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 11

Выполнил: ст. гр. ТКИ-142

Луканкин Вячеслав Сергеевич

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

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

Москва 2023

Оглавление

**Задание 3.11**

1.1 ФОРМУЛИРОВКА ЗАДАНИЯ 1

1.2 БЛОК-СХЕМА АЛГОРИТМА2-5

1.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C6-8

1.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ9

1.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ10

1.6 ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ11

**Задание 3.1 дополнение** 12

1.1 ФОРМУЛИРОВКА ЗАДАНИЯ 12

1.2 БЛОК-СХЕМА АЛГОРИТМА13-14

1.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C15-17

1.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ18

1.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ19

1.6 ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ20

**Задание 3.2** 21

1.1 ФОРМУЛИРОВКА ЗАДАНИЯ 21

1.2 БЛОК-СХЕМА АЛГОРИТМА22-27

1.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C28-32

1.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ33

1.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ34-35

1.6 ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ36

**Задание 3.2 дополнение**37

1.1 ФОРМУЛИРОВКА ЗАДАНИЯ37

1.2 БЛОК-СХЕМА АЛГОРИТМА38-40

1.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C41-43

1.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ44

1.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ45

1.6 ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ46

**Задание 3.3**47

1.1 ФОРМУЛИРОВКА ЗАДАНИЯ47

1.2 БЛОК-СХЕМА АЛГОРИТМА48-51

1.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C52-54

1.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ55

1.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ56

1.6 ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ57

1. зАДАНИЕ 3.1
   1. ФОРМУЛИРОВКА ЗАДАНИЯ

Протабулировать заданную в таблице функцию. Использовать данные в таблице (Таблица 1) 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.

Таблица  – Исходные данные

|  |  |  |
| --- | --- | --- |
| Вариант | Формулы | Константы |
| 11 |  |  |

* 1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1. Рисунок 2. Рисунок 3). Блок-схемы вспомогательных функций представлены ниже (Рисунок 4, Рисунок 5, Рисунок 6).

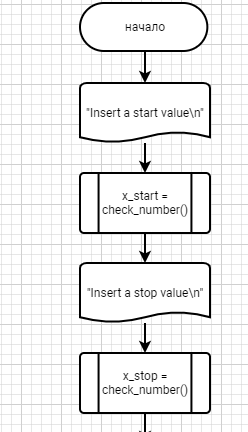


Рисунок  ­ Блок-схема основного алгоритма часть 1

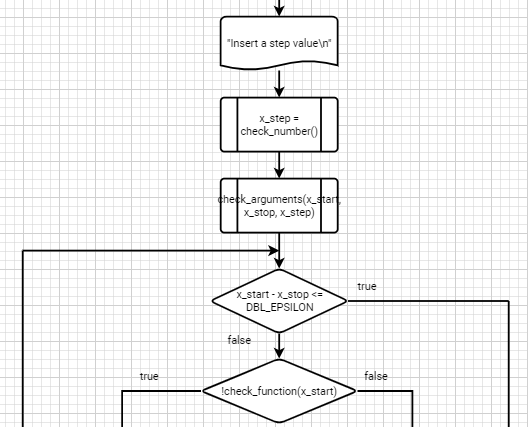


Рисунок  – Блок-схема основного алгоритма часть 2

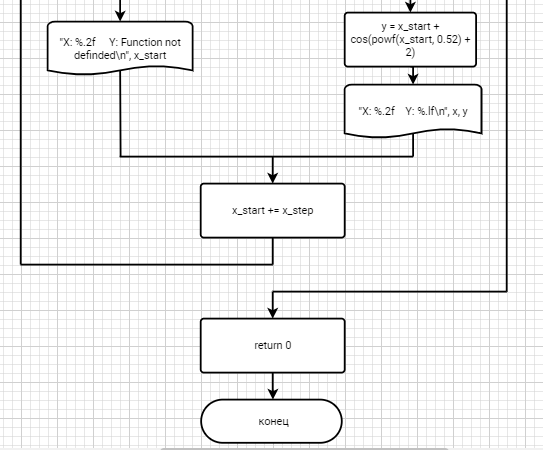


Рисунок  – Блок-схема основного алгоритма часть 3

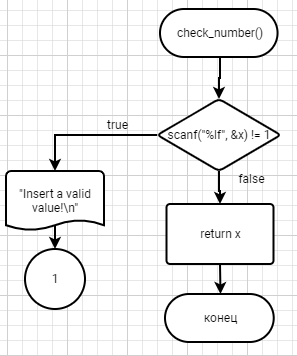


Рисунок 4 – Блок-схема функции check\_number()

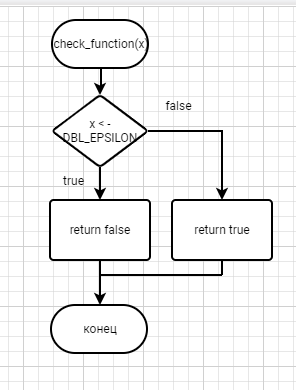


Рисунок  – Блок-схема функции check\_function(x)

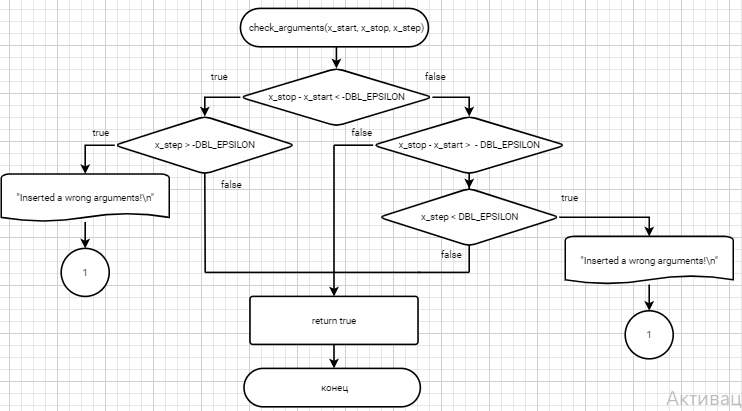


Рисунок 6 – Блок-схема функции check\_arguments(x\_start, x\_stop, x\_step)

* 1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <stdbool.h>

#include <float.h>

/\*\*

\* @brief Проверяет введенное значение как цифру

\* @return Веденная переменная

\*/

double check\_number();

/\*\*

\* @brief Проверяет на определенность функции при введенном аргументе

\* @param x Введенный аргумент

\* @return false Функция не определена

\* @return true Функция определена

\*/

bool check\_function(double x);

