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 №1

По дисциплине «Языки программирования»

ВАРИАНТ 11

Выполнил: ст. гр. ТКИ-141

Колчихина Виктория Андреевна

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

Москва 2024

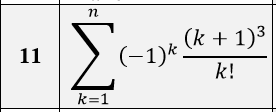
1. Формулировка задания

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  – Исходные данные



1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций для проверки числа GetInt(void) и GetNonNegativeInt(void), GetEpsilon(void) и GetDouble(void)представлены ниже (Рисунок 2, Рисунок 3). Блок-схема функции для рекуррентного члена GetReccurent(const size\_t k) представлена ниже (Рисунок 4). Блок-схемы функций для подсчёта суммы GetSumEpsilon(const double epsilon) и GetSumCount(const size\_t count)представлены ниже (Рисунок 5).

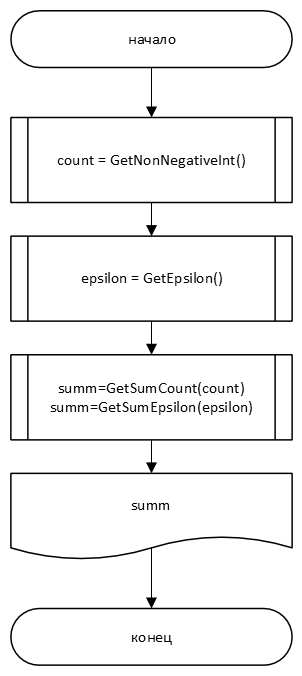
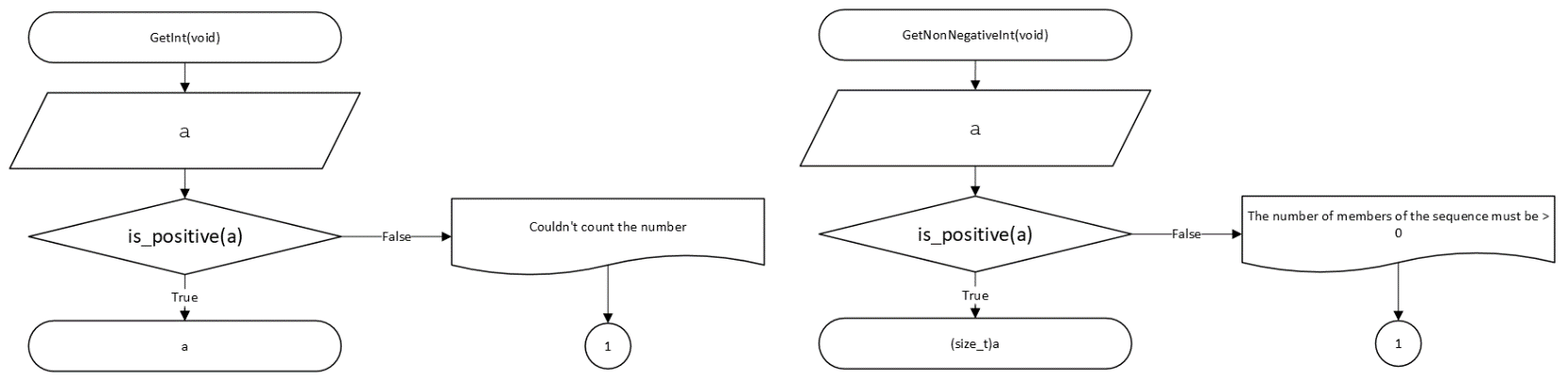


Рисунок  ­ Блок-схема основного алгоритма

Рисунок  – Блок-схемы используемых функций проверки числа на положительность(GetInt(void) и GetNonNegativeInt(void))

