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 №4

По дисциплине «Языки программирования»

ВАРИАНТ 17

Выполнил: ст. гр. ТКИ-141

Лагутин Владимир 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 – Исходные данные

|  |  |
| --- | --- |
| Вариант | Задачи |
| 17 |  |

1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций расчета значений a и b представлены ниже (Рисунок 2).

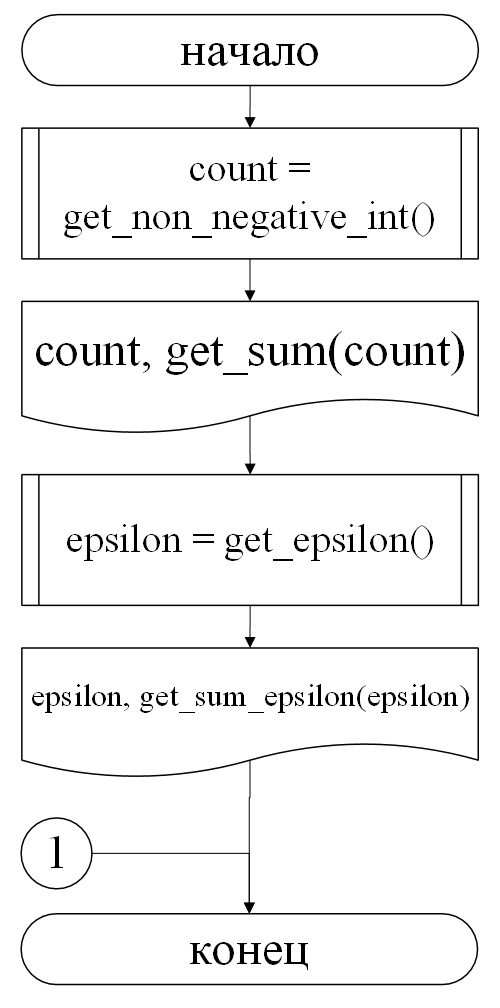
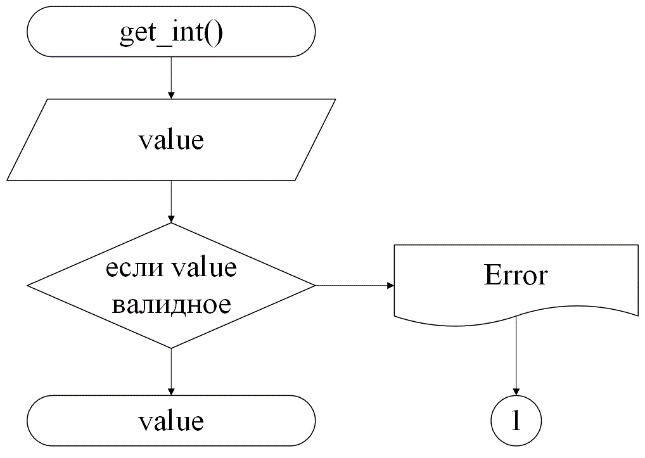
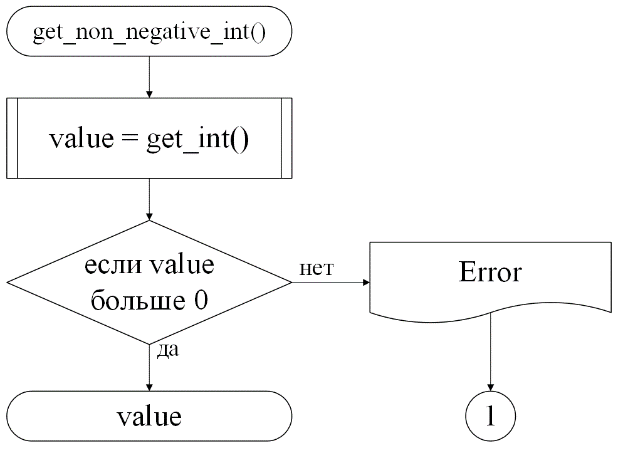
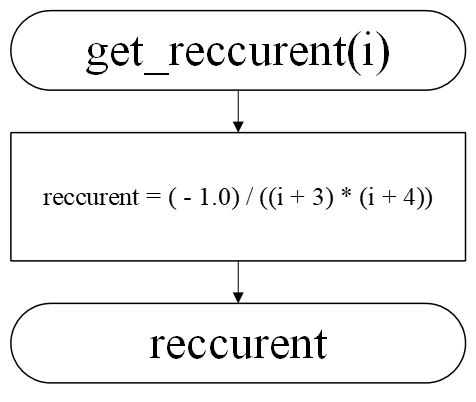
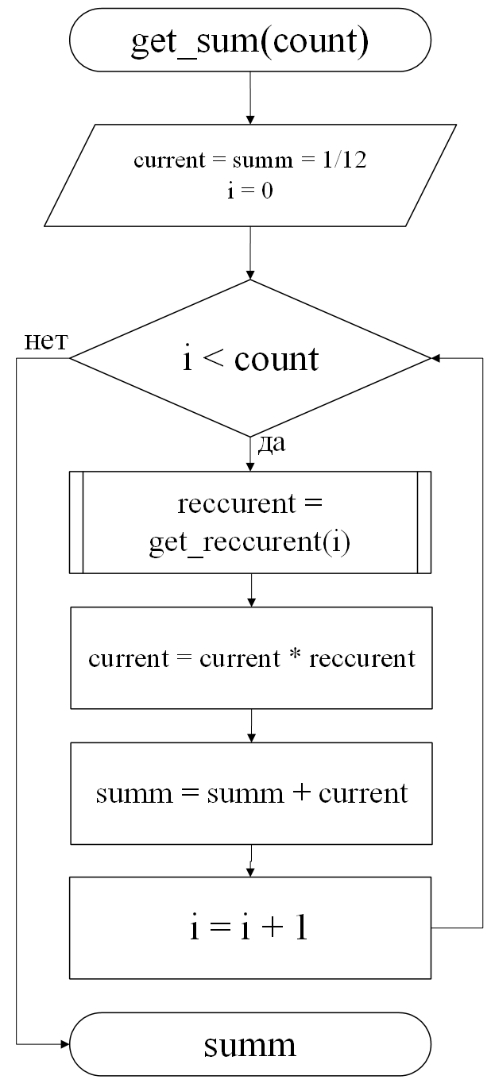
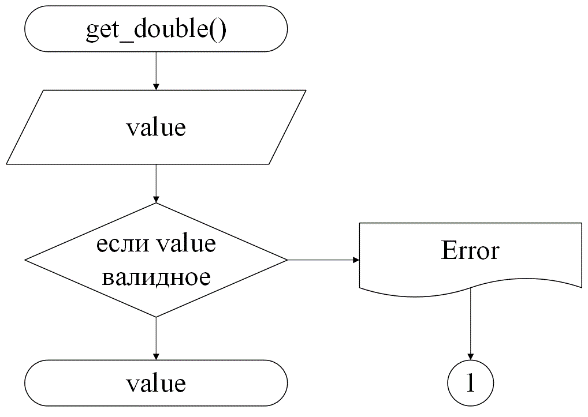
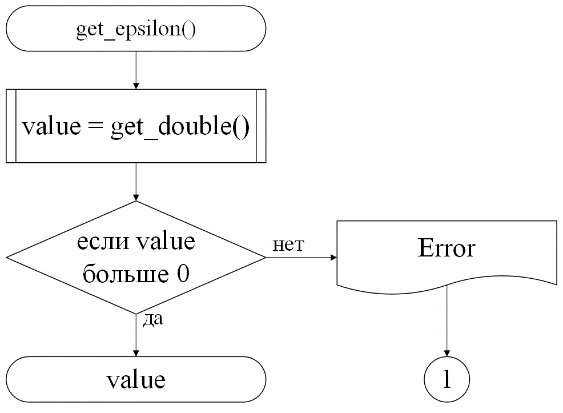


