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 7

Выполнил: ст. гр. ТКИ - 142

Костин Роман Сергеевич

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

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

Москва 2024

Оглавление

[1 РЕшение задания 3.1 3](#_Toc156470676)

[1.1 Формулировка задачи 3](#_Toc156470677)

[1.2 Блок-схема алгоритма 4](#_Toc156470678)

[1.3 Код задания 3.1 7](#_Toc156470679)

[1.4 Решение различных тестовых примеров на C 9](#_Toc156470680)

[1.5 Решение различных тестовых примеров в Excel 10](#_Toc156470681)

[1.6 Подтверждение Approve 3.1 10](#_Toc156470682)

[2 Решение задания 3.2 11](#_Toc156470683)

[2.1 Формулировка задачи 11](#_Toc156470684)

[2.2 Блок-схема алгоритма 12](#_Toc156470685)

[2.3 Код задачи 3.2 15](#_Toc156470686)

[2.4 Решение различных тестовых примеров на C 18](#_Toc156470687)

[2.5 Решение различных тестовых примеров в Excel 18](#_Toc156470688)

[2.6 Подтверждение Approve 18](#_Toc156470689)

1. РЕшение задания 3.1
   1. Формулировка задачи

Протабулировать заданную в таблице функцию. Использовать данные в таблице значения шага и интервала в качестве ввода пользователя для решения тестового примера. При невозможности расчёта функции в конкретной точке выводить её значение и надпись, означающую отсутствие решения. При решении данного задания в MatLab необходимо построить график

Таблица 1 – Условия задания 3.1

|  |  |  |
| --- | --- | --- |
| Вариант | Функция | Константы |
| 7 |  | Интервал и шаг вводит пользователь |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена ниже (Рисунок 1, Рисунок 2, Рисунок 3, Рисунок 4, Рисунок 5)

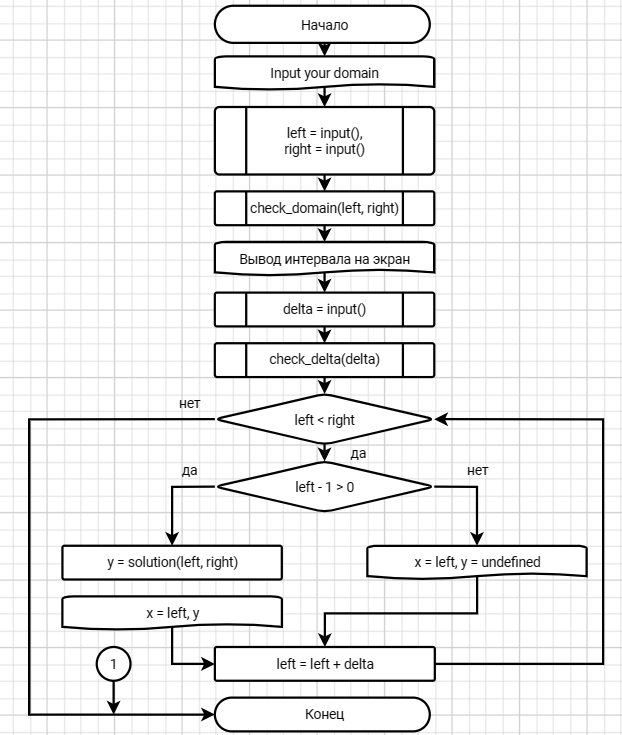


Рисунок 1 – Блок-схема функции main()

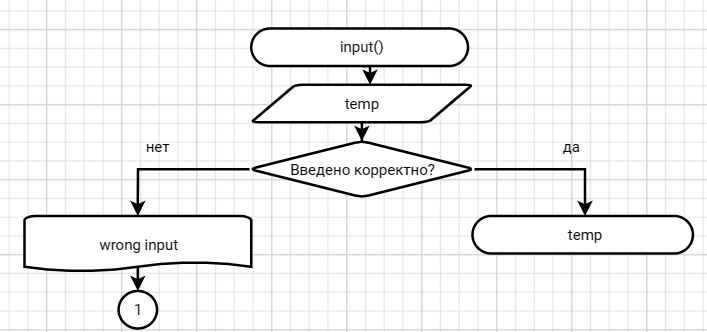


Рисунок 2 – Блок-схема функции input()

