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

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 1 – Формулировка задания 3.2

|  |  |
| --- | --- |
| Вар | Ряд |
| 5 |  |

1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций расчета суммы первых *n* членов последовательности, суммы всех членов последовательности, по модулю не меньших заданного числа *e*, и ввода представлены ниже (Рисунок 2, 3, 4, 5, 6, 7).



Рисунок 1 ­ Блок-схема основного алгоритма



Рисунок 2 – Блок-схема функции input()



Рисунок 3 – Блок-схема функции inputInt()



Рисунок 4 – Блок-схема функций checkQuantity(quantity)



Рисунок 5 – Блок-схема функции nextElement(previousElement, k)



Рисунок 6 – Блок-схема функции getNSum(n)



Рисунок 7 – Блок-схема функции getSumNotLessThanE(e)

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 считанное значение*

*\*/*

double input(void);

*/\*\**

*\* @brief Считывает введеное целое значение*

*\* @return Возвращает считанное значение*

*\*/*

int inputInt(void);

*/\*\**

*\* @brief Прооверяет неотрицательность введенного значения*

*\* @param quantity введенное значение*

*\* @return Возвращает ошибку, если значение меньше нуля*

*\*/*

void checkQuantity(const int quantity);

*/\*\**

*\* @brief Рассчитывает значение следующего элемента последовательности*

*\* @param previousElement значение предыдущего элемента последовательности*

*\* @param k текущий индекс последовательности*

*\* @return Возвращает значение следующего элемента последовательности*

*\*/*

double nextElement(const double previousElement, const int k);

*/\*\**

*\* @brief Рассчитывает значение суммы n членов*

*\* @param n Количество членов последовательности*

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

*\*/*

double getNSum(const int n);

*/\*\**

*\* @brief Рассчитывает значение суммы всех членов последовательности, не меньших заданного числа e*

*\* @param n Количество членов последовательности*

*\* @param e Заданное число*

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

*\*/*

double getSumNotLessThanE(const double e);

*/\*\**

*\* @brief Точка входа в программу*

*\* @return В случае успеха, возвращает 0*

*\*/*

int main(void)

{

    puts("Enter number of sequence elements:");

    const int n = inputInt();

    puts("Enter e value:");

    const double e = input();

    checkQuantity(e);

    printf("Sum of N elements = %.3lf\n", getNSum(n));

    printf("Sum of all elements not less than e = %.3lf\n", getSumNotLessThanE(e));

    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;

}

int inputInt(void)

{

    int quantity = 0;

    int result = scanf\_s("%d", &quantity);

    if (result != 1)

    {

        errno = EIO;

        perror("Input Error!");

        exit(EXIT\_FAILURE);

    }

    checkQuantity(quantity);

    return quantity;

}

void checkQuantity(int quantity)

{

    if (quantity < 0)

    {

        errno = EIO;

        perror("Value cannot be lower than zero!");

        exit(EXIT\_FAILURE);

    }

}

double nextElement(const double previousElement, const int k)

{

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

}

double getNSum(const int n)

{

    double element = 1.0; *// Первый член последовательности равен 1^4/1! = 1*

    double sum = element;

    for (int k = 1; k < n; ++k)

    {

        element = nextElement(element, k);

        sum += element;

    }

    return sum;

}

double getSumNotLessThanE(const double e)

{

    double element = 1.0; *// Первый член последовательности равен 1^4/1! = 1*

    double sum = element;

    int k = 1;

    while (fabs(element) >= e+DBL\_EPSILON)

    {

        element = nextElement(element, k);

        sum += element;

        k++;

    }

    return sum;

}

1. Результаты выполнения программы

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

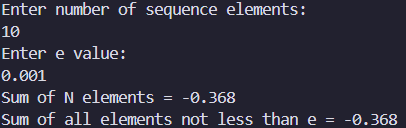


Рисунок 8 – Результаты выполнения программы



Рисунок 9 – Результаты выполнения программы при вводе отрицательного числа в количество членов последовательности.



Рисунок 10 – Результаты выполнения программы если введенное число – буква

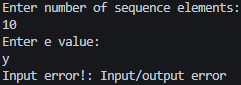
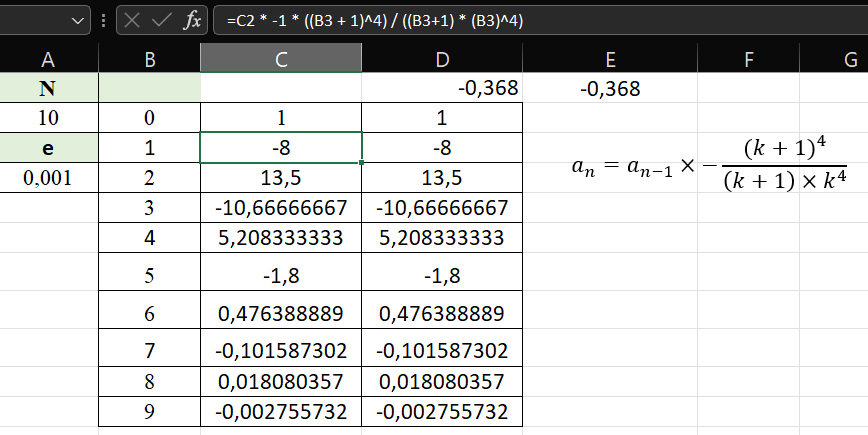


Рисунок 11 – Результаты выполнения программы, когда введенное число точности - буква

1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 12, 13).



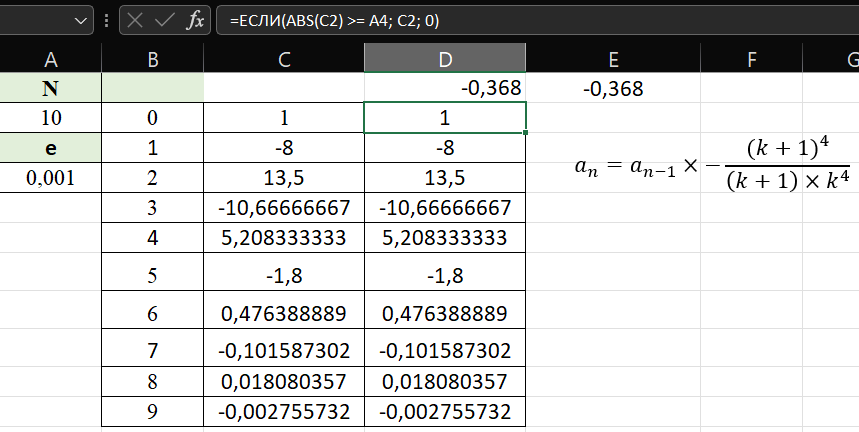
Рисунок 12 – Результат расчета суммы n членов

Рисунок 13 – Результат расчета суммы меньше чем e

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