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 ПРАКТИЧЕСКОЙ РАБОТЕ №1

По дисциплине «Введение в языки программирования»

Выполнил: ст. гр. ТКИ – 142

Плешанов И.С.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2024

Оглавление

[**Задание 3–1** 3](#_Toc91434532)

[**Условие задания** 3](#_Toc91434533)

[**Блок-схема алгоритма** 4](#_Toc91434534)

[**Программа на языке С** 5](#_Toc91434535)

[**Решение тестовых примеров на С** 6](#_Toc91434536)

[**Решение тестовых примеров в Python** 7](#_Toc91434537)

[**Задание 3–2** 8](#_Toc91434538)

[**Формулировка задания** 8](#_Toc91434539)

[**Блок-схема алгоритма** 9](#_Toc91434540)

[**Программа на языке С** 10](#_Toc91434541)

[**Решение тестовых примеров на С** 11](#_Toc91434542)

[**Решение тестовых примеров в** **Python** 12](#_Toc91434543)

# **Задание 3–1**

## **Условие задания**

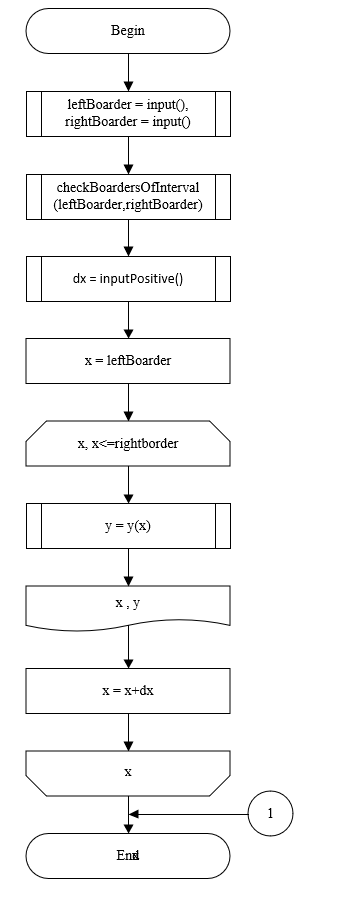
Протабулировать заданную в таблице функцию. Использовать данные в таблице значения шага и интервала в качестве ввода пользователя для решения тестового примера. При невозможности расчёта функции в конкретной точке выводить её значение и надпись, означающую отсутствие решения.

При решении данного задания в MatLab необходимо построить график!

