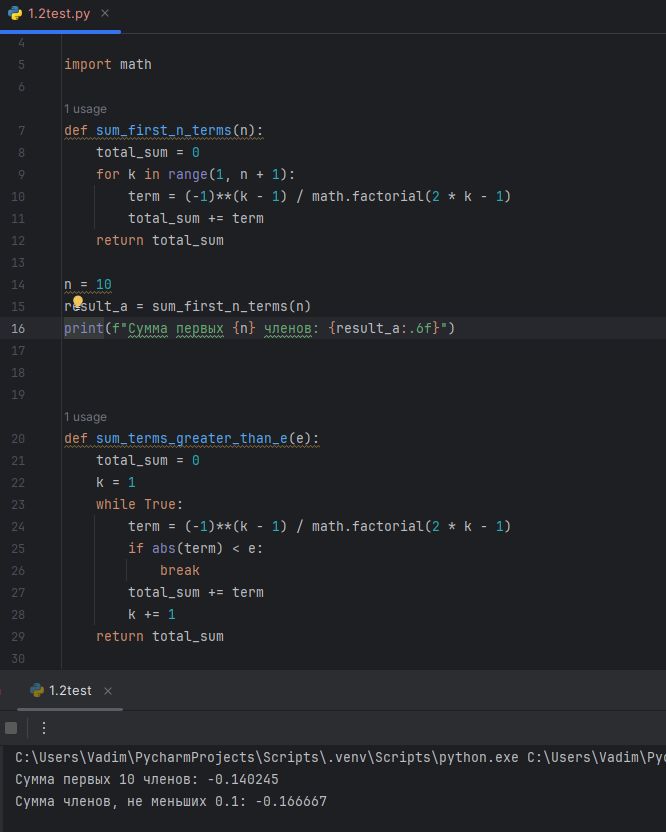


Рисунок – Результат расчёта тестового примера

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий задание 3–2

Ниже представлено доказательство того, что задание 3–2 было принято. (Рисунок 22)

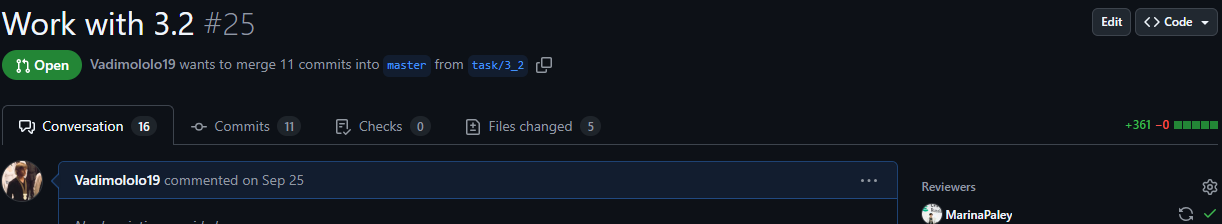


Рисунок – Approve задачи 3–2

1. Решение задачи 3–3
   1. Формулировка задачи 2–3

Протабулировать заданную функцию и сумму функционального ряда разложения этой функции на интервале [a, b] и с шагом h (шаг и интервал задается в константах). Функциональный 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  – Исходные данные

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | Функция y | Сумма | Интервал | ɛ |
| 9 |  |  |  | 20-4 |

* 1. Блок-схема алгоритма задание 3–3

Блок-схема основного алгоритма представлена ниже (Рисунок 23). Блок-схемы функций get\_sum (Рисунок 24), input (Рисунок 25), get\_current (Рисунок 26), get\_function (Рисунок 28), get\_output (Рисунок 28), check\_interval (Рисунок 29), check\_step (Рисунок 30), представлены ниже.

