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 5

Выполнил: ст. гр. ТКИ-142

Скрипников Егор Сергеевич

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 1 – Формулировка задания 3.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вар | Функция y | Сумма | Интервал | ɛ |
| 5 |  |  |  | 30-5 |

1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций расчета значений функции, последовательности и ввода представлены ниже (Рисунок 2, 3, 4, 5, 6, 7).



Рисунок  ­ Блок-схема основного алгоритма



Рисунок 2 – Блок-схема функции input()



Рисунок 3 – Блок-схема функции checkInterval(a, b)



­Рисунок 4 – Блок-схема функции checkStep(step)



Рисунок 5 – Блок-схема функции getFunction(x)



Рисунок 6 – Блок-схема функции getElement(x, n)



Рисунок 7 – Блок-схема функции getSum(x, epsilon)

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <math.h>

#include <float.h>

*/\*\**

*\* @brief Считывает введенное значение*

*\* @return Возвращает введенное значение*

*\*/*

double input(void);

*/\*\**

*\* @brief Проверяет правильность введенных значений интервала*

*\* @param a начало интервала*

*\* @param b конец интервала*

*\* @return Возвращает ошибку, если введен некорректный интервал*

*\*/*

void checkInterval(const double a, const double b);

*/\*\**

*\* @brief Проверяет правильность введенного значения шага*

*\* @param step величина шага*

*\* @return Возвращает ошибку, если введен некорректный шаг*

*\*/*

void checkStep(const double step);

*/\*\**

*\* @brief Вызвает функцию*

*\* @param x значение переменной x*

*\* @return Возвращает значение функции если нет деления на нуль, в ином случае вызывает ошибку*

*\*/*

double getFunction(const double x);

*/\*\**

*\* @brief Рассчитывает значение следующего элемента последовательности*

*\* @param x значение переменной*

*\* @param n текущий индекс элемента последовательности*

*\* @return Возвращает значение следующего элемента последовательности*

*\*/*

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

*/\*\**

*\* @brief Рассчитывает значение суммы n членов*

*\* @param x значение переменной*

*\* @param epsilon значение величины с точностью которой следует вычислить значение суммы членов последовательности*

*\* @return Возвращает значение суммы n членов последовательности*

*\*/*

double getSum(const double x, const double epsilon);

*/\*\**

*\* @brief Точка входа в программу*

*\* @return Возвращает 0 в случае успеха*

*\*/*

int main(void)

{

    const double epsilon = pow(30, -5);

    puts("Enter interval value:");

    const double a = input();

    const double b = input();

    checkInterval(a, b);

    puts("Enter step value:");

    const double step = input();

    checkStep(step);

    for (double x = a; x <= b + DBL\_EPSILON; x += step)

    {

        if ((x) >= 1)

        {

            errno = EDOM;

            perror("Invalid input for log function!");

        }

        printf("\nx = %.2f\tf(x) = %.6f\t\tS(x) = %.6f\n", x, getFunction(x), getSum(x, epsilon));

    }

    return 0;

}

double input(void)

{

    double value = 0.0;

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

    if (result != 1)

    {

        errno = EIO;

        perror("Input error!");

        exit(EXIT\_FAILURE);

    }

    return value;

}

void checkInterval(const double a, const double b)

{

    if (b - a < DBL\_EPSILON)

    {

        errno = EINVAL;

        perror("Interval input error!");

        exit(EXIT\_FAILURE);

    }

}

void checkStep(const double step)

{

    if (step <= DBL\_EPSILON)

    {

        errno = EINVAL;

        perror("Step input error!");

        exit(EXIT\_FAILURE);

    }

}

double getFunction(double x)

{

    return (0.25 \* log((1 + x) / (1 - x))) + (0.5 \* atan(x));

}

double getElement(const double x, const int n)

{

    return (pow(x, (4 \* n + 1)) / (4 \* n + 1));

}

double getSum(const double x, const double epsilon)

{

    double element = getElement(x, 0);

    double sum = 0.0;

    int n = 0;

    while (fabs(element) > epsilon)

    {

        sum += element;

        n++;

        element = getElement(x, n);

    }

    return sum;

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 8, 9, 10, 11, 12, 13).