/\*\*

\* @brief Проверяет значения на правильность

\* @param x\_start начальное значение

\* @param x\_stop конечное значение

\* @param x\_step значение шага

\* @return true переменные правильно введены

\*/

bool check\_arguments(double x\_start, double x\_stop, double x\_step);

/\*\*

\* @brief Точка входа в программу

\* @return 0 Программа исправна

\*/

int main()

{

puts("Insert a start value\n");

double x\_start = check\_number();

puts("Insert a stop value\n");

double x\_stop = check\_number();

puts("Insert a step number\n");

double x\_step = check\_number();

check\_arguments(x\_start, x\_stop, x\_step);

while (x\_start - x\_stop <= DBL\_EPSILON)

{

if (!check\_function(x\_start))

{

printf("X: %.2lf Y: Function not defined\n", x\_start);

}

else

{

double y = x\_start + cos(powf(x\_start, 0.52)+2);

printf("X: %.2lf Y: %lf\n", x\_start, y);

}

x\_start += x\_step;

}

return 0;

}

double check\_number()

{

double x;

if (scanf("%lf", &x) != 1)

{

puts("Insert valid value!\n");

abort();

}

return x;

}

bool check\_function(double x)

{

if (x < -DBL\_EPSILON)

{

return false;

}

return true;

}

bool check\_arguments(double x\_start, double x\_stop, double x\_step)

{

if (x\_stop - x\_start < DBL\_EPSILON)

{

if (x\_step > -DBL\_EPSILON)

{

puts("Inserted a wrong arguments!");

abort();

}

}

else if (x\_stop - x\_start > -DBL\_EPSILON)

{

if (x\_step < DBL\_EPSILON)

{

puts("Inserted a wrong arguments!");

abort();

}

}

return true;

* 1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 7).

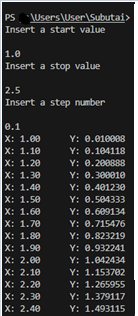


Рисунок  – Результаты выполнения программы

* 1. Выполнение тестовых примеров

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

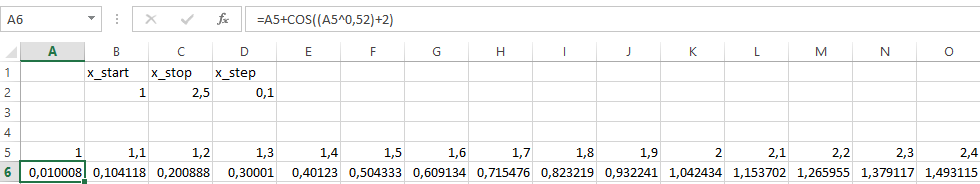
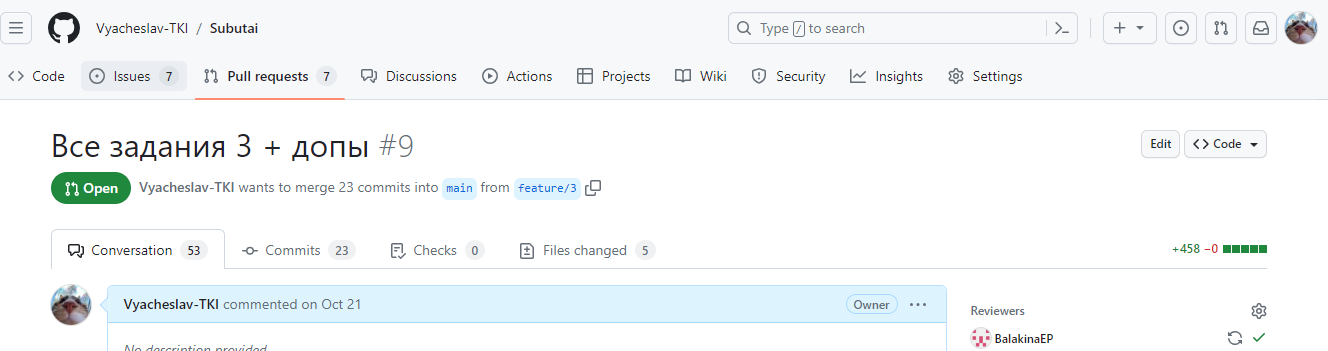


Рисунок  – Результат расчета значений функции

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



1. зАДАНИЕ 3.1 дополнение
   1. Формулировка задания

Определить количество цифр в натуральном числе N, (Таблица 2).

Таблица  –Исходное условие



* 1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 9). Блок-схема вспомогательной функции представлена ниже (Рисунок 10).

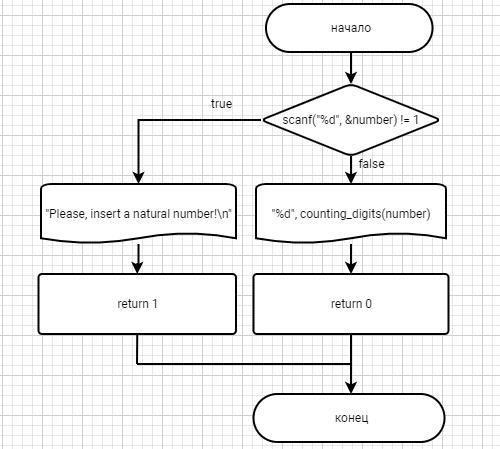


Рисунок  ­ Блок-схема основного алгоритма

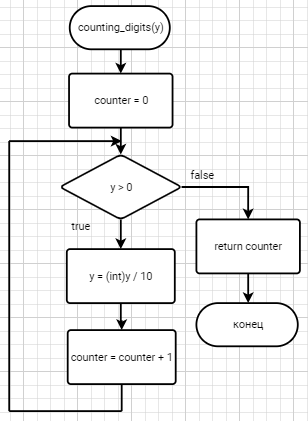


Рисунок  – Блок-схема вспомогательной функции

* 1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

/\*\*

\* @brief Функция подсчитывает разряды целого числа

\* @param y Введенное целочисленное число

\* @return кол-во разрядов

\*/

int counting\_digits(int y);

/\*\*

\* @brief Точка входа в программу

\* @return 0 программа исправна

\* @return 1 программа с ошибкой

\*/

int main(){

int number;

if (scanf("%d", &number) != 1){

puts("Please, insert a natural number!\n");

return 1;

}

printf("%d", counting\_digits(number));

return 0;

}

int counting\_digits(int y){

int counter = 0;

while (y > 0){

y = (int)y / 10;

counter = counter + 1;

}

return counter;

}

* 1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 11).