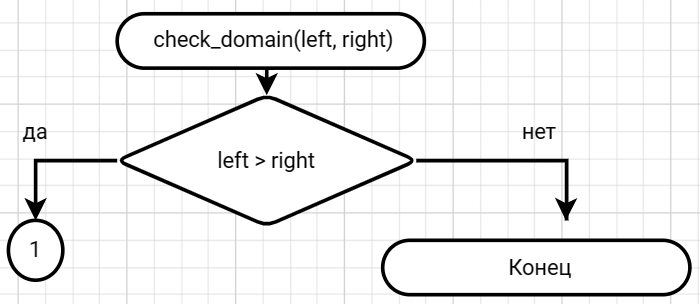


Рисунок 3 – Блок-схема функции check\_domain()

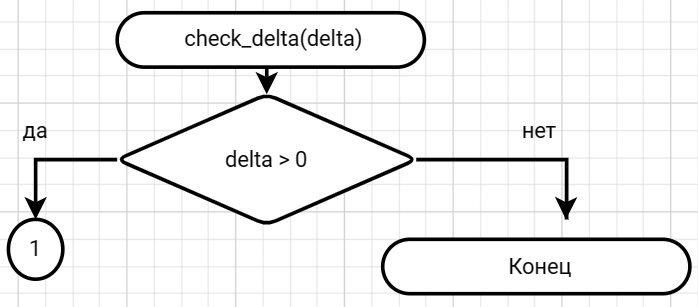


Рисунок 4 – Блок-схема функции check\_delta()

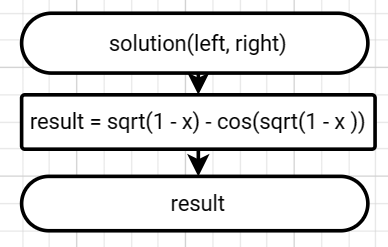


Рисунок 5 – Блок-схема функции solution()

* 1. Код задания 3.1

#include <stdio.h>

#include <math.h>

#include <float.h>

#include <stdlib.h>

#include <errno.h>

/\*\*

\* @brief function is used to get the value from the user

\* @return temp - result of user's input

\*/

double input();

/\*\*

\* @brief function for solution of the equation.

\* @param left - left border of the domain

\* @param right - right border of the domain

\* @return sqrt(1 - i) - cos(sqrt(1 - i)) if function exists in this point, otherwise undefined

\*/

double solution(double left, double right);

/\*\*

\* @brief function checks delta x. If delta x is equal or less than 0 -> abort();

\* @param delta - delta x

\*/

void check\_delta(double delta);

/\*\*

\* @brief fuction checks if domain exists. If value of left border is more than value of the right -> abort();

\* @param left - left border of the domain

\* @param right - rigth border of the domain

\*/

void check\_domain(double left, double right);

/\*\*

\* @brief entrance point into the programm

\* @return 0 if programm works correctly

\*/

int main()

{

printf\_s("input your domain: ");

double left = input();

double right = input();

check\_domain(left, right);

printf\_s("your domain is [%lf, %lf]\n", left, right);

double delta = input();

check\_delta(delta);

for (left; left < right + delta; left += delta)

{

if (left - 1 > DBL\_EPSILON) // -DBL\_EPSILON is less than 0, but i need it to include 0, so DBL\_EPSILON

{

printf\_s("x = %lf ; y = undefined\n", left);

}

else

{

printf\_s("x = %lf ; y = %lf\n", left, solution(left, right));

}

}

return 0;

}

double input()

{

double temp;

int res = scanf\_s("%lf", &temp);

if (res != 1)

{

errno = EIO;

perror("wrong input");

abort();

}

return temp;

}

double solution(double left, double right)