Таблица 1

|  |  |  |
| --- | --- | --- |
| 2 |  |  |

## **Блок-схема алгоритма**

****

**Рисунок 1 – Блок-схема 3-1**

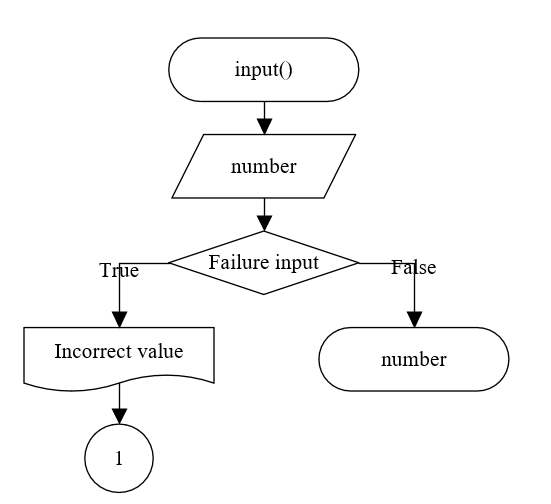


Рисунок 2 – Блок схема input

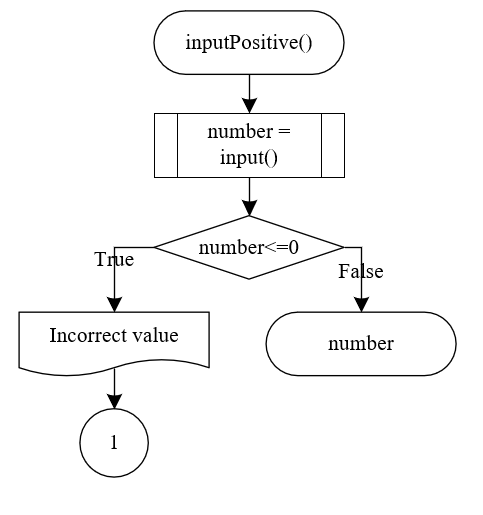


Рисунок 3 – Блок схема inputPositive

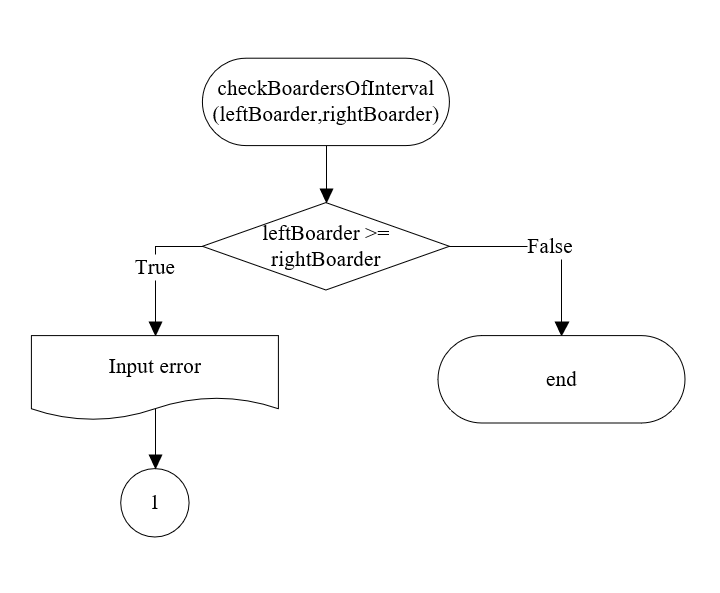


Рисунок 4 – Блок схема checkBoardersOfInterval (leftBoarder,rightBoarder)

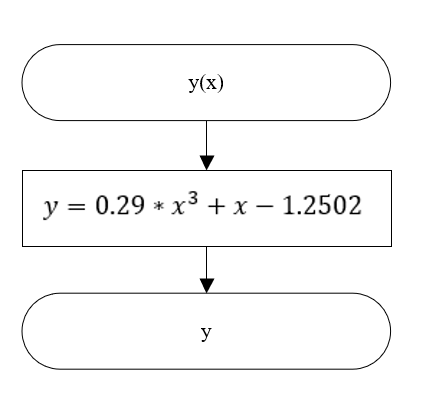
****

Рисунок 5 – Блок схема y(x)

## **Программа на языке С**

#define \_USE\_MATH\_DEFINES

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <float.h>

/\*

\* @brief Функция для вычисления значения функции y при аргументе x.

\* @param x Значение аргумента x.

\* @return Результат вычисления значения функции y при аргументе x.

\*/

double y(const double x);

/\*

\* @brief Функция проверки правильности ввода границ интервала.

\* @param leftBoarder Значение левой границы интервала.

\* @param rightBoarder Значение правой границы интервала.

\*/

void checkBoardersOfInterval(const double leftBoarder, const double rightBoarder);

/\*

\* @brief Функция ввода числа типа double с консоли.

\* @return Возвращает введенное с консоли число типа double.

\*/

double inputDouble(void);

/\*

\* @brief Функция для ввода и проверки корректности шага интервала.

\* @return Возвращает положительное значение шага интервала.

\*/

double inputPositiveStep(void);

/\*

\* @brief Точка входа в программу.

\* @return Возвращает 0 в случае успешного выполнения программы.