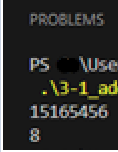


Рисунок  – Результаты выполнения программы

* 1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 12, Рисунок 13).

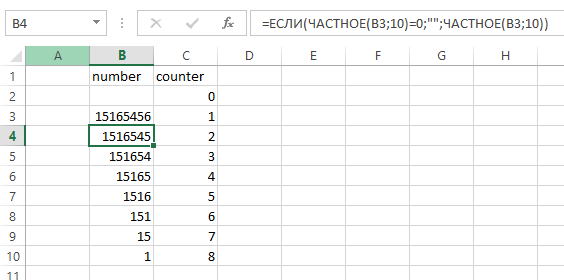


Рисунок  – Результат цикла для переменной number

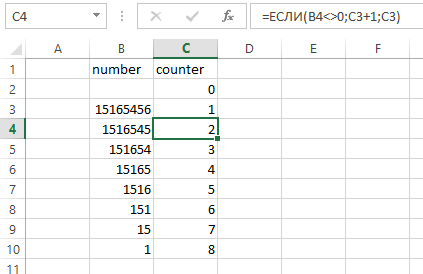
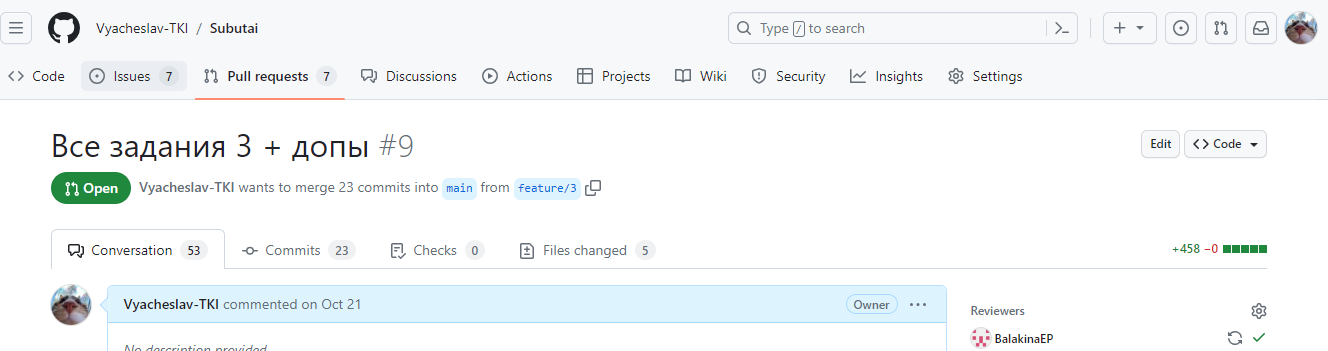


Рисунок  – Результат цикла переменной counter

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



1. зАДАНИЕ 3.2
   1. Формулировка задания

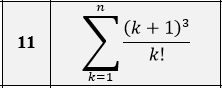
Составьте две программы:

a) вычислить сумму первых *n* членов последовательности (*k* = 1, 2, 3 ..., *n*).

b) 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. Формула суммы представлена ниже (Таблица 1)

Таблица  – Исходные данные



* 1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 14, Рисунок 15). Блок-схемы вспомогательных функций представлены ниже (Рисунок 16, Рисунок 17, Рисунок 18, Рисунок 19, Рисунок 20, Рисунок 21, Рисунок 22, Рисунок 23).