{

{

return sqrt(1 - left) - cos(sqrt(1 - left));

}

}

void check\_domain(double left, double right)

{

if (left - right > -DBL\_EPSILON)

{

printf\_s("Wrong domain!");

abort();

}

}

void check\_delta(double delta)

{

if (delta < DBL\_EPSILON)

{

printf\_s("wrong delta x!");

abort();

}

}

* 1. Решение различных тестовых примеров на C

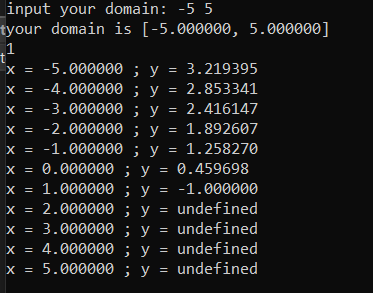


Рисунок 6 – Вывод программы 3.1 на C при корректном вводе значений



Рисунок 7 – Вывод программы 3.1 на C при некорректном вводе интервала

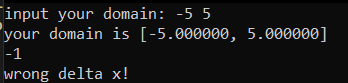


Рисунок 8 – Вывод программы 3.1 на C при некорректном вводе шага

* 1. Решение различных тестовых примеров в Excel

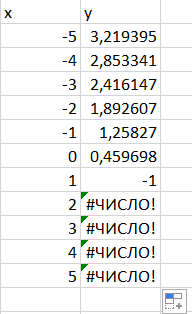


Рисунок 9 – Решение задания 3.1 в Excel

* 1. Подтверждение Approve 3.1

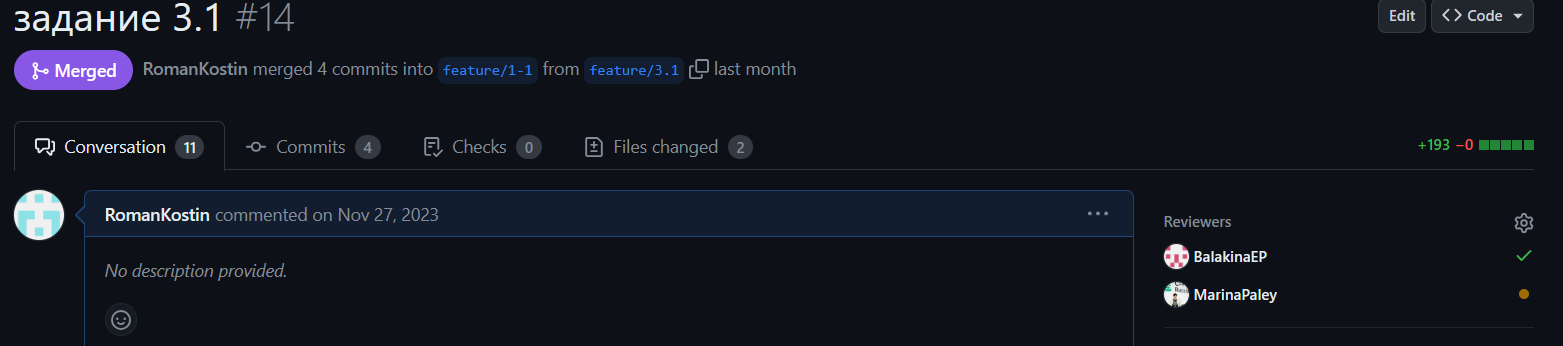


Рисунок 10 – Approve 3.1

1. Решение задания 3.2
   1. Формулировка задачи

Составьте две программы:

* вычислить сумму первых n членов последовательности (k = 1, 2, 3 ..., n).
* вычислить сумму всех членов последовательности, не меньших заданного числа e.

Помните о проверке пользовательского ввода. Все результаты вывести на экран. Отчёт дополнить блок-схемой. При вычислении факториалов рекомендуется отказаться от использования рекурсивных методов.