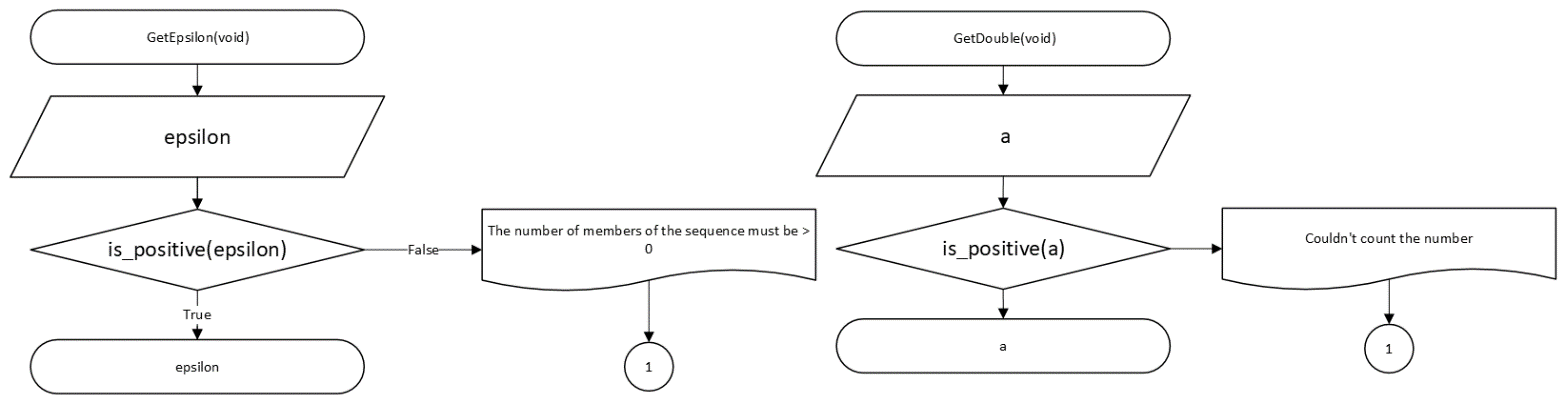


Рисунок 3 – Блок-схемы используемых функций проверки числа на положительность(GetEpsilon(void) и GetDouble(void))

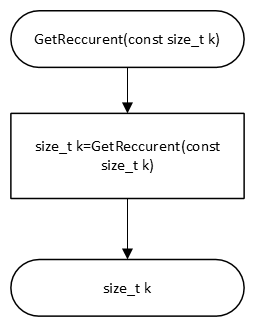


Рисунок 4 –  Блок-схема используемой функции для рекуррентного члена (GetReccurent(const size\_t k))

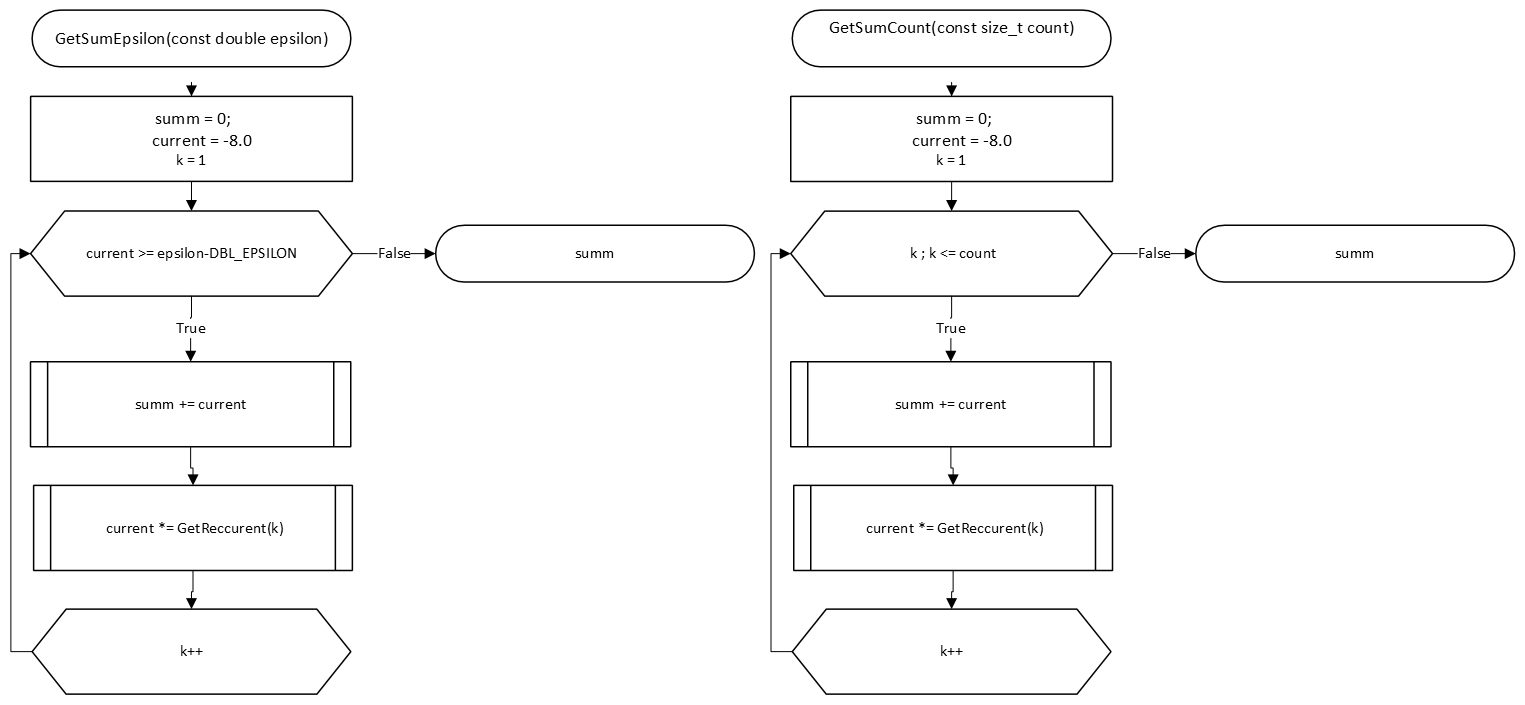


Рисунок 5 – Блок-схемы используемых функций подсчёта суммы (GetSumEpsilon(const double epsilon) и GetSumCount(const size\_t count)

1. Текст программы на языке C

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#include <float.h>

#include <math.h>

/\*\*

\*@brief считывает целочисленное значение.