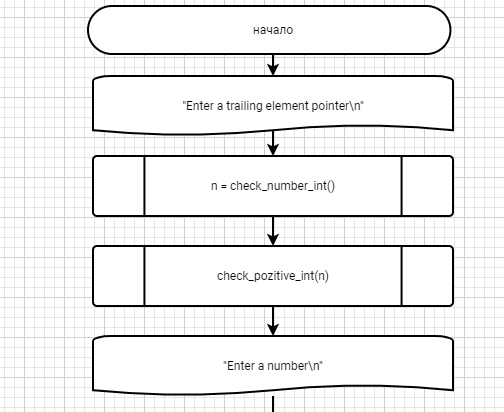


Рисунок  ­ Блок-схема основного алгоритма часть 1

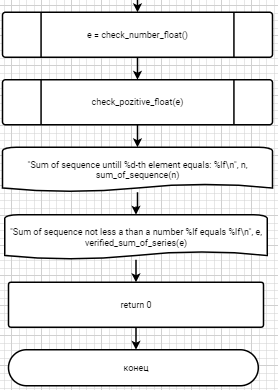


Рисунок  – Блок-схема основного алгоритма часть 2

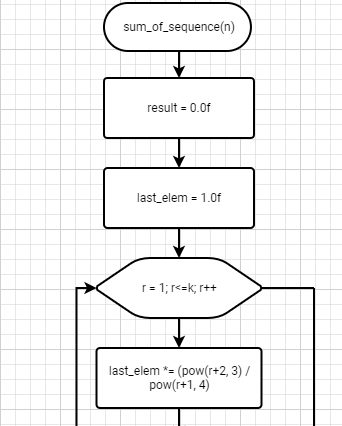


Рисунок 16 – Блок-схема функции sum\_of\_sequence(n) часть 1

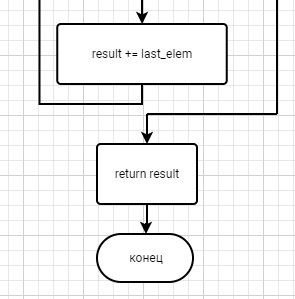


Рисунок 17 – Блок-схема функции sum\_of\_sequence(n) часть 2

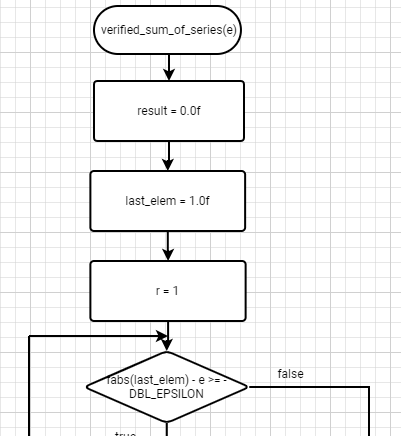


Рисунок 18 – Блок-схема функции verified\_sum\_of\_series(e) часть 1

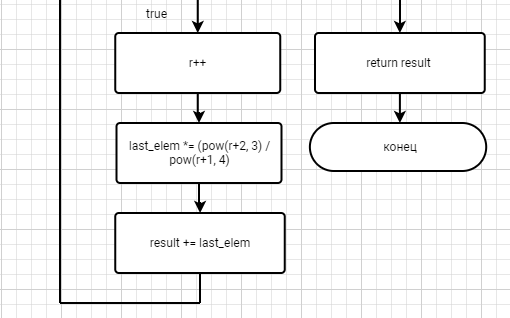


Рисунок 19 – Блок-схема функции verified\_sum\_of\_sequence(e) часть 2

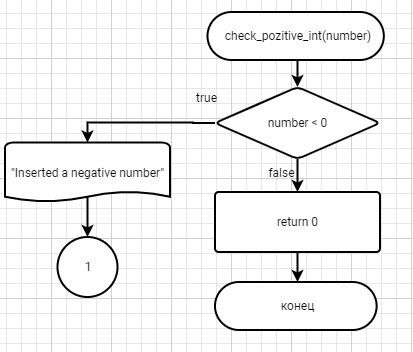


Рисунок  – Блок-схема функции check\_pozitive(number)

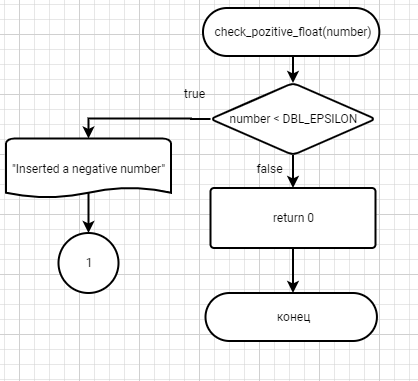


Рисунок 21 – Блок-схема функции check\_pozitive\_float(number)

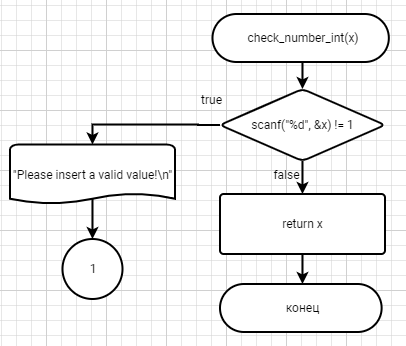


Рисунок  – Блок-схема функции check\_number\_int(x)

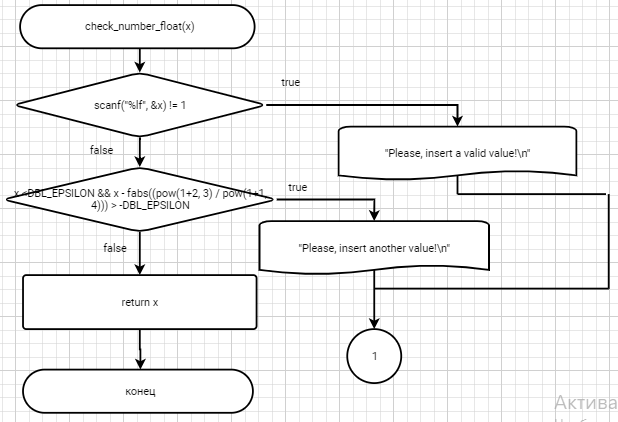


Рисунок 23 – Блок-схема check\_number\_float(x)

* 1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <float.h>

/\*\*

\* @brief Функция проверяет число на положительность

\* @param number введенное число

\* @return 0 число положительное

\*/

double check\_pozitive\_float(double number);

/\*\*

\* @brief Функция проверяет число на положительность

\* @param number введенное число

\* @return 0 число положительное

\*/

int check\_pozitive\_int(int number);

/\*\*

\* @brief Функция проверяет введеный символ на число

\* @return x непосредственно число

\*/

int check\_number\_int();

/\*\*

\* @brief Функция проверяет введеный символ на число

\* @return x непосредственно число

\*/

double check\_number\_float();

/\*\*

\* @brief Функция вычисляет сумму членов заданного ряда

\* @param k Указатель, до какого числа последовательности необходимо подсчитать

\* @return result возвращает сумму до k-ого элемента

\*/

double sum\_of\_sequence(int k);

/\*\*

\* @brief Функция вычисляет сумму членов заданного ряда, не меньших заданного числа

\* @param e Заданное число

\* @return result возвращает сумму, элементы которых не меньше параметра e

\*/

double verified\_sum\_of\_series(double e);

/\*\*

\* @brief Точка входа в программу

\* @return 0 Программа исправно завершилась

\* @return 1 Программа завершилась с ошибкой

\*/

int main(){

puts("Enter a trailing element pointer\n");

int n = check\_number\_int();

check\_pozitive\_int(n);

puts("Enter a number\n");

double e = check\_number\_float();

check\_pozitive\_float(e);

printf("Sum of sequence untill %d-th element equals: %lf\n", n, sum\_of\_sequence(n));

printf("Sum of sequence not less than a number %lf equals: %lf\n", e, verified\_sum\_of\_series(e));

return 0;

}

double sum\_of\_sequence(int k){

double result = 0.0f;

double last\_elem = 1.0f;

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

{

last\_elem \*= (pow(r+2, 3) / pow(r+1, 4));

result += last\_elem;

}

return result;

}

double verified\_sum\_of\_series(double e){

double result = 0.0f;

double last\_elem = 1.0f;

int r = 1;

while(fabs(last\_elem) - e >= -DBL\_EPSILON)

{

r++;

last\_elem \*= (pow(r+2, 3) / pow(r+1, 4));

result += last\_elem;

}

return result;

}

double check\_number\_float(){

double x;

if(scanf("%lf", &x) != 1)

{

puts("Please, insert a valid value!\n");

abort();

}

if (x < DBL\_EPSILON && x - fabs((pow(1+2, 3) / pow(1+1, 4))) > -DBL\_EPSILON)

{

puts("Please, insert another value!\n");

abort();

}

return x;

}

int check\_number\_int(){

int x;

if(scanf("%d", &x) != 1)

{

puts("Please, insert a valid value!\n");

abort();

}

return x;

}

int check\_pozitive\_int(int number)

{

if (number < 0)

{

puts("Inserted a negative number");

abort();

}

return 0;

}

double check\_pozitive\_float(double number)

{

if (number < DBL\_EPSILON)

{

puts("Inserted a negative number");

abort();

}

return 0;

}}

* 1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 24).

