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

По дисциплине «Языки программирования»

ВАРИАНТ 1

Выполнил: ст. гр. ТКИ - 141

Абузина Елизавета Юрьевна

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

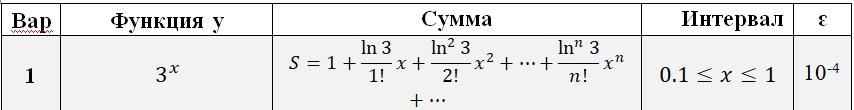
(Проверил: к.т.н, доц. Балакина Е. П.)

Москва 2024

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  – Исходные данные



1. Блок-схема алгоритма

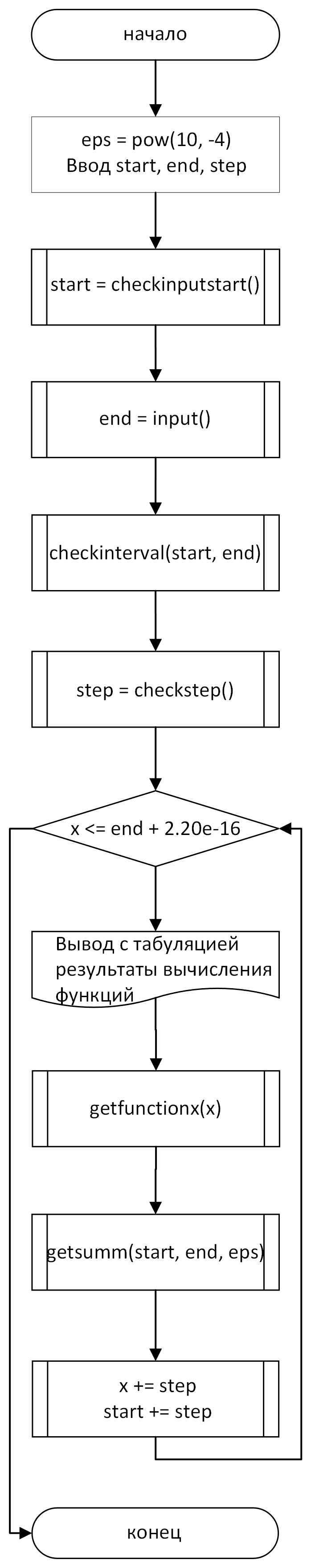


Рисунок  ­ Блок-схема основного алгоритма

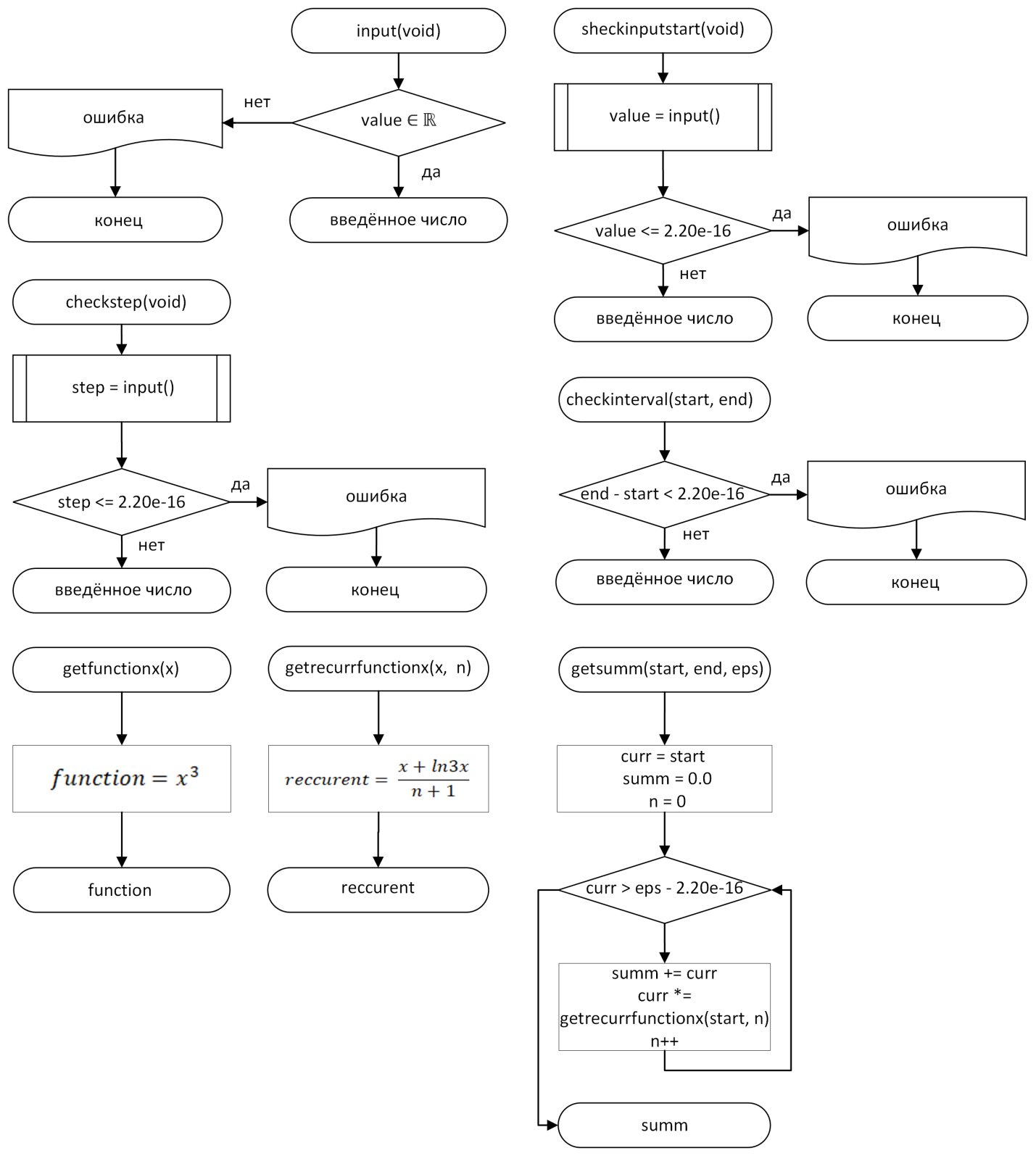


Рисунок  – Блок-схема используемых функций

1. Текст программы на языке C

#include <stdio.h>

#include <locale.h>

#include <math.h>

#include <stdlib.h>

#include <errno.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 отрицательном значении ввода программа завершает выполнение

\* @return Вещественное число

\*/

double checkinputstart(void);

/\*\*

\* @brief Проверка интервала.

\* @param start - начальное значение интервала.

\* @param end - конечное значение интервала.

\* @return Значение "true" в случае выполнения условия.

\*/

double checkinterval(const double start, const double end);

/\*\*

\* @brief Проверка шага.

\* @param При неправильном вводе программа завершает выполнение.

\* @return Значение шага в случае выполнения условия.

\*/

double checkstep(void);

/\*\*

\* @brief Вычисление значения функции.

\* @param x - значение аргумента функции.

\* @return Значение функции при значении аргумента x.

\*/

double getfunctionx(const double x);

/\*\*

\*@brief Вычисление значения функции.