Таблица  – Условия задачи 3.2

|  |  |
| --- | --- |
| Вариант | Ряд |
| 7 |  |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена ниже ()

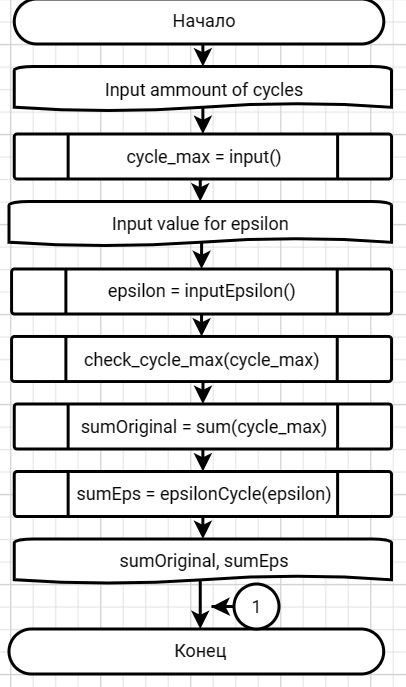


Рисунок 11 – Блок-схема функции main()

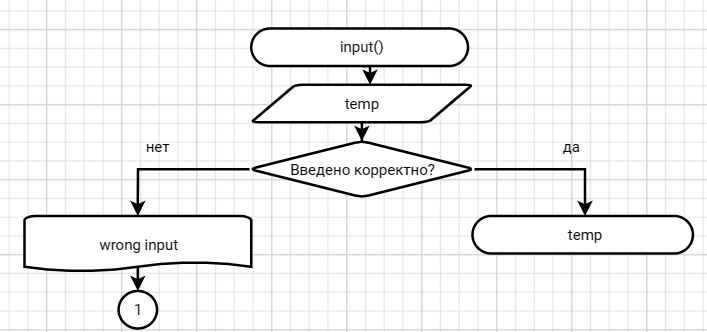


Рисунок 12 – Блок-схема функции input()

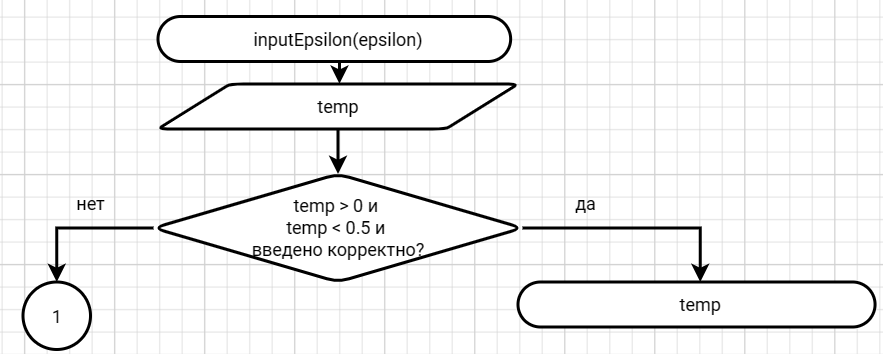


Рисунок 13 – Блок-схема функции inputEpsilon()

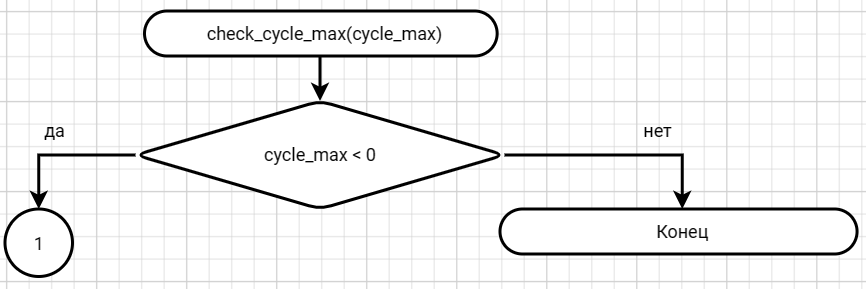


Рисунок 14 – Блок-схема функции check\_cycle\_max()

