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 6

Выполнил: ст. гр. ТКИ-142

Дохолян Герман Оганесович

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 |
| 6 |  |

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 (2, 3, 4, 5, 6, 7, 8).

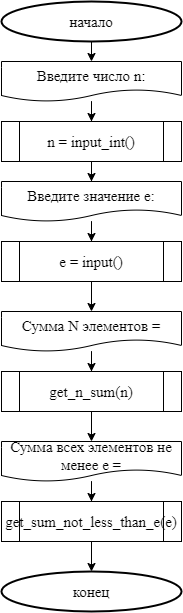


Рисунок  ­ Блок-схема основного алгоритма

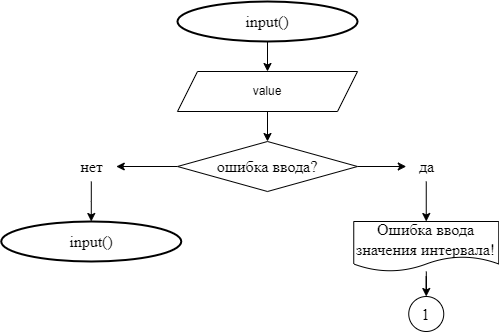


Рисунок 2 – Блок-схема функции input()

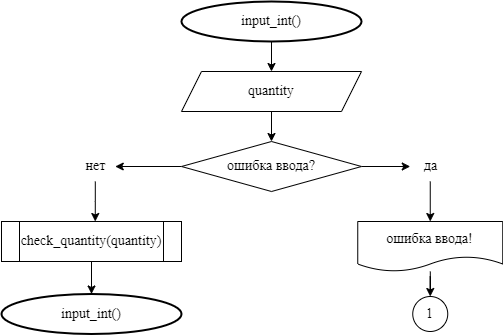


Рисунок 3 – Блок-схема функции input\_int()

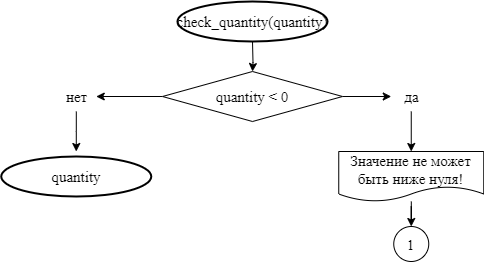


Рисунок 4 – Блок-схема функций check\_quantity(quantity)

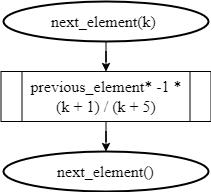


Рисунок 5 – Блок-схема функции next\_element(k)

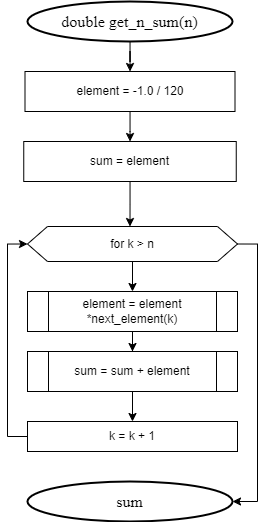


Рисунок 6 – Блок-схема функции get\_n\_sum(n)

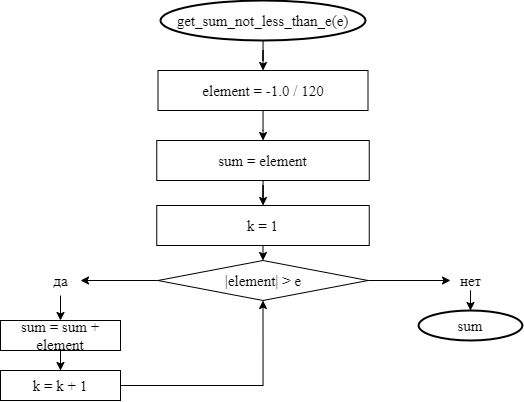


Рисунок 7 – Блок-схема функции get\_sum\_not\_less\_than\_f(e)

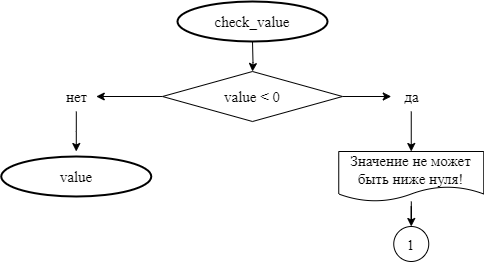


Рисунок 8 – Блок-схема функции check\_value(value)

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 input\_int(void);

/\*\*

\* @brief Проверяет неотрицательность введенного значения

\* @param quantity введенное значение

\* @return Возвращает ошибку, если значение меньше нуля

\*/

void check\_quantity(const int quantity);

/\*\*

\* @brief Проверяет неотрицательность вещественного значения

\* @param value введеное значение

\* @return Возвращает ошибку, если введеное значение не положительно

\*/

void check\_value(const double value);

/\*\*

\* @brief Рассчитывает значение следующего элемента последовательности

\* @param k текущий индекс последовательности

\* @return Возвращает значение следующего элемента последовательности

\*/

double next\_element(const int k);

/\*\*

\* @brief Рассчитывает значение суммы n членов

\* @param n Количество членов последовательности

\* @return Возвращает значение суммы n членов последовательности

\*/

double get\_n\_sum(const int n);

/\*\*

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

\* @param e Заданное число

\* @return Возвращает значение суммы всех членов последовательности

\*/

double get\_sum\_not\_less\_than\_e(const double e);

/\*\*

\* @brief Точка входа в программу

\* @return В случае успеха, возвращает 0

\*/

int main(void)