Рисунок 1 ­ Блок-схема основного алгоритма







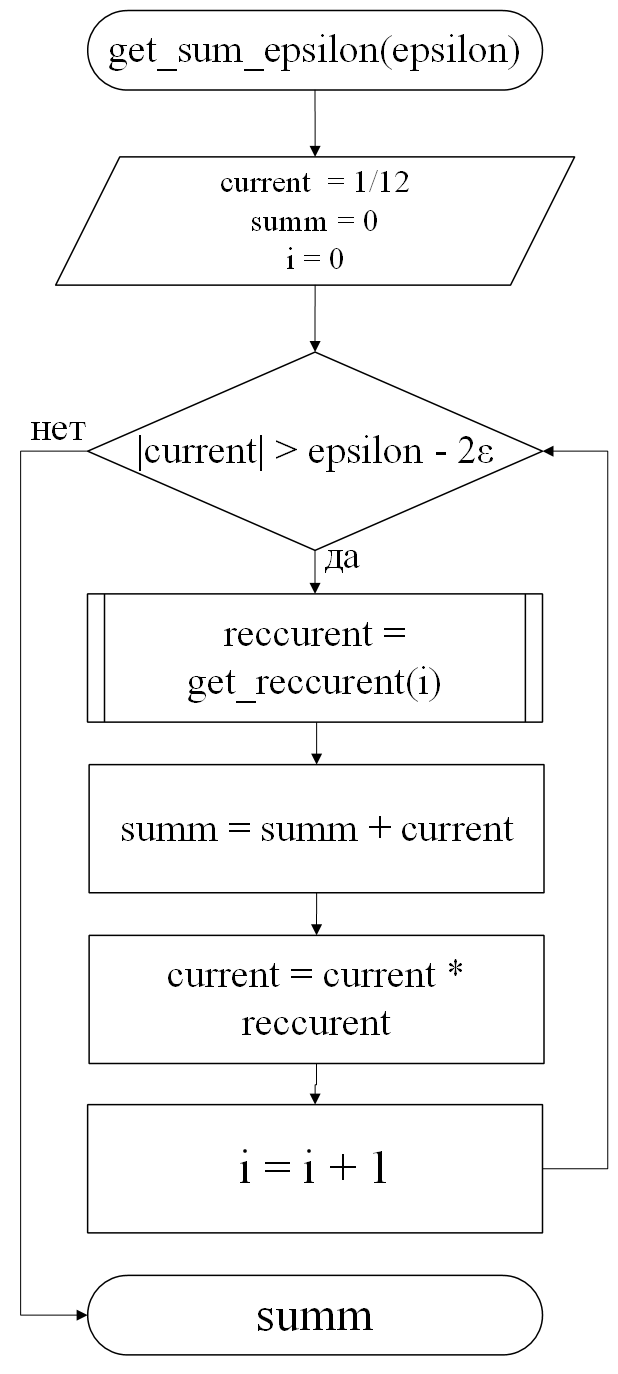


Рисунок 2 – Блок-схема используемых функций

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <math.h>

#include <float.h>

#include <locale.h>

/\*\*

\* **@brief** Провереяет вводимое значение.

\* **@return** Если 1 выводит введенное значение, если 0 выводит ошибку.

\*/

int get\_int(void);

/\*\*

\* **@brief** Проверяет введеное значение на то, что оно больше 0.

\* **@return** Если 1 выводит введенное значение, если 0 выводит ошибку.

\*/

size\_t get\_non\_negative\_int(void);

/\*\*

\* **@brief** Считает сумму членов последовательности.

\* **@param** **count** Количество членов последовательности.

\* **@return** Возвращает сумму членов последовательности.

\*/

double get\_sum(const size\_t count);

/\*\*

\* **@brief** Считает член последовательности.

\* **@param** **k** Меняющийся параметр в функции.

\* **@return** Возвращает высчитанный член последовательности.

\*/

double get\_reccurent(const size\_t k);

/\*\*

\* **@brief** Провереяет вводимое значение.

\* **@return** Если 1 выводит введенное значение, если 0 выводит ошибку.

\*/

double get\_double(void);

/\*\*

\* **@brief** Проверяет введеное значение на то, что оно больше 0.

\* **@return** Если 1 выводит введенное значение, если 0 выводит ошибку.

\*/

double get\_epsilon(void);

/\*\*

\* **@brief** Считает сумму членов последовательности.

\* **@param** **epsilon** Число е.

\* **@return** Возвращает сумму членов последовательности.

\*/

double get\_sum\_epsilon(const double epsilon);

/\*\*

\* **@brief** Точка входа в программу.

\* **@return** 0 В случае успеха

\*/

int main(void) {

setlocale(LC\_ALL, "Russian");

puts("Количество членов последовательности:");

size\_t count = get\_non\_negative\_int();

printf("Сумма последовательности из %zu элементов равна %.10lf\n", count, get\_sum(count));

puts("Введите точность вычисления:");

double epsilon = get\_epsilon();

printf("Сумма последовательности с точностью %lf равна %.10lf\n", epsilon, get\_sum\_epsilon(epsilon));

return EXIT\_SUCCESS;

}

int get\_int(void) {

int value = 0;

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

if (result != 1) {

errno = EDOM;

perror("Неверное введеное значение!");

exit(EXIT\_FAILURE);

}

return value;

}

size\_t get\_non\_negative\_int(void) {

int value = get\_int();

if (value <= 0) {

errno = EDOM;

perror("Неверное введеное значение!");

exit(EXIT\_FAILURE);

}

return (size\_t)value;

}

double get\_sum(size\_t count) {

double current = 1.0 / 12;

double summ = current;

for (size\_t i = 0; i < count; ++i) {

current \*= get\_reccurent(i);

summ += current;

}

return summ;

}

double get\_reccurent(size\_t k) {

return ( - 1.0) / ((k + 3) \* (k + 4));

}

double get\_epsilon(void) {

double epsilon = get\_double();

if (epsilon <= 0.0) {

puts("Error input!");

exit(EXIT\_FAILURE);

};

return epsilon;

}

double get\_double(void) {

double value = 0.0;

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

if (result != 1) {

errno = EIO;

perror("Неверное введеное значение!");

exit(EXIT\_FAILURE);

}

return value;

}

double get\_sum\_epsilon(const double epsilon) {

double summ = 0;

double current = 1.0/12;

for (size\_t i = 0; fabs(current) > epsilon - DBL\_EPSILON; ++i) {

summ += current;

current \*= get\_reccurent(i);

}

return summ;

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 3).