\*/

int main(void) {

printf("Enter the left border of the interval:\t");

const double leftBoarder = inputDouble();

printf("Enter the right border of the interval:\t");

const double rightBoarder = inputDouble();

checkBoardersOfInterval(leftBoarder, rightBoarder);

printf("Enter the interval step:\t");

const double dx = inputPositiveStep();

printf("x\t\t y\n");

printf("-----------------------\n");

for (double x = leftBoarder; x <= rightBoarder + DBL\_EPSILON; x += dx) {

printf("%.2lf\t %.5lf\n", x, y(x));

}

return 0;

}

double y(const double x) {

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

}

void checkBoardersOfInterval(const double leftBoarder, const double rightBoarder) {

if (leftBoarder > rightBoarder) {

printf("Input error (the left border should not be greater than the right one)\n");

exit(EXIT\_FAILURE);

}

}

double inputDouble(void) {

double number = 0.0;

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

printf("Input error. Please enter a valid number.\n");

exit(EXIT\_FAILURE);

}

return number;

}

double inputPositiveStep(void) {

double dx = inputDouble();

if (dx <= DBL\_EPSILON) {

printf("Step must be a positive number greater than epsilon.\n");

exit(EXIT\_FAILURE);

}

return dx;

}

## **Решение тестовых примеров на С**

Рисунок 6 – Решение тестового примера для задания 3-1 на С

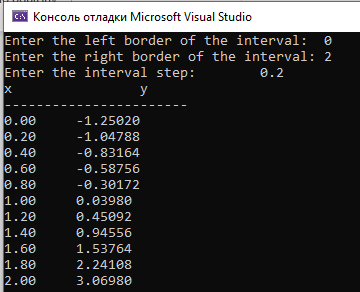
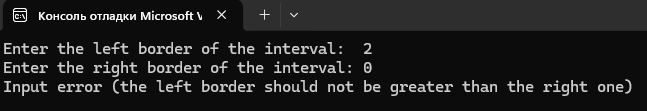
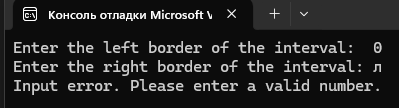
Рисунок 6 – Решение тестового примера для задания 3-1 на С

Рисунок 7 – ошибка решение тестового примера для задания 3-1 на С



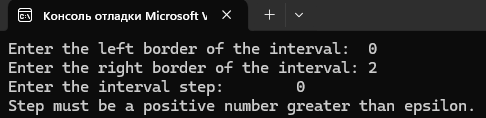
Рисунок 8 – ошибка решение тестового примера для задания 3-1 на С

Рисунок 9 – ошибка решение тестового примера для задания 3-1 на С

## **Решение тестовых примеров в Python**

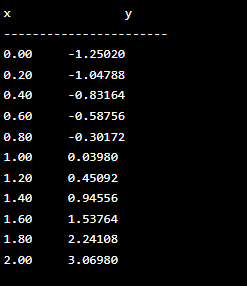


Рисунок 10 - Решение тестового примера задания 3-1 в Python

# **Задание 3–2**

## **Формулировка задания**

Составьте две программы:

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

Помните о проверке пользовательского ввода. Все результаты вывести на экран. Отчёт дополнить блок-схемой. При вычислении факториалов необходимо использовать рекуррентное выражение.

