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 №3.2

По дисциплине «Языки программирования»

ВАРИАНТ 14

Выполнил: ст. гр. ТКИ-142

Кузнецов Денис Константинович

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 |
| 14 |  |

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, 8).



Рисунок  ­ Блок-схема основного алгоритма



Рисунок 2 – Блок-схема функции input()



Рисунок 3 – Блок-схема функции inputInt()



Рисунок 4 – Блок-схема функций isPositiveInt()



Рисунок 5 – Блок-схема функции isPositiveDouble()



Рисунок 6 – Блок-схема функции getSum(n)



Рисунок 7 – Блок-схема функции getGreaterThanE(e)



Рисунок 8 – Блок-схема функции nextTerm(current, k)

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <stdbool.h>

#include <math.h>

#include <float.h>

/\*\*

\* @brief Проверка введенного значения

\* @return Возвращает значение

\*/

double input(void);

/\*\*

\* @brief Считывает введенное целое значение

\* @return Возвращает считанное проверенное значение, иначе - ошибка

\*/

int intInput(void);

/\*\*

\* @brief Выводит сумму n элементов

\* @param n - количество элементов последовательности для суммирования

\* @return Рассчитанное значение

\*/

double getSum(const int n);

/\*\*

\* @brief Рассчитывает значение суммы всех членов последовательности, по модулю не меньше заданного числа e

\* @param e Значение заданного числа e

\* @return Рассчитанное значение суммы

\*/

double getGreaterThanE(const double e);

/\*\*

\* @brief Проверяет, что введенное целое число положительное

\* @return Возвращает считанное проверенное значение, иначе - ошибка

\*/

int isPositiveInt(void);

/\*\*

\* @brief Проверяет, что введенное число положительное

\* @return Возвращает считанное проверенное значение, иначе - ошибка

\*/

double isPositiveDouble(void);

/\*\*

\* @brief Рассчитывает следующий рекуррентный член последовательности

\* @param currentTerm Текущий член последовательности

\* @param k Текущий индекс элемента

\* @return Выводит рассчитанный следующий рекуррентный член последовательности

\*/

double nextTerm(const double current, const int k);

/\*\*

\* @brief Точка входа в программу

\* @return 0 в случае успеха

\*/

int main(void) {

printf("Введите количество элементов последовательности (n): ");

const int n = isPositiveInt();

printf("Введите значение e (точность для модуля): ");

const double e = isPositiveDouble();

printf("Сумма первых %d членов последовательности: %.10lf\n", n, getSum(n));

printf("Сумма членов последовательности, по модулю не меньше числа %.10lf: %.10lf\n", e, getGreaterThanE(e));

return 0;

}

double input(void) {

double value = 0.0;

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

if (result != 1) {

errno = EIO;

perror("Ошибка ввода!");

exit(EXIT\_FAILURE);

}

return value;

}

double isPositiveDouble(void) {

double value = input();

if (value <= DBL\_EPSILON) {

errno = EINVAL;

perror("Число слишком маленькое или отрицательное!");

exit(EXIT\_FAILURE);

}

return value;

}

int intInput(void) {

int value = 0;

int result = scanf("%d", &value);

if (result != 1) {

errno = EIO;

perror("Ошибка ввода");

exit(EXIT\_FAILURE);

}

return value;

}

int isPositiveInt(void) {

int value = intInput();

if (value <= 0) {

errno = EINVAL;

perror("Число должно быть положительным!");

exit(EXIT\_FAILURE);

}

return value;

}

double nextTerm(const double current, const int k) {

return current \* -1.0 \* (2.0 + k) / (k + 1.0);

}

double getSum(const int n) {

double sum = 0.0;

double currentTerm = 1.0;

for (int k = 0; k < n; k++) {

sum += currentTerm;

currentTerm = nextTerm(currentTerm, k);

}

return sum;

}

double getGreaterThanE(const double e) {

double sum = 0.0;

double currentTerm = 1.0;

int k = 1;

while (fabs(currentTerm) >= e+DBL\_EPSILON) {

sum += currentTerm;

currentTerm = nextTerm(currentTerm, k);

k++;

}

return sum;