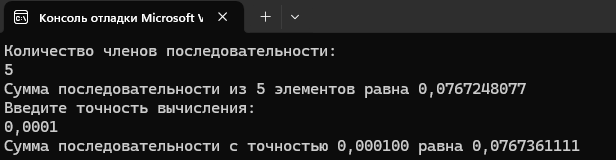


Рисунок 3 – Результаты выполнения программы

1. Выполнение тестовых примеров

В программе PyCharm выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 4). Выполнены тестовые примеры на неправильные входные данные (Рисунок 5).

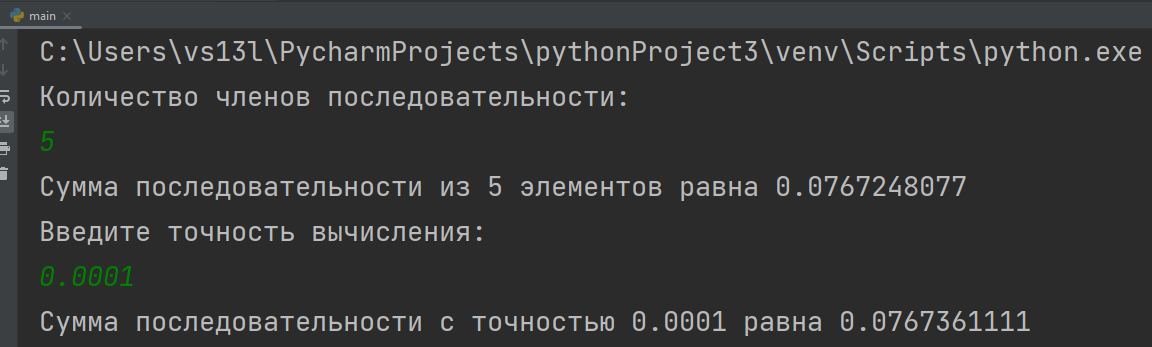
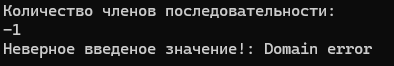
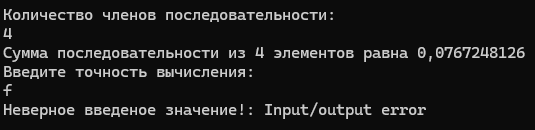


Рисунок  – Результат расчета







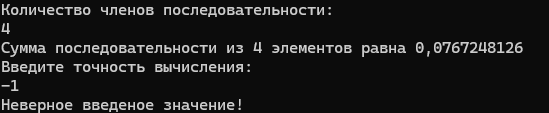


Рисунок 5 – Результат выполнения тестовых примеров с неправильными входными данными

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

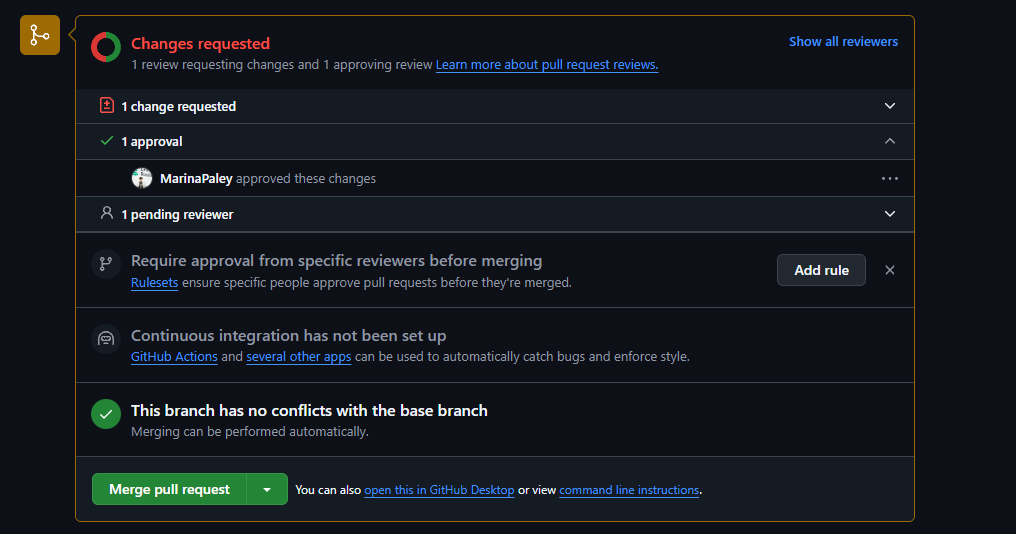


Рисунок 6 – Одобренный код по заданию 3-2