Таблица 2

|  |  |
| --- | --- |
| Вар | Ряд |
| 2 |  |

## **Блок-схема алгоритма**

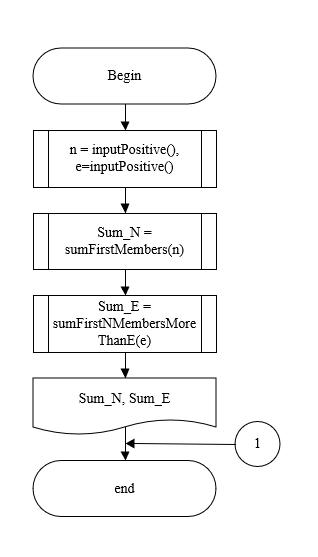


Рисунок 11 – Блок-схема 3-2

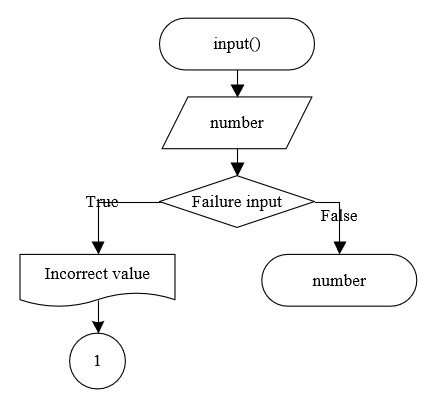


Рисунок 12 – Блок схема input

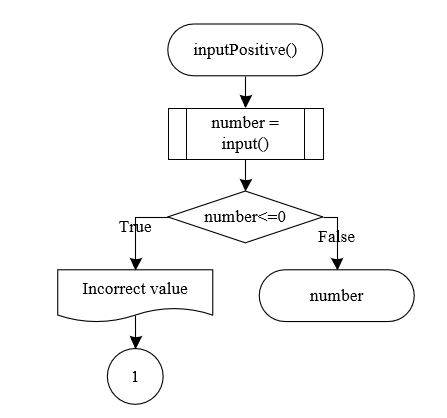


Рисунок 13 – Блок схема inputPositive()

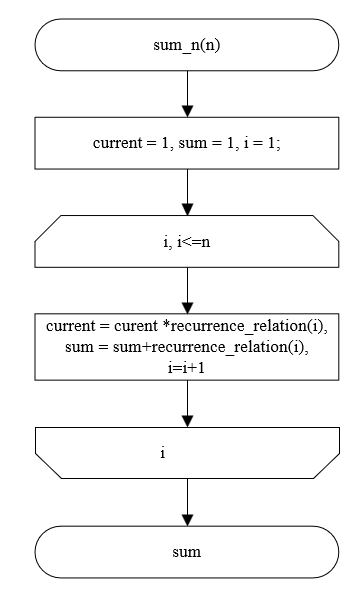


Рисунок 14 – Блок схема sum\_n(n)

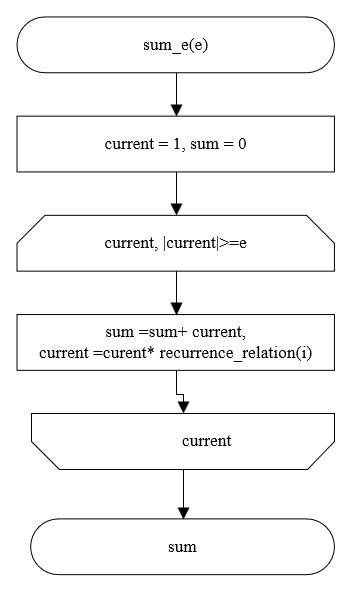


Рисунок 15 – Блок схема sum\_e(e)

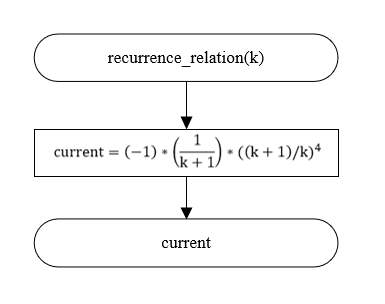


Рисунок 16 - блок схема recurrence\_relation(k)

## **Программа на языке С**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <float.h>

#include <locale.h>

/\*

\* @brief Вычисляет следующее значение в последовательности.

\* @param k Параметр последовательности.

\* @return Значение следующего члена последовательности.

\*/

float recurrence\_relation(const float k);

/\*

\* @brief Суммирует последовательность до достижения заданной точности.

\* @param e Точность, с которой нужно учитывать члены последовательности.

\* @return Сумма всех членов, модуль которых больше или равен e.