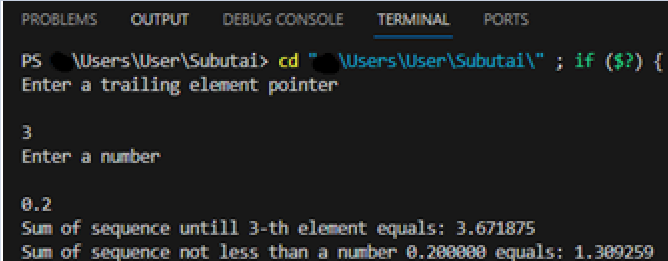


Рисунок  – Результат выполнения программы

* 1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 25, Рисунок 26), (Рисунок 27, Рисунок 28, Рисунок 29).

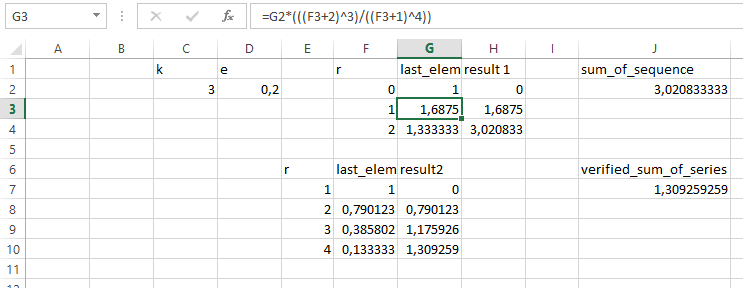


Рисунок  – Изменение переменной last\_elem в функции программы a

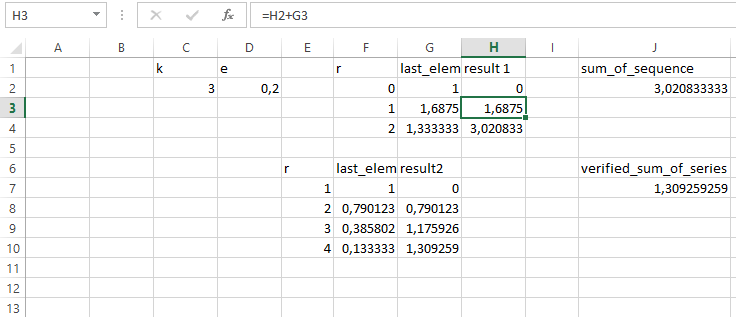


Рисунок  – Результаты расчета переменной result в функции программы a

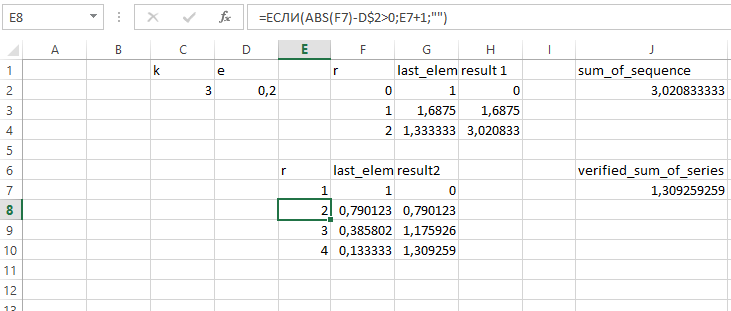


Рисунок  – Изменение переменной r в программе b

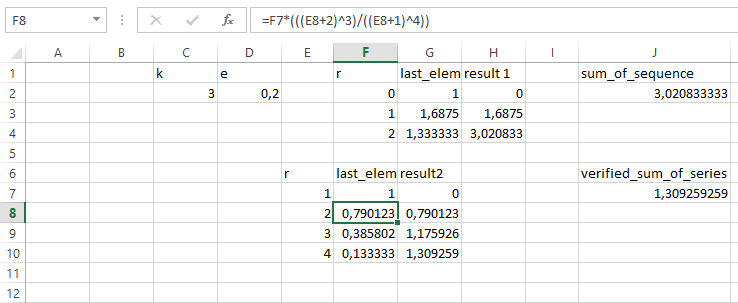


Рисунок  – Изменение переменной last\_elem в программе b

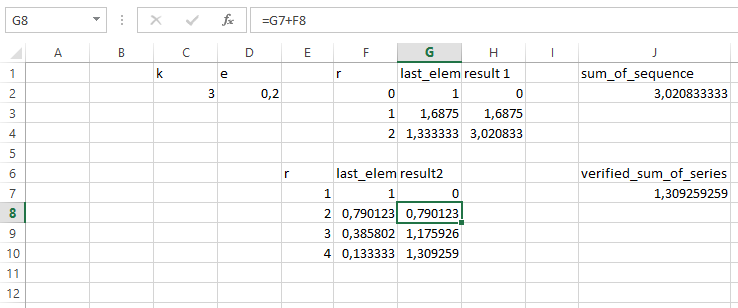
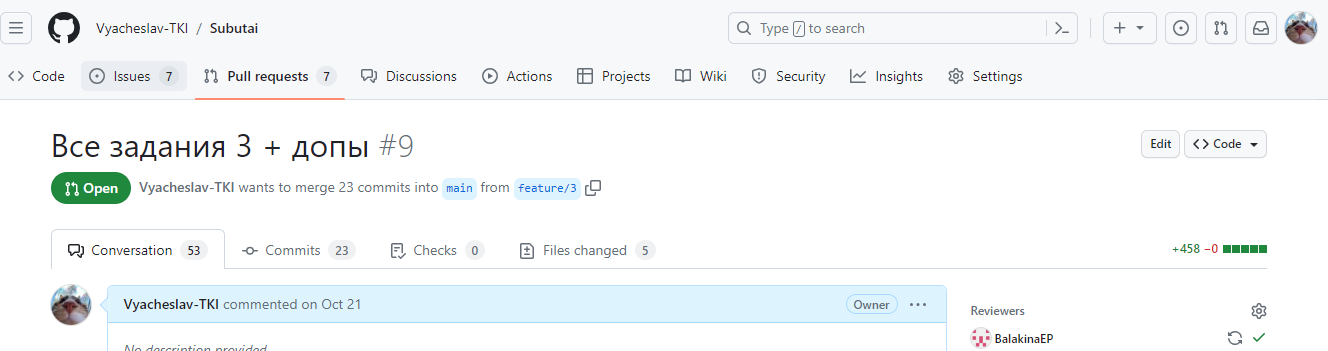


Рисунок  – Результат расчета переменной result в программе b

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



1. задание 3.2 дополнение
   1. Формулировка задания

Найти количество трехзначных чисел, сумма цифр которых равна А, а само число заканчивается цифрой В. А и В задаются (Таблица 4).

