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 №2.1

По дисциплине «Языки программирования»

ВАРИАНТ 4

Выполнил: ст. гр. ТКИ-141

Бышовец Михаил Александрович

Проверил: к.т.н., доц. Васильева М. А.

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

Москва 2024

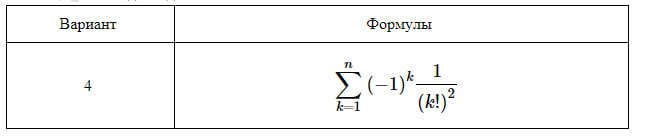
1. Формулировка задания

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.

Ниже представлены исходные данные (Таблица 1).

Таблица  – Исходные данные

1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций ввода finput и dinput (Рисунок 2), функций проверки числа на положительность is\_positive, is\_positive\_d (Рисунок 3), функий подсчёта сумм get\_summ (Рисунок 4) и get\_summ\_e (Рисунок 5) и функции подсчёта следующего элемента числовой последовательности get\_next\_element (Рисунок 6).

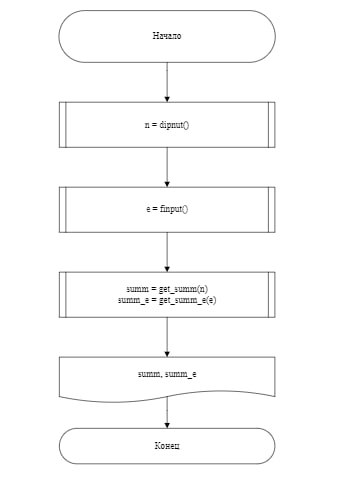


Рисунок 1 ­ Блок-схема основного алгоритма

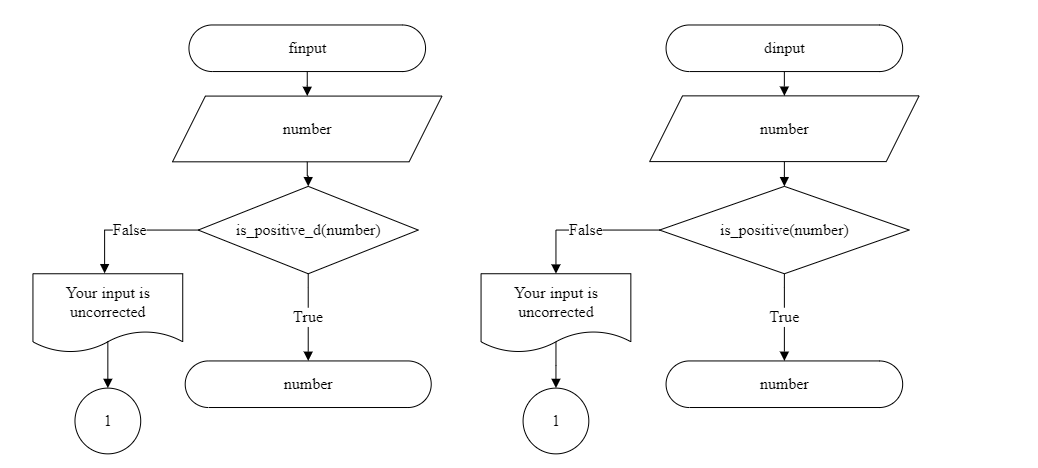


Рисунок  – Блок-схемы используемых функций ввода finput и dinput

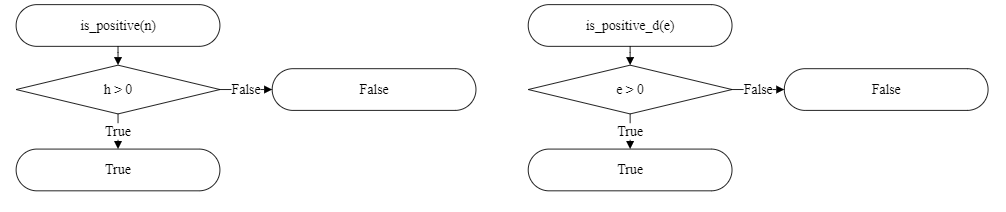


Рисунок 3 – Блок-схемы используемых функций is\_positive, is\_positive\_d

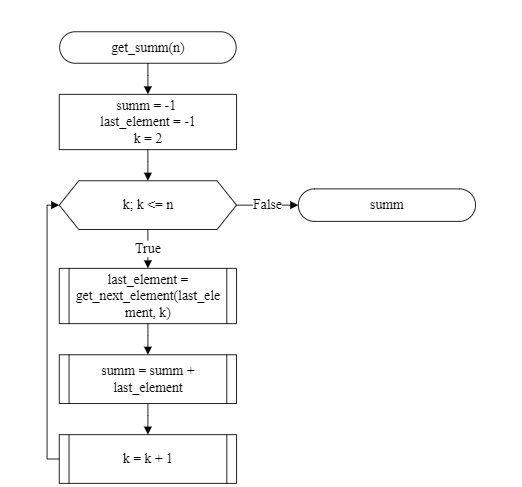


Рисунок 4 - Блок-схема используемой функции подсчёта суммы get\_summ

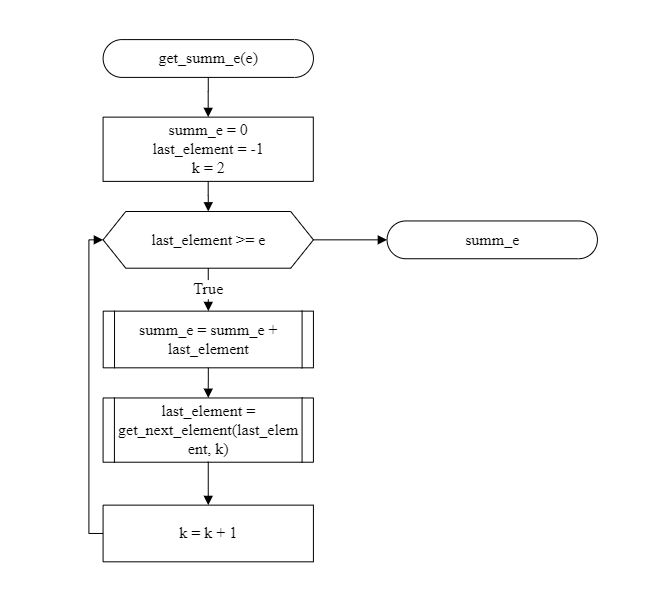


Рисунок 5 - Блок-схема используемой функции подсчёта суммы числе больше заданного числа e get\_summ\_e

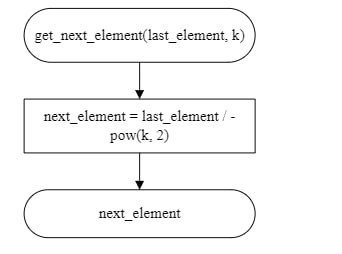


Рисунок 6 - Блок-схема используемой функции подсчёта следующего элемента числовой последовательности get\_next\_element

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <float.h>

#include <stdbool.h>

/\*\*

@brief dinput считывает данные типа int, вводимые пользователем

@return возвращает значение, введённое пользователем

\*/

int dinput(void);

/\*\*

@brief finput считывает значения типа double, вводимые пользователем

@return возвращает значение, введённое пользователем

\*/

double finput(void);

/\*\*

@brief get\_next\_element считает следующий элемент последовательности

@param last\_element последний элемент последовательности

@return возвращает следующий элемент последовательности

\*/

double get\_next\_element(const double last\_element, const int k);

/\*\*

@brief get\_summ считает сумму для первых n элементов

@param n параметр n

@return возвращает сумму первых n элементов

\*/

double get\_summ(const int n);

/\*\*

@brief get\_summ\_e считает сумму элементов не меньше e

@param e параметр e