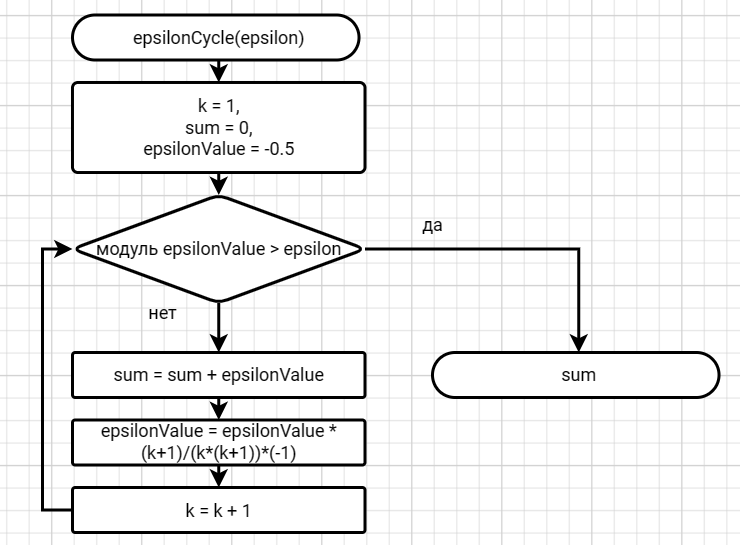


Рисунок 15 – Блок-схема функции epsilonCycle()

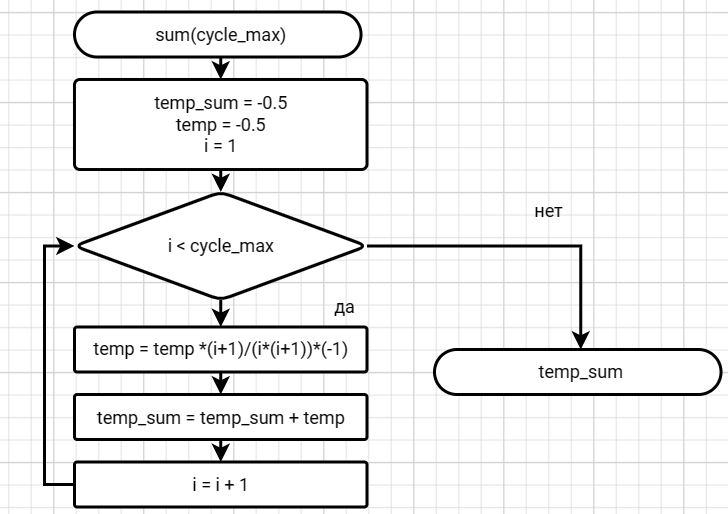


Рисунок 16 – Блок-схема функции sum()

* 1. Код задачи 3.2

#include <stdio.h>

#include <float.h>

#include <stdlib.h>

#include <errno.h>

#include <math.h>

/\*\*

\* @brief function finds ammount of cycles required for the second part of the task (compare element of sum to an epsilon and then print sum up to this value)

\* @param epsilon - value that user chose as an epsilon

\* @return epsilonMaxCycle - max value of cycles required for second part of the task

\*/

double epsilonCycle(double epsilon);

/\*\*

\* @brief function used to get value from user

\* @return temp - result of user's input

\*/

int input();

/\*\*

\* @brief function used to get value as an epsilon from the user

\* @return epsilon - result of user's input

\*/

double inputEpsilon();

/\*\*

\* @brief function checks amount of the cycles. If there is no cycles - abort()

\* @param cycle\_max - amount of cycles that user chooses

\*/

void check(int cycle\_max);

/\*

\* @brief function that summarizes elements of progression. ((k + 1) / (k \* (k + 2))) is the difference between "element" and "element + 1"

\* @param cycle\_max - ammount of cycles

\* @temp - value of the first element of progression

\*/

double sum(int cycle\_max);

/\*\*

\* @brief entry point into the programm

\* @return 0 if programm works correctly

\*/

int main()