Таблица  – Исходные условия



* 1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 30).Блок-схема вспомогательных функций представлены ниже (Рисунок 31, Рисунок 32, Рисунок 33, Рисунок 34, Рисунок 35).

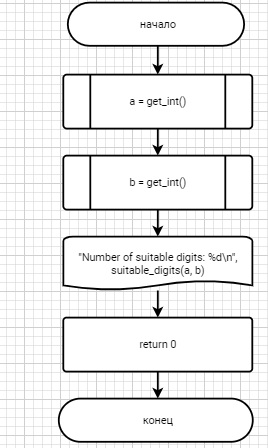


Рисунок  – Блок-схема основной программы

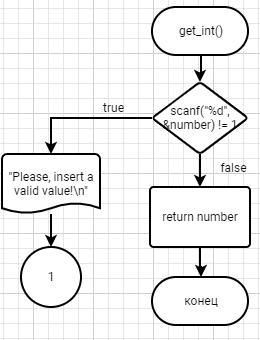


Рисунок  – Блок-схема функции get\_int()

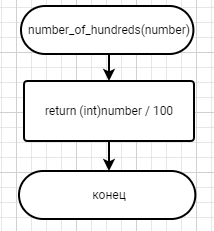


Рисунок 32 – Блок-схемы функции number\_of\_hundreds(number)

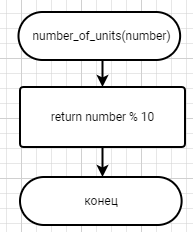


Рисунок 33 – Блок-схема функции number\_of\_units(number)

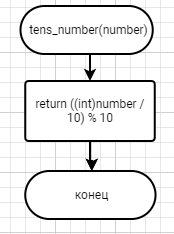


Рисунок 34 – Блок-схема функции tens\_number(number)

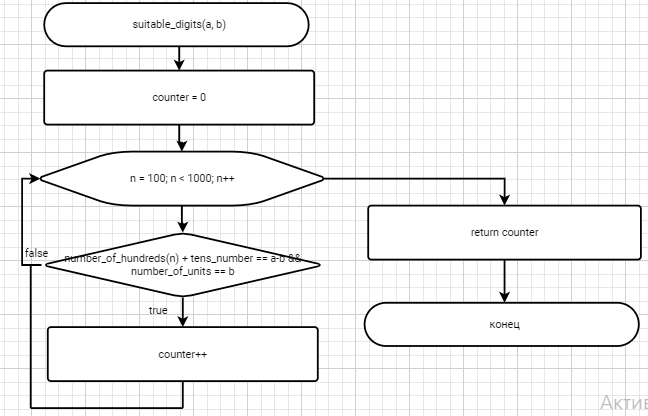


Рисунок 35 – Блок-схема функции suitable\_digits(a, b)

* 1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

/\*\*

\* @brief Функция выделяет число сотен из введенного числа

\* @param number введенное число

\*/

int number\_of\_hundreds(int number);

/\*\*

\* @brief Функция выделяет число десятков из введенного числа

\* @param number введенное число

\*/

int tens\_number(int number);

/\*\*

\* @brief Функция выделяет число единиц из введенного числа

\* @param number введенное число

\*/

int number\_of\_units(int number);

/\*\*

\* @brief Функция вычисляет количество подходящих чисел, где сумма чисел равна а, а заканчивается числом b

\* @param a сумма цифр

\* @param b последняя цифра числа

\*/

int suitable\_digits(int a, int b);

/\*\*

\* @brief функция получает и проверяет целочисленное число

\* @return number введенное число

\*/

int get\_int();

/\*\*

\* @brief точка входа в программу

\* @return 0 программа исправна

\*/

int main()

{

int a = get\_int();

int b = get\_int();

printf("Number of suitable digits: %d\n", suitable\_digits(a, b));

}

int number\_of\_hundreds(int number)

{

return (int)number / 100;

}

int tens\_number(int number)

{

return ((int)number / 10) % 10;

}

int number\_of\_units(int number)

{

return number % 10;

}

int suitable\_digits(int a, int b)

{

int counter = 0;

for(int n = 100; n < 1000; n++)

{

if (number\_of\_hundreds(n) + tens\_number(n) == a - b && number\_of\_units(n) == b)

{

counter++;

}

}

return counter;

}

int get\_int()

{

int number;

if (scanf("%d", &number) != 1)

{

puts("Inserted a invalid value!");

abort();

}

}

* 1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 36)

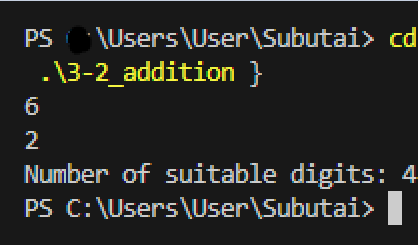


Рисунок  – Результат работы программы

* 1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 37)