\* @param x - значение аргумента функции.

\* @param n - индекс элемента.

\* @return Значение функции при значении аргумента x.

\*/

double getrecurrfunctionx(const double x, const int n);

/\*\*

\* @brief Вычисление суммы функционального ряда с 1 по n - ный элемент.

\* @param start - начальное значение интервала.

\* @param end - конечное значение интервала.

\* @param eps - заданная точность вычисления.

\* @return Сумма функционального ряда с 1 по n - ный элемент.

\*/

double getsumm(const double start, const double end, const double eps);

/\*

\* @brief Точка входа в программу.

\* @return Код ошибки.

\*/

int main(void)

{

setlocale(LC\_ALL, "Russian");

const double eps = pow(10, -4);

printf("Введите начало интервала: ");

double start = checkinputstart();

printf("Введите конец интервала: ");

double end = input();

checkinterval(start, end);

printf("Введите шаг на интервале: ");

double step = checkstep();

double x = start;

printf("%-10s %-20s %-20s", "x", "Значение функции", "Значение функционального ряда\n");

while (x <= end + DBL\_EPSILON)

{

printf("%-10.2lf | %-20.10lf | %-20.10lf\n", x, getfunctionx(x), getsumm(start, end, eps));

x += step;

start += step;

}

return 0;

}

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 checkinputstart(void)

{

double value = input();

if (value <= DBL\_EPSILON)

{

errno = EDOM;

printf("Значение функционального ряда не существует на заданном начале интервала");

exit(EXIT\_FAILURE);

}

return value;

}

double checkinterval(const double start, const double end)

{

if (end - start < DBL\_EPSILON)