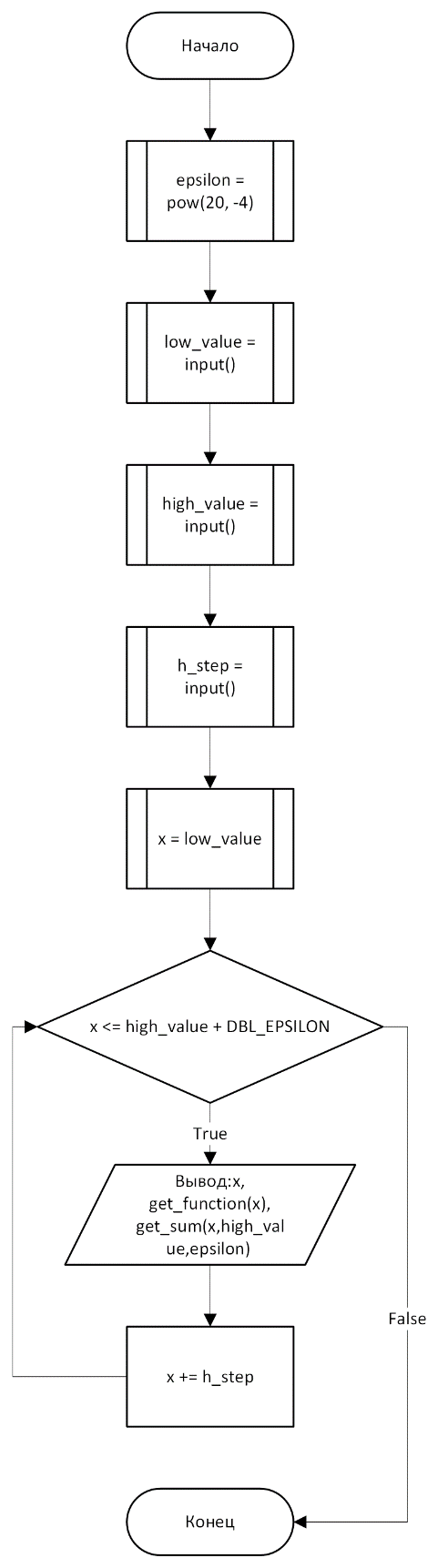


Рисунок  –­ Блок-схема основного алгоритма

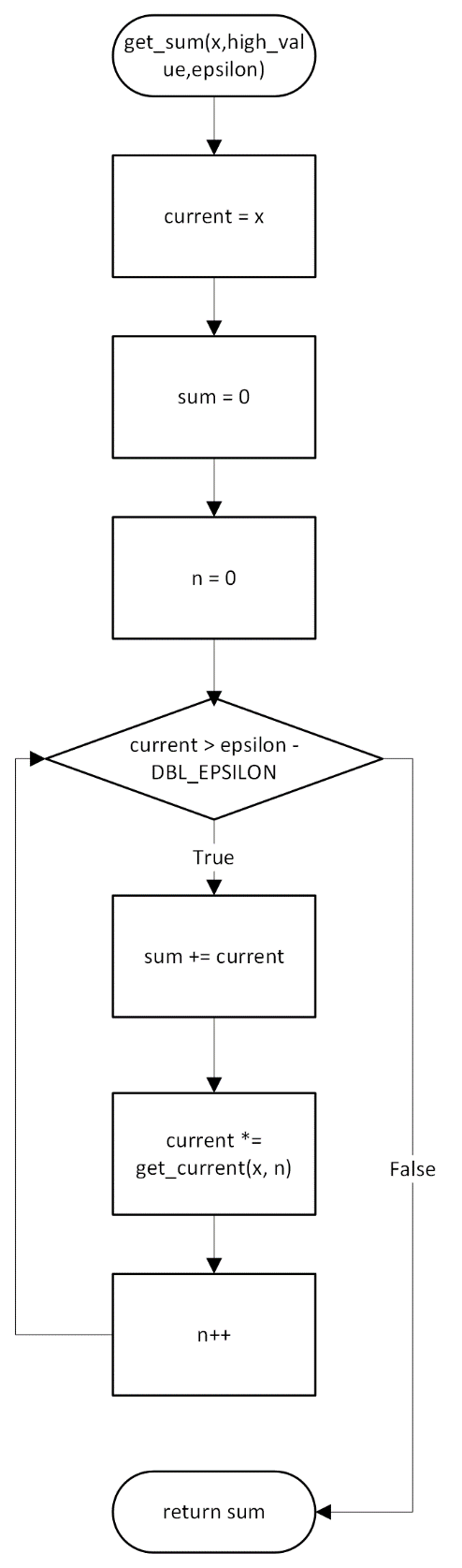


Рисунок – Блок-схема функции get\_sum

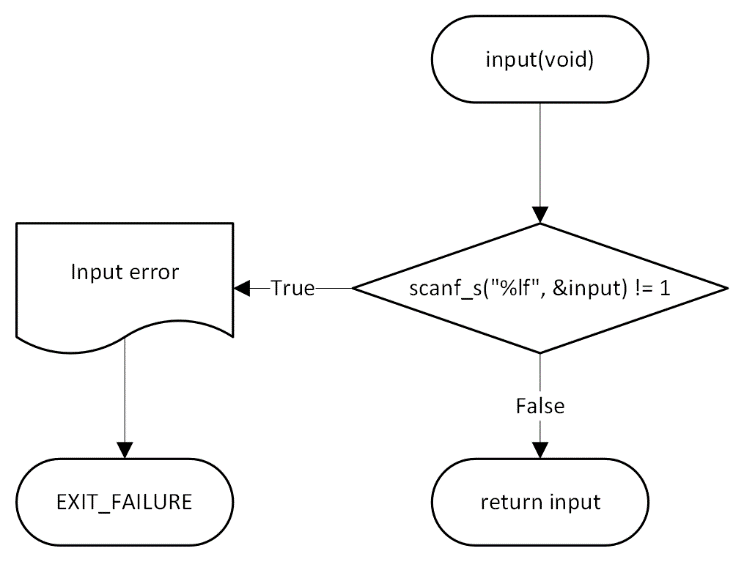


Рисунок – Блок-схема функции input

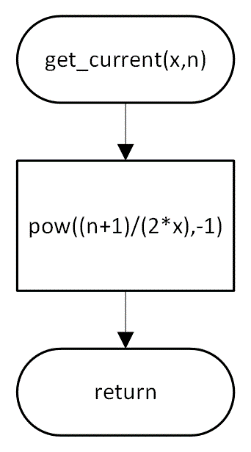


Рисунок – Блок-схема функции get\_current

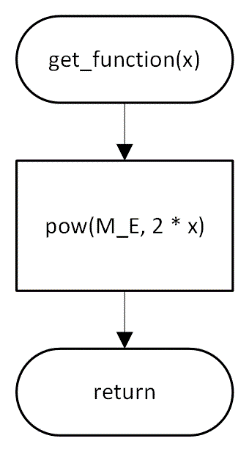


Рисунок – Блок-схема функции get\_function

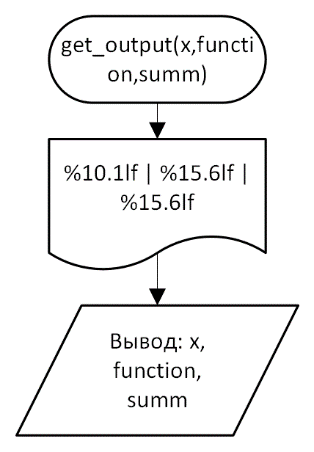


Рисунок – Блок-схема функции get\_output

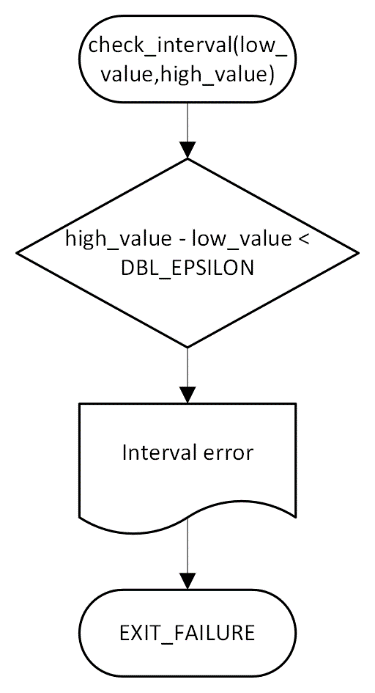


Рисунок – Блок-схема функции check\_interval

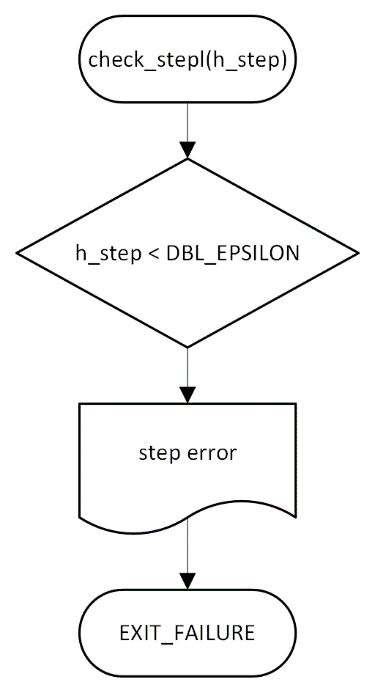


Рисунок – Блок-схема функции check\_step

* 1. Текст программы на языке С задание 3–3

#define \_USE\_MATH\_DEFINES

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <float.h>

/\*\*

\* @brief Функция считает сумму с 1 по n-ный элемент.

\* @param n - значение задаваемое пользователем.

\* @return Возвращает значение суммы.

\*/

double get\_sum(double low\_value, double high\_value, double epsilon);