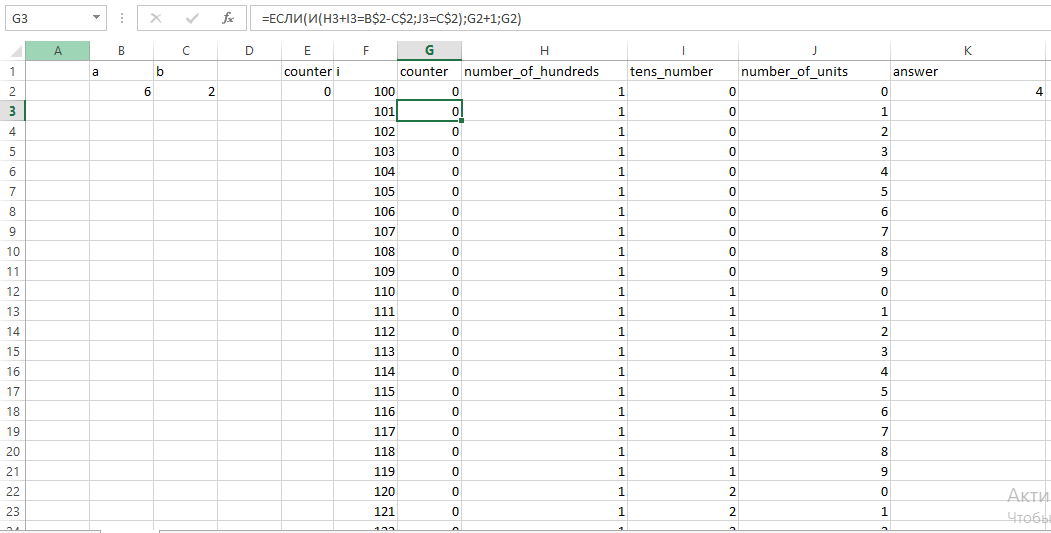
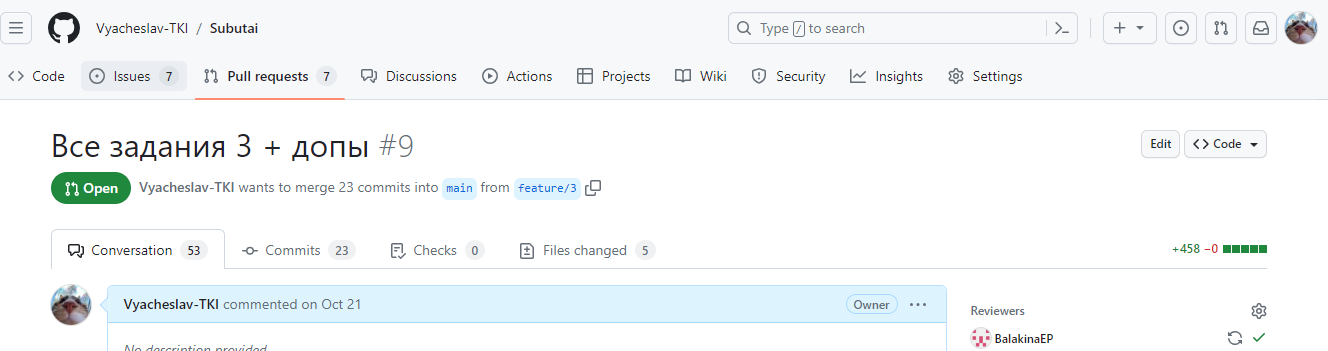


Рисунок  – Результат расчета значения counter

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



1. задание 3.3
   1. Формулировка задания

Протабулировать заданную функцию и сумму функционального ряда разложения этой функции на интервале [*a,b*] и с шагом *h* (шаг и интервал задается в константах). Функциональнй ряд вычисляется по соответствующей рекуррентной формуле с заданной точностью *ɛ*. В результате показать три столбца: значение аргумента, значение функции в данной точке и значение суммы ряда, вычисленное с заданной точностью в данной точке. Два последних столбца должны иметь близкие результаты. Таблица с данными представлена ниже (Таблица 3).

Таблица  – Формулировка задания

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вар | Функция y | Сумма | Интервал | ɛ |
| 11 |  |  |  | 40-5 |

* 1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 38, Рисунок 39). Блок-схемы вспомогательных функций представлена ниже (Рисунок 40, Рисунок 41, Рисунок 42, Рисунок 43, Рисунок 44).

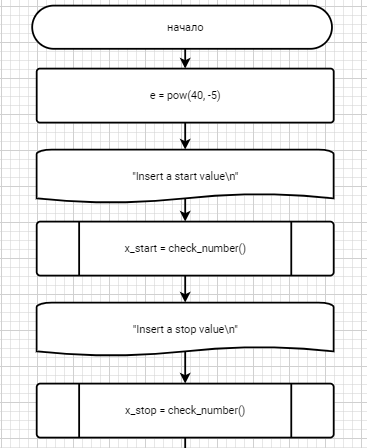


Рисунок  – Блок-схема основного алгоритма часть 1

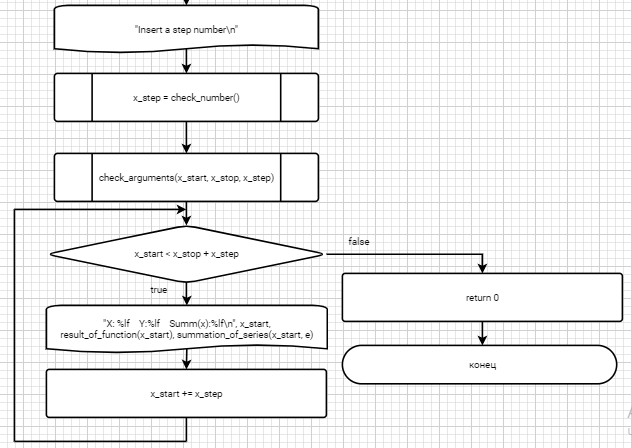


Рисунок  – Блок-схема основного алгоритма часть 2

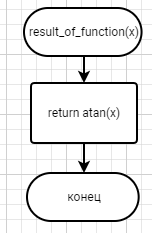


Рисунок 40 – Блок-схема функции result\_of\_function(x)