\*@return целое значение.

\*/

int GetInt(void);

/\*\*

\*@brief считывает неотрицательное целое число.

\*@return неотрицательное целое значение.

\*/

size\_t GetNonNegativeInt(void);

/\*\*

\*@brief вычисляет сумму на основе заданного значения count.

\*@param count количество слагаемых, используемых при расчете суммы.

\*@return вычисленную сумму в виде double.

\*/

double GetSumCount(const size\_t count);

/\*\*

\*@brief вычисляет значение на основе рекуррентного отношения.

\*@param k - параметр, используемый при повторном вычислении.

\*@return ввычисленное значение в виде double.

\*/

double GetReccurent(const size\_t k);

/\*\*

\*@brief считывает число с плавающей запятой.

\*@return число типа double .

\*/

double GetDouble(void);

/\*\*

\*@brief считывает положительное число с плавающей запятой.

\*@return положительное число типа double .

\*/

double GetEpsilon(void);

/\*\*

\*@brief вычисляет сумму на основе значения epsilon.

\*@param epsilon - пороговое значение, используемое для остановки вычисления суммы.

\*@return вычисленную сумму на основе указанного значения epsilon.

\*/

double GetSumEpsilon(const double epsilon);

/\*\*

\* @brief точка входа в программу

\* @return возращает summ в случае успеха

\*/

int main(void)

{

puts("Количество членов последовательности");

size\_t count = GetNonNegativeInt();

printf("Сумма последовательности из %zu элементов равна %.18lf", count, GetSumCount(count));

puts("\nВведите точность вычиления");

double epsilon = GetEpsilon();

printf("Сумма последовательности с точностью %lf равна %.18lf", epsilon, GetSumEpsilon(epsilon));

return EXIT\_SUCCESS;

}

int GetInt(void)

{

int a = 0;

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

if (result != 1)

{

errno = EIO;

perror("Не удалось считать число");

exit(EXIT\_FAILURE);

}

return a;

}

size\_t GetNonNegativeInt(void)

{

int a = GetInt();

if (a <= 0)

{

errno = EDOM;

perror("Количество членов последовательности должно быть > 0");

exit(EXIT\_FAILURE);

}

return (size\_t)a;

}

double GetSumCount(const size\_t count)

{

double summ = 0;

double current = -8.0;

for(size\_t k = 1; k <= count; ++k)

{

summ += current;

current \*= GetReccurent(k);

}

return summ;

}

double GetReccurent(const size\_t k)

{

return pow(-1,k)\*((pow(k+1,3))/static\_cast<double>(k));

}

double GetEpsilon(void)

{

double epsilon = GetDouble();

if (epsilon <= DBL\_EPSILON)

{

errno = EDOM;

perror("Точность должна быть > 0");

exit(EXIT\_FAILURE);

}

return epsilon;

}

double GetDouble(void)

{

double a = 0.0;

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

if (result != 1)

{

errno = EIO;

perror("Не удалось считать число");

exit(EXIT\_FAILURE);

}

return a;

}

double GetSumEpsilon(const double epsilon)

{

double summ = 0;

double current = -8.0;

int k = 1;

while(fabs(current) >= epsilon-DBL\_EPSILON)

{

summ += current;

current \*= GetReccurent(k);

++k;

}

return summ;

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 6).

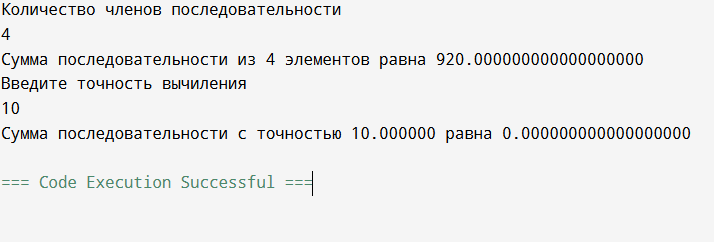


Рисунок 6 – Результаты выполнения программы

1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 7).

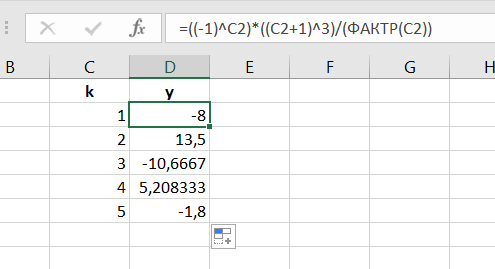


Рисунок 7 – Результат расчета выражения

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