{

printf("Введите число n:\n");

const int n = input\_int();

printf("Введите значение e:\n");

const double e = input();

printf("Сумма N элементов = %.7lf\n", get\_n\_sum(n));

printf("Сумма всех элементов не менее e = %.3lf\n", get\_sum\_not\_less\_than\_e(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);

}

check\_value(value);

return value;

}

int input\_int(void)

{

int quantity = 0;

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

if (result != 1)

{

errno = EIO;

perror("ошибка ввода!");

exit(EXIT\_FAILURE);

}

check\_quantity(quantity);

return quantity;

}

void check\_quantity(const int quantity)

{

if (quantity < 0)

{

errno = EIO;

perror("Значение не может быть ниже нуля!");

exit(EXIT\_FAILURE);

}

}

void check\_value(const double value)

{

if (value < 0)

{

errno = EIO;

perror("Значение не может быть ниже нуля!");

exit(EXIT\_FAILURE);

}

}

double next\_element(const int k)

{

return -1 \* ((double)k + 1) / ((double)k + 5);

}

double get\_n\_sum(const int n)

{

double element = -1.0 / 120; // Первый член последовательности

double sum = element;

for (int k = 1; k < n; ++k)

{

element \*= next\_element(k);

sum += element;

}

return sum;

}

double get\_sum\_not\_less\_than\_e(const double e)

{

double element = -1.0 / 120; // Первый член последовательности

double sum = 0;

int k = 0;

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

{

sum += element;

k++;

element \*= next\_element(k);

}

return sum;

}

1. Результаты выполнения программы

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

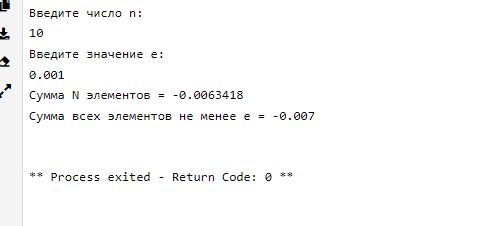


Рисунок 9 – Результаты выполнения программы

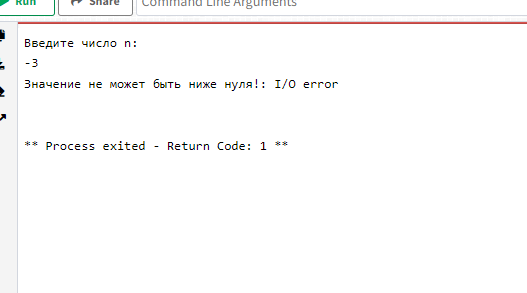


Рисунок 10 – Результаты выполнения программы при вводе отрицательного числа в количество членов последовательности.

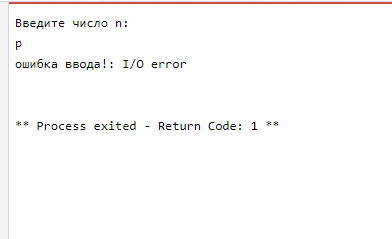


Рисунок 11 – Результаты выполнения программы если введенное число – буква

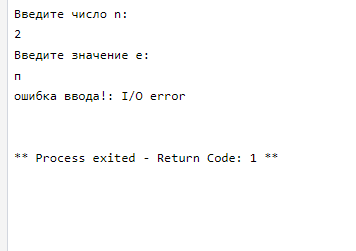


Рисунок 12 – Результаты выполнения программы, когда введенное число точности – буква

1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 13, 14).

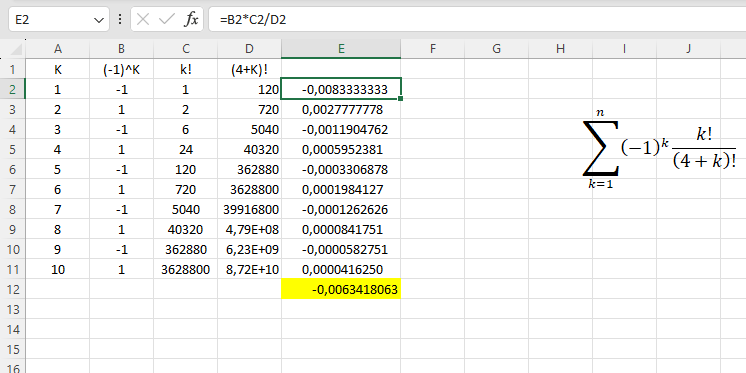


Рисунок 13 – Результат расчета суммы n членов

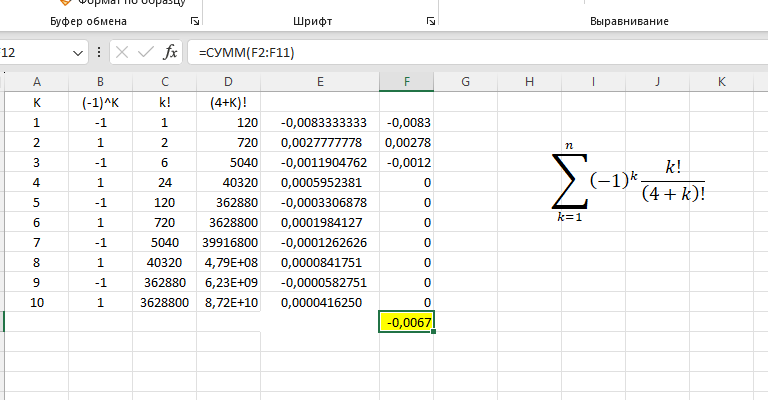


Рисунок 14 – Результат расчета суммы меньше чем e

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