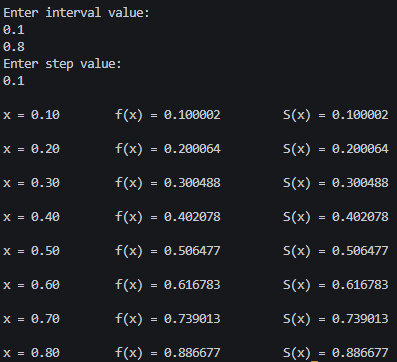


Рисунок 8 – Результаты выполнения программы



Рисунок 9 – Вывод, когда начало интервала - буква

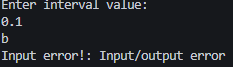


Рисунок 10 – Вывод, когда конец интервала буква

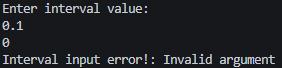


Рисунок 11 – Вывод, когда конец интервала меньше начала

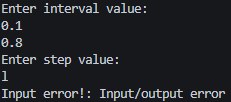


Рисунок 12 – Вывод, когда введенный шаг - буква

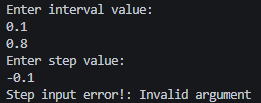


Рисунок 13 – Вывод, когда шаг отрицательный

1. Выполнение тестовых примеров

При помощи программы Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 14, 15, 16).

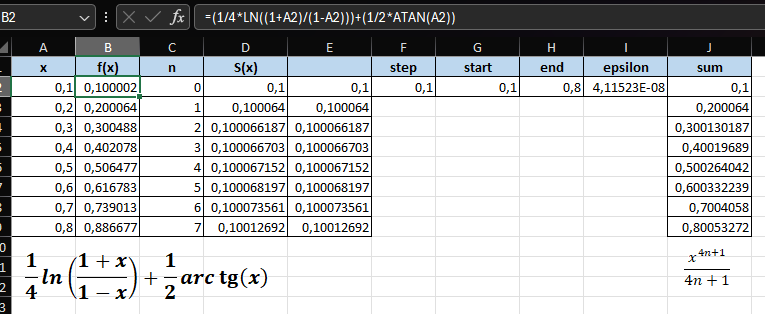


Рисунок 14 – Расчет значения функции по формуле

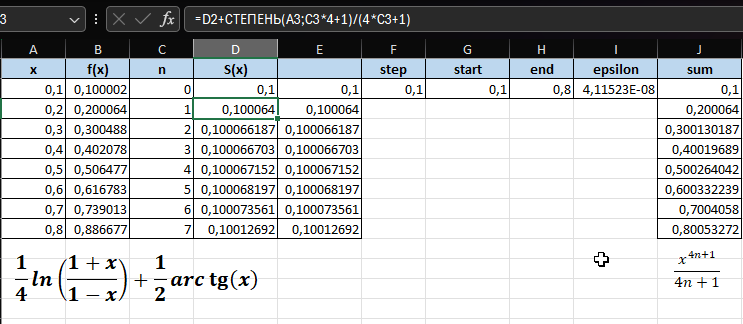


Рисунок 15 – Расчет значения члена последовательности

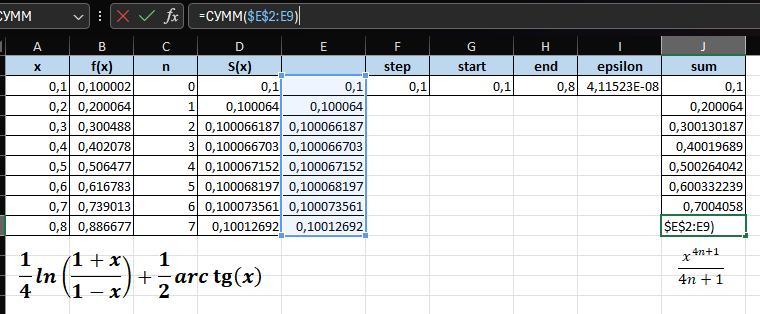


Рисунок 16 - Расчет значения суммы текущего количества элементов последовательности

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