}

1. Результаты выполнения программы

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

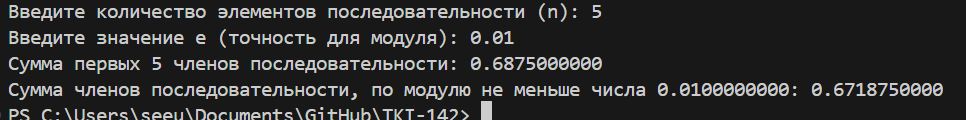


Рисунок 9 – Результаты выполнения программы



Рисунок 10 – Результаты выполнения программы при вводе отрицательного числа, как количества элементов последовательности



Рисунок 11 – Результаты выполнения программы при вводе букв, как количества элементов последовательности

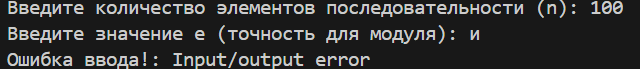


Рисунок 12 – Результаты выполнения программы, при вводе числа E как буквы

1. Выполнение тестовых примеров

На языке Python выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 13).

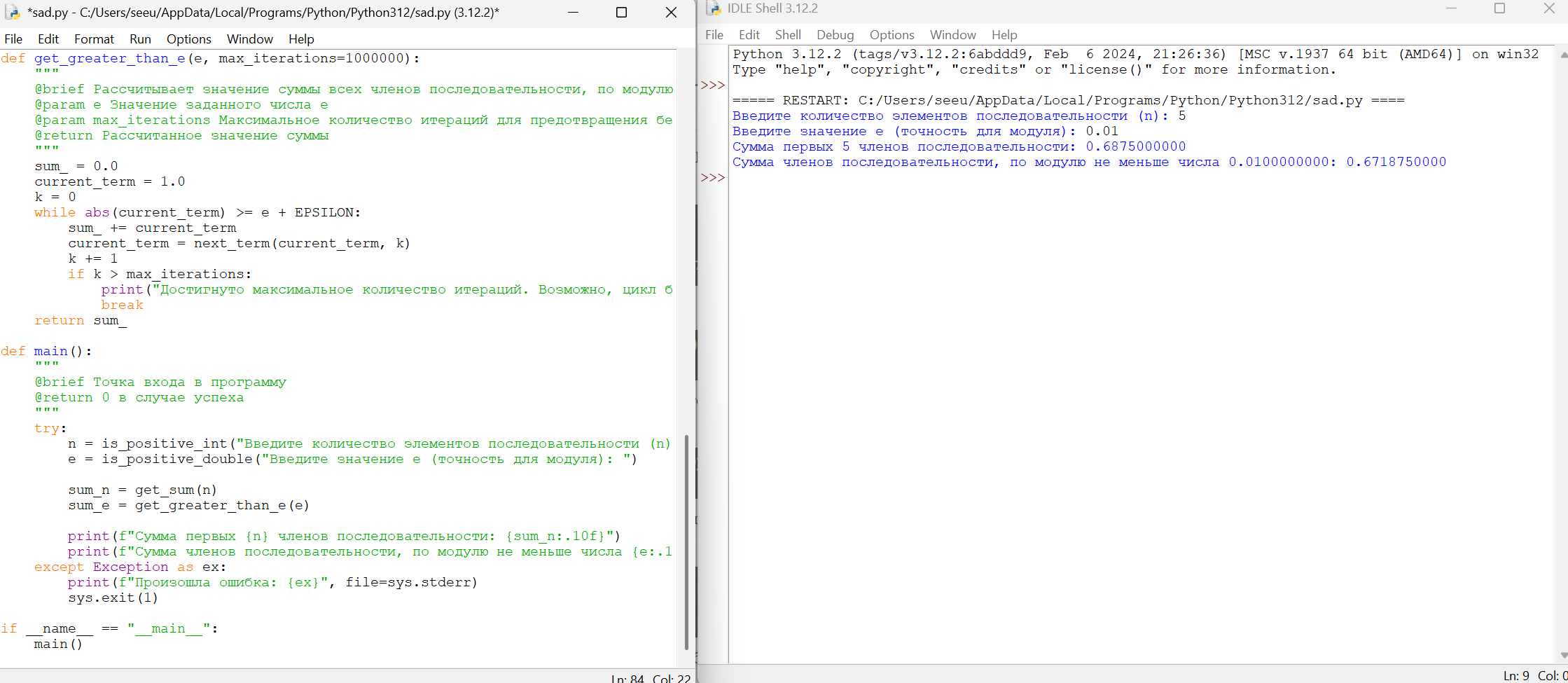


Рисунок 13 – Результат расчета суммы n членов и суммы меньше, чем e

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