{

errno = EIO;

perror("Ошибка ввода числового значения: ");

exit(EXIT\_FAILURE);

}

}

double checkstep(void)

{

double step = input();

if (step < DBL\_EPSILON)

{

errno = EIO;

perror("Ошибка ввода числового значения: ");

exit(EXIT\_FAILURE);

}

return step;

}

double getfunctionx(const double x)

{

return pow(3, x);

}

double getrecurrfunctionx(const double x, const int n)

{

return (x + log(3\*x))/(n+1);

}

double getsumm(const double start, const double end, const double eps)

{

double curr = start;

double summ = 0.0;

int n = 0;

while (curr > eps - DBL\_EPSILON)

{

summ += curr;

curr \*= getrecurrfunctionx(start, n);

n++;

}

return summ;

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 3,Рисунок 4, Рисунок 5, Рисунок 6, Рисунок 7, Рисунок 8).

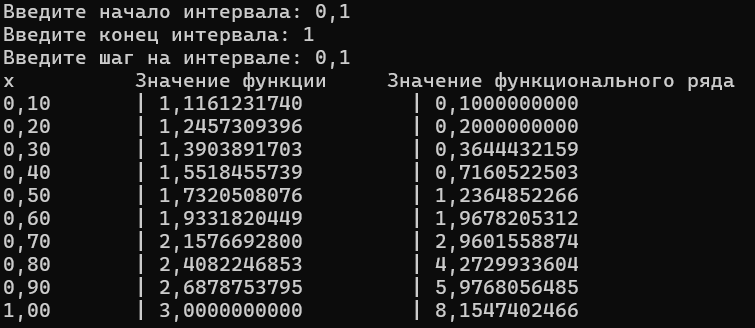


Рисунок  – Результаты выполнения программы



Рисунок 4 – Результаты выполнения программы при альтернативном вводе

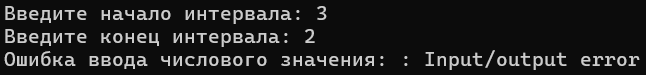


Рисунок 5 – Результаты выполнения программы при альтернативном вводе



Рисунок 6 – Результаты выполнения программы при альтернативном вводе

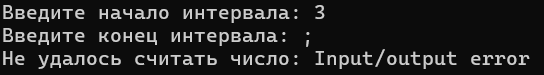


Рисунок 7 – Результаты выполнения программы при альтернативном вводе

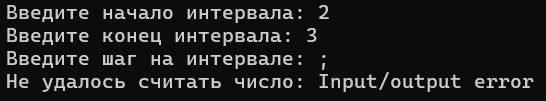


Рисунок 8 – Результаты выполнения программы при альтернативном вводе

1. Выполнение тестовых примеров

В программе IDLE выполнен тестовый пример. Результаты его выполнения представлен ниже (Рисунок 9, Рисунок 10, Рисунок 11, Рисунок 12, Рисунок 13, Рисунок 14).

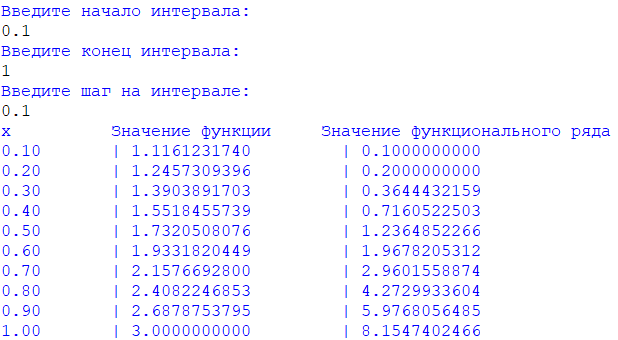


Рисунок 9 – Результат работы программы



Рисунок 10 – Результат работы программы при альтернативном вводе

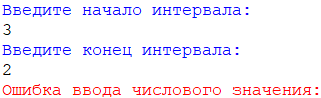


Рисунок 11 – Результат работы программы при альтернативном вводе



Рисунок 12 – Результат работы программы при альтернативном вводе

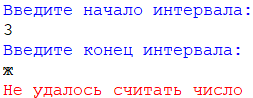


Рисунок 13 – Результат работы программы при альтернативном вводе

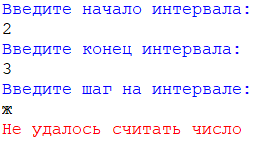


Рисунок 14 – Результат работы программы при альтернативном вводе

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