/\*\*

\* @brief Функция проверки ввода значения.

\* @remarks При неправильном вводе программа будет закрыта с кодом ошибки /c EXIT\_FAILURE.

\* @return возвращает значение при успешном вводе.

\*/

double input(void);

/\*\*

\* @brief Высчитывет рекурентую функцию

\* @param x - значение аргумента

\* @param n - индекс взятого элемента

\* @return Возвращает посчитанную рекурентную функцию

\*/

double get\_current(double x, int n);

/\*\*

\* @brief Высчитывает заданную функцию

\* @param x - значение аргумента

\* @return Возвращает значение функции

\*/

double get\_function(double x);

/\*\*

\* @brief Удобный вывод всех значений

\* @param x - значение аргумента

\* @param function - значение функции

\* @param summ - значение суммы

\*/

void get\_output(double x, double function, double summ);

/\*\*

\* @brief Функция проверки интервала

\* @param low\_value - нижнее значение интервала

\* @param high\_value - верхнее значение интервала

\* @remarks завершает с /c EXIT\_FAILURE если интервал не удовлетворяет требованиям

\* @return возврщает true если выполняется условие

\*/

double check\_interval(double low\_value, double high\_value);

/\*\*

\* @brief Функция проверки шага

\* @param h\_step - значение шага

\* @remarks завершает с /c EXIT\_FAILURE если шаг не удовлетворяет требованиям

\* @return возвращает true если выполянется условие

\*/

double check\_step(double h\_step);

int main(void)

{

double const epsilon = pow(20, -4),low\_value = input(), high\_value = input(), h\_step = input();

double x = low\_value;

while (x <= high\_value + DBL\_EPSILON)

{

get\_output(x, get\_function(x), get\_sum(x,high\_value,epsilon));

x += h\_step;

}

return 0;

}

void get\_output(double x, double function, double summ)

{

printf("%10.1lf | %15.6lf | %15.6lf\n", x, function, summ);

}

double get\_current(double x,int n)

{

return pow((n+1)/(2\*x),-1);

}

double get\_sum(double low\_value,double high\_value, double epsilon)

{

double current = low\_value;

double sum = 0;

int n = 0;

while (current > epsilon - DBL\_EPSILON)