\*/

float sum\_e(const float e);

/\*

\* @brief Вычисляет сумму первых n членов последовательности.

\* @param n Количество членов для суммирования.

\* @return Сумма первых n членов.

\*/

float sum\_n(const int n);

/\*

\* @brief Запрашивает и получает ввод целого числа.

\* @return Введенное пользователем целое число.

\* @throws EXIT\_FAILURE при ошибке ввода.

\*/

int input(void);

/\*

\* @brief Получает и проверяет, что введенное число больше нуля.

\* @return Проверенное положительное целое число.

\* @throws EXIT\_FAILURE, если число не больше нуля.

\*/

int check\_above\_zero(void);

/\*

\* @brief Запрашивает и получает ввод числа с плавающей точкой.

\* @return Введенное пользователем число.

\* @throws EXIT\_FAILURE при ошибке ввода.

\*/

float input\_e(void);

/\*

\* @brief Получает и проверяет, что введенное число положительное.

\* @return Проверенное положительное число с плавающей точкой.

\* @throws EXIT\_FAILURE, если число не положительное.

\*/

float check\_e(void);

/\*

\* @brief Главная функция программы.

\* @details Устанавливает локаль, получает ввод от пользователя и вычисляет суммы ряда.

\* @return 0, если программа завершилась успешно.

\*/

int main(void) {

setlocale(LC\_ALL, "Rus");

puts("Введите верхний порог:");

int n = check\_above\_zero();

puts("Введите погрешность:");

float e = check\_e();

printf("Сумма %d членов: %f\n", n, sum\_n(n));

printf("Сумма членов с погрешностью: %f\n", sum\_e(e));

return 0;

}

float recurrence\_relation(const float k) {

return (-1) \* (1.0 / (k + 1)) \* pow(((k + 1) / k), 4);

}

float sum\_e(const float e) {

float current\_ = 1;

float sum = 0;

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

sum += current\_;

current\_ \*= recurrence\_relation(i);

}

return sum;

}

float sum\_n(const int n) {

float current\_ = 1;

float sum = current\_;

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

current\_ \*= recurrence\_relation(i);

sum += current\_;

}

return sum;

}

int input(void) {

int input\_ = 0;

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

printf("Input error\n");

exit(EXIT\_FAILURE);

}

return input\_;

}

int check\_above\_zero(void) {

int input\_ = input();

if (input\_ <= 0) {

puts("Error Input\n");

exit(EXIT\_FAILURE);

}

return input\_;

}

float input\_e(void) {

float input = 0.0;

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

printf("Input error\n");

exit(EXIT\_FAILURE);

}

return input;

}

float check\_e(void) {

float input\_ = input\_e();

if (input\_ <= 0) {

puts("Error Input\n");

exit(EXIT\_FAILURE);

}

return input\_;

}

## **Решение тестовых примеров на С**

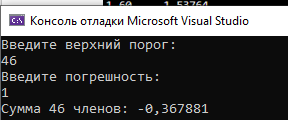


Рисунок 17 – Решение тестового примера для задания 3-2 на С

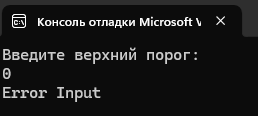
Рисунок 18 – ошибка решение тестового примера для задания 3-2 на С 

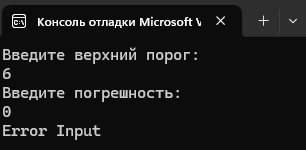
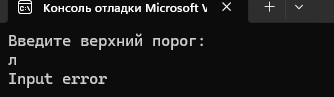
Рисунок 19 – ошибка решение тестового примера для задания 3-2 на С 

Рисунок 20 – ошибка решение тестового примера для задания 3-2 на С 

## **Решение тестовых примеров в Python**

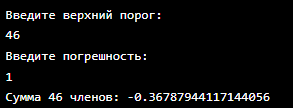


Рисунок 21 – Решение тестового примера для задания 3-2 в Python