{

printf\_s("input ammount of cycles: ");

int cycle\_max = input();

printf\_s("\ninput value for epsilon: ");

double epsilon = inputEpsilon();

check(cycle\_max);

printf\_s("sum of the first task: %lf\nsum with epsilon: %lf", sum(cycle\_max), epsilonCycle(epsilon));

return 0;

}

int input()

{

int temp;

int res = scanf\_s("%d", &temp);

if (res != 1)

{

errno = EIO;

perror("wrong input");

abort();

}

return temp;

}

double inputEpsilon()

{

double epsilon;

int res = scanf\_s("%lf", &epsilon);

if (res != 1 || epsilon - 0.5 > -DBL\_EPSILON || epsilon < DBL\_EPSILON) // 0.5 - is a value of abs(first element of the sum)

{

errno = EIO;

perror("wrong input");

abort();

}

return epsilon;

}

void check(int cycle\_max)

{

if (cycle\_max < 1)

{

printf\_s("error");

abort();

}

}

double sum(int cycle\_max)

{

double temp\_sum = -0.5, // -0.5 - is a value of abs(first element of the sum)

temp = -0.5; // -0.5 - is a value of abs(first element of the sum)

for (int i = 1; i < cycle\_max; i += 1)

{

double k = i;

temp \*= -(k + 1) / (k \* (k + 2));

temp\_sum += temp;

}

return temp\_sum;

}

double epsilonCycle(double epsilon)

{

double k = 1,

sum = 0,

epsilonValue = -0.5; // 0.5 - is a value of abs(first element of the sum)

while (fabs(epsilonValue) - epsilon > -DBL\_EPSILON)

{

sum += epsilonValue;

epsilonValue \*= -(k + 1) / (k \* (k + 2));

k++;

}

return sum;

}

* 1. Решение различных тестовых примеров на C

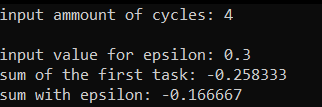


Рисунок 17 – Вывод программы 3.2 на C при корректном вводе значений

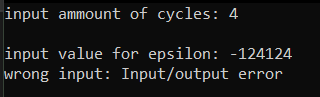


Рисунок 18 – Вывод программы 3.2 на C при некорректном вводе значений

* 1. Решение различных тестовых примеров в Excel



Рисунок 19 – Решение задания 3.2 в Excel

* 1. Подтверждение Approve

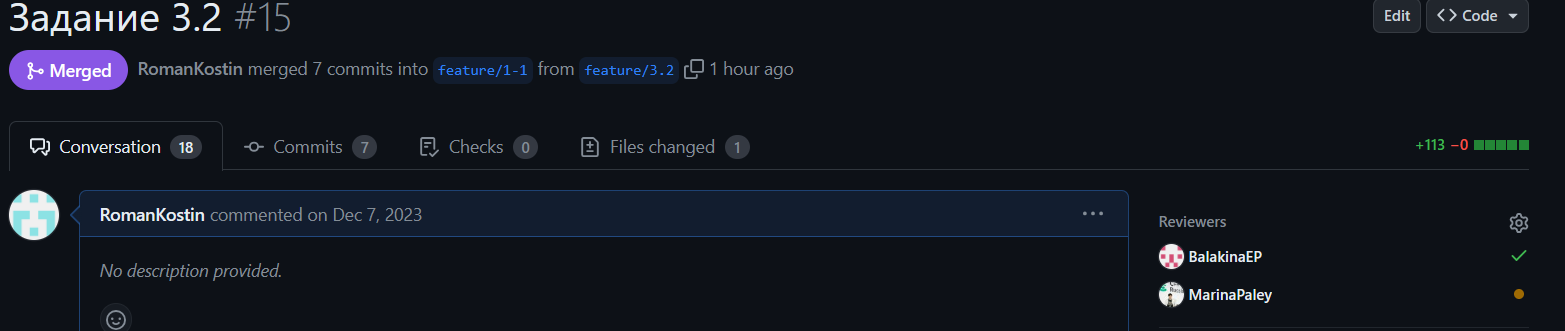


Рисунок 20 – Approve 3.2