{

sum += current;

current \*= get\_current(low\_value, n);

n++;

}

return sum;

}

double get\_function(double x)

{

return pow(M\_E, 2 \* x);

}

double input(void)

{

double input;

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

{

printf("Input error");

exit(EXIT\_FAILURE);

}

return input;

}

double check\_interval(double low\_value, double high\_value)

{

if (high\_value - low\_value < DBL\_EPSILON) {

printf("Interval error");

exit(EXIT\_FAILURE);

}

}

double check\_step(double h\_step)

{

if (h\_step < DBL\_EPSILON)

{

printf("Step error");

exit(EXIT\_FAILURE);

}

}

* 1. Результаты выполнения программы 3–3

Результаты выполнения программы в C представлена ниже (Рисунок 31)

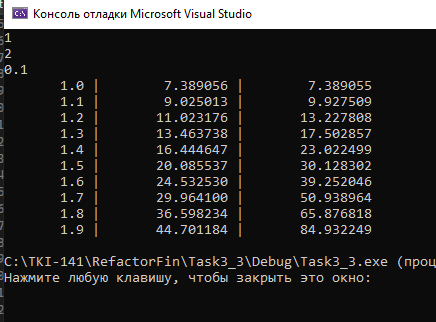


Рисунок – Результаты выполнения программы

* 1. Выполнение тестовых примеров задание 3–3

В программе Pythonвыполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 32).

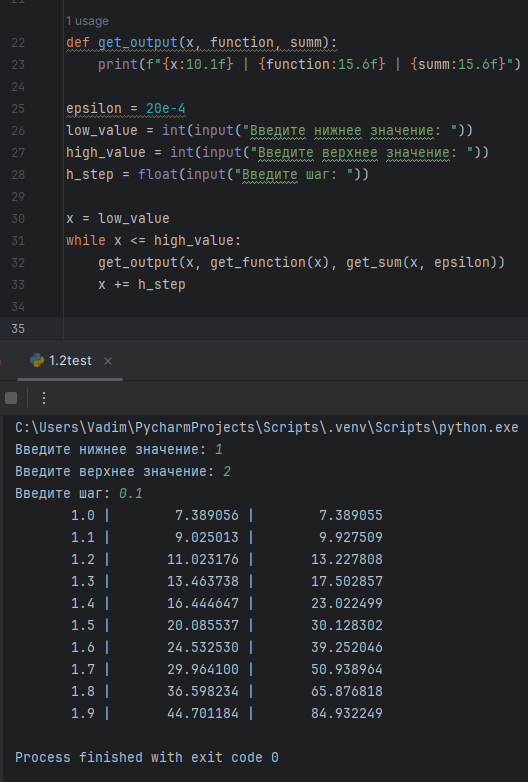


Рисунок – Результат расчета тестового примера

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий задание 3–3

Ниже представлено доказательство того, что задание 3–3 было принято. (Рисунок 33)

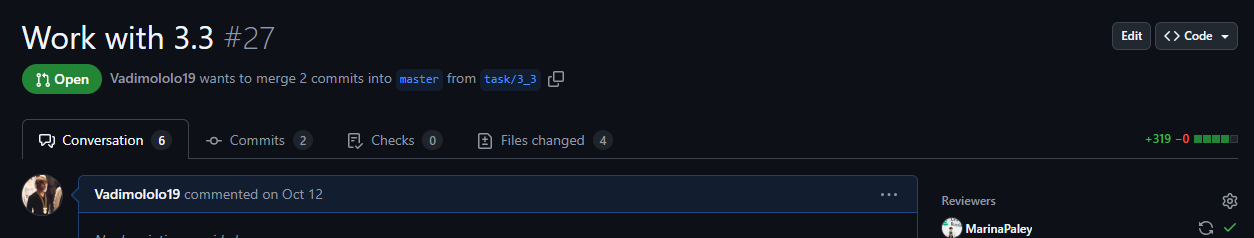


Рисунок – Approve задачи 3–3