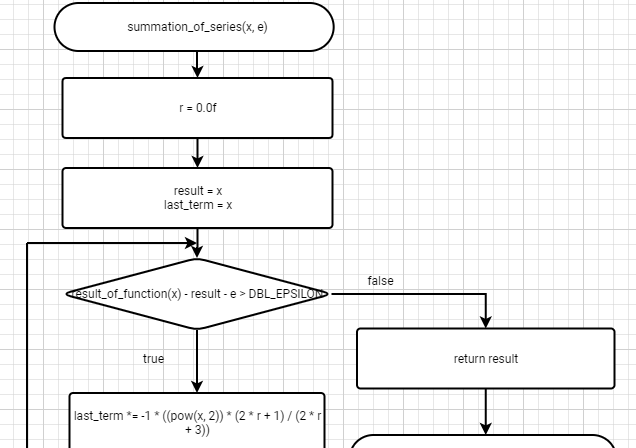


Рисунок 41 – Блок-схема функции summation\_of\_series(x, e) часть 1

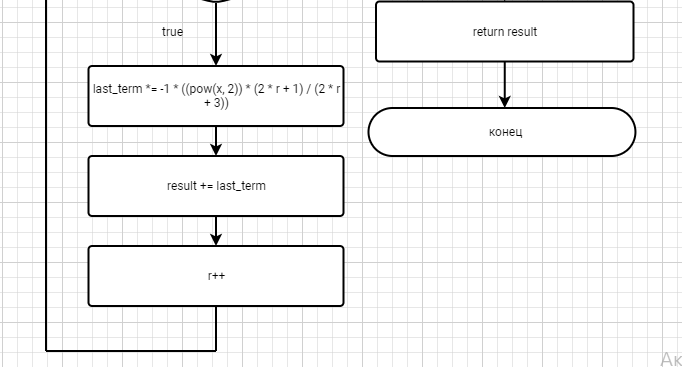


Рисунок 42 – Блок-схема функции summation\_of\_series(x, e) часть 2

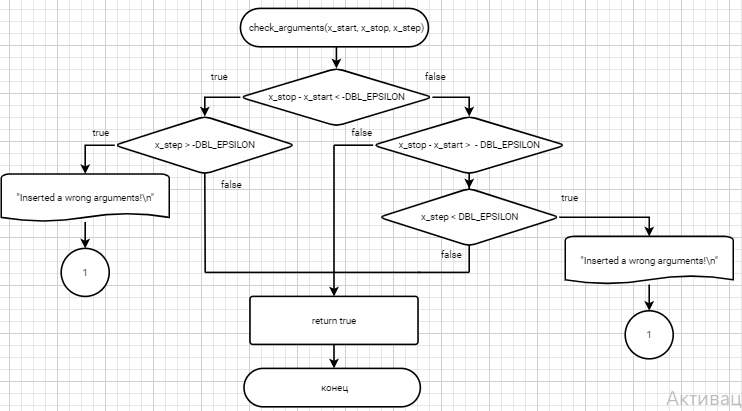


Рисунок 43 – Блок-схема функции check\_arguments(x\_start, x\_stop, x\_step)

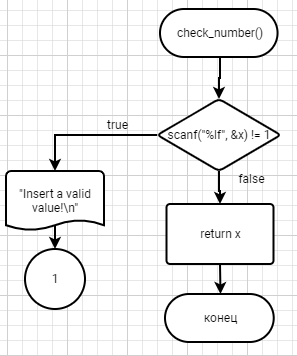


Рисунок  – Блок-схема функции check\_number()

* 1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <stdbool.h>

#include <float.h>

/\*\*

\* @brief Проверяет введенное значение как цифру

\* @return Веденная переменная

\*/

double check\_number();

/\*\*

\* @brief Проверяет значения на правильность

\* @param x\_start начальное значение

\* @param x\_stop конечное значение

\* @param x\_step значение шага

\* @return true переменные правильно введены

\*/

bool check\_arguments(double x\_start, double x\_stop, double x\_step);

/\*\*

\* @brief Вычисляет результат заданной функции

\* @param x аргумент функции

\* @return atan(x) значение функции

\*/

double result\_of\_function(double x);

/\*\*

\* @brief Вычисляет сумму ряда с заданной точностью

\* @param x переменная ряда

\* @param e точность вычисления

\* @return result сумма ряда

\*/

double summation\_of\_series(double x, const double e);

/\*\*

\* @brief точка входа в программу

\* @return 0 программа исправна

\*/

int main(){

const double e = pow(40, -5);

puts("Insert a start value\n");

double x\_start = check\_number();

puts("Insert a stop value\n");

double x\_stop = check\_number();

puts("Insert a step number\n");

double x\_step = check\_number();

check\_arguments(x\_start, x\_stop, x\_step);

while (x\_start < x\_stop + x\_step){

printf("X: %lf Y: %lf Summ(x): %lf\n", x\_start, result\_of\_function(x\_start), summation\_of\_series(x\_start, e));

x\_start += x\_step;

}

return 0;

}

double check\_number(){

double x;

if (scanf("%lf", &x) != 1)

{

puts("Insert valid value!\n");

abort();

}

return x;

}

double result\_of\_function(double x){

return atan(x);

}

double summation\_of\_series(double x, const double e)

{

double r = 0.0f;

double result = x, last\_term = x;

while (result\_of\_function(x) - result - e > DBL\_EPSILON)

{

last\_term \*= -1 \* ((pow(x, 2)) \* (2 \* r + 1) / (2 \* r + 3));

result += last\_term;

r++;

}

return result;

}

bool check\_arguments(double x\_start, double x\_stop, double x\_step)

{

if (x\_stop - x\_start < DBL\_EPSILON)

{

if (x\_step > -DBL\_EPSILON)

{

puts("Inserted a wrong arguments!");

abort();

}

}

else if (x\_stop - x\_start > -DBL\_EPSILON)

{

if (x\_step < DBL\_EPSILON)

{

puts("Inserted a wrong arguments!");

abort();

}

}

return true;

}

* 1. Результаты выполнения программы

Результат выполнения программы представлена ниже (Рисунок 45).

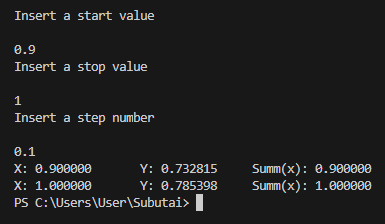


Рисунок  – Результат выполнения программы

* 1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 46, Рисунок 47)

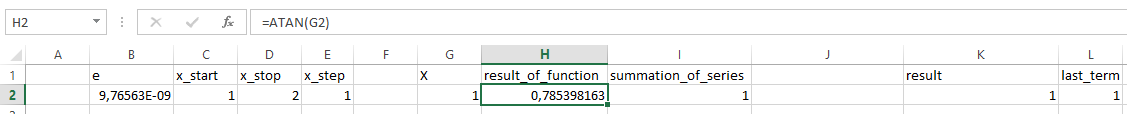


Рисунок  – Результат работы подпрограммы взятия функции

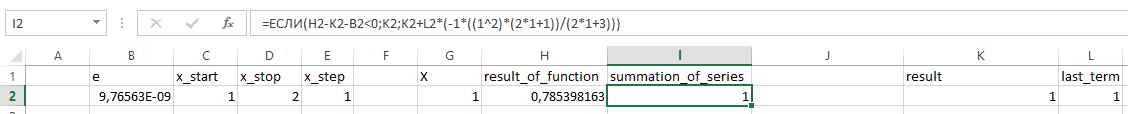


Рисунок  – Результат работы подпрограммы расчета суммы ряда с точностью e

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