@return возвращает сумму элементов не меньших e

\*/

double get\_summ\_e(const double e);

/\*\*

@brief is\_positive проверяет положителен ли n

@param n параметр n

@reеurn возвращает true, если n положителен и false, если равен или меньше 0

\*/

bool is\_positive(const int n);

/\*\*

@brief is\_positive проверяет положителен ли e

@param e параметр e

@reеurn возвращает true, если e положителен и false, если равен или меньше 0

\*/

bool is\_positive\_d(const double e);

/\*\*

@brief main выводит значения сумм

@return возвращает 0 в случае успеха

\*/

int main(void) {

puts("Enter the n:");  
  
int n = dinput();  
  
puts("Enter the e");  
  
double e = finput();  
  
printf("%lf\n %lf", get\_summ(n), get\_summ\_e(n, e));  
  
return 0;

}

double get\_summ(const int n) {

double summ = -1;  
  
double last\_element = -1;  
  
for (int k = 2; k <= n; k++)  
{  
  
 last\_element = get\_next\_element(last\_element, k);  
  
 summ += last\_element;  
}  
  
return summ;

}

double get\_next\_element(const double last\_element, const int k) {

return last\_element / -pow(k, 2);

}

double get\_summ\_e(const double e) {

double summ\_e = 0;  
  
double last\_element = -1;  
  
int k = 2;  
  
while(fabs(last element) >= e + DBL\_EPSILON)  
{  
 summ\_e += last\_element;  
 last\_element = get\_next\_element(last\_element, k);  
 k += 1;  
}  
  
return summ\_e;

}

int dinput(void) {

int number = 0;  
  
if (scanf\_s("%d", &number) != 1 || !is\_positive(number))  
{  
  
 puts("Your input is uncorrected");  
  
 exit(EXIT\_FAILURE);  
}  
  
return number;

}

double finput(void) {

double number = 0;  
  
if (scanf\_s("%lf", &number) != 1 || !is\_positive\_d(number))  
{  
  
 puts("Your input is uncorrected");  
  
 exit(EXIT\_FAILURE);  
}  
  
return number;

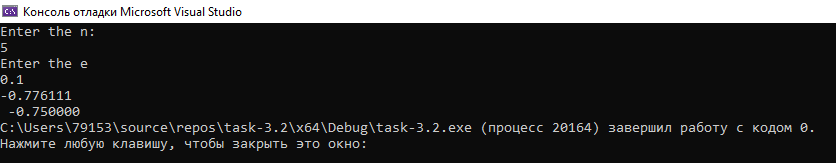
}

bool is\_positive(const int n) { return n > 0; }

bool is\_positive\_d(const double e) { return e > DBL\_EPSILON; }

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 7).

Рисунок 7 - результаты выполнения программы

1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 8, Рисунок 9).

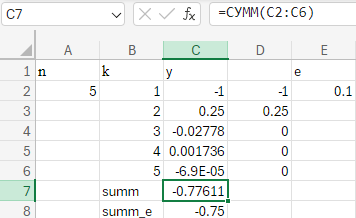


Рисунок 8 - результат выполнения тестового примера summ

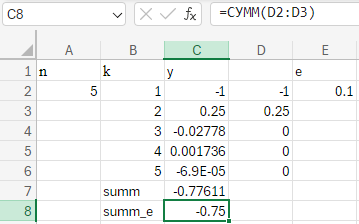


Рисунок 9 - результат выполнения тестового примера summ\_e

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

Ниже представлено доказательство того, что задание 3-2 было принято (Рисунок 10)



Рисунок 10 – Approve задания 3-2