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 ПРАКТИЧЕСКОЙ РАБОТЕ №3

По дисциплине «Введение в языки программирования»

Выполнил: ст. гр. ТКИ – 111

Насонова А.Н.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2021

**Оглавление**

[**Задание 3-1** 3](#_Toc93830745)

[**Формулировка задания** 3](#_Toc93830746)

[**Блок-схема алгоритма** 5](#_Toc93830747)

[**Программа на языке С++** 6](#_Toc93830748)

[**Решение тестового примера на С++** 7](#_Toc93830749)

[**Решение тестового примера в Excel** 8](#_Toc93830750)

[**Задание 3-2(а)** 9](#_Toc93830751)

[**Формулировка задания** 9](#_Toc93830752)

[**Вычисление рекуррентного члена** 10](#_Toc93830753)

[**Блок-схема алгоритма** 11](#_Toc93830754)

[**Программа на языке С++** 12](#_Toc93830755)

[**Решение тестового примера на C++** 14](#_Toc93830756)

[**Решение тестового примера в Excel** 15](#_Toc93830757)

[**Задание 3-2(б)** 16](#_Toc93830758)

[**Формулировка задания** 16](#_Toc93830759)

[**Блок-схема алгоритма** 17](#_Toc93830760)

[**Программа на языке C++** 18](#_Toc93830761)

[**Решение тестового примера C++** 20](#_Toc93830762)

[**Задание 3-3** 21](#_Toc93830763)

[**Формулировка задания** 21](#_Toc93830764)

[**Вычисление рекуррентного члена** 22](#_Toc93830765)

[**Блок-схема алгоритма** 23](#_Toc93830766)

[**Программа на языке С++** 24](#_Toc93830767)

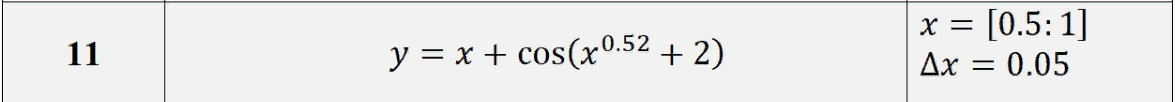
[**Решение тестового примера на C++** 26](#_Toc93830768)

[**Решение тестового примера в Excel** 27](#_Toc93830769)

# **Задание 3-1**

# **Формулировка задания**

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.

****

# **Блок-схема алгоритма**

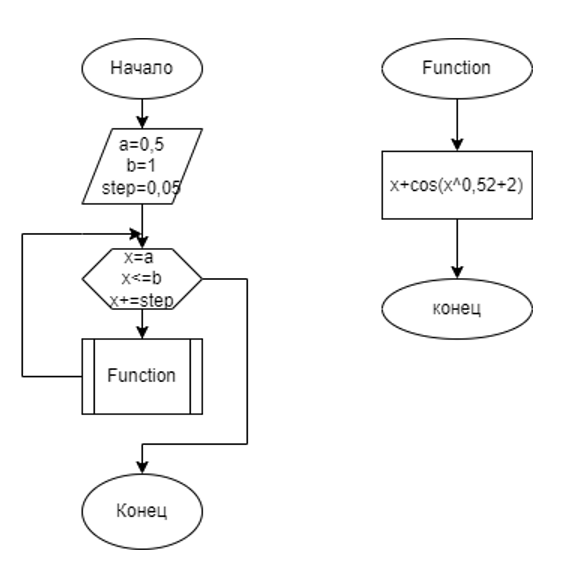
****

Рисунок 1 – Блок-схема алгоритма

# **Программа на языке С++**

#include<iostream>

#include <iomanip>

#include<cmath>

using namespace std;

/\*\*

\* \brief функция для расчета значений табуляции.

\* \x начало интервала.

\* \return значение табуляции.

\*/

double Function(const double x);

/\*\*

\* \brief Возможность выполнения функции.

\* \return в случае успеха, возвращает 1)

\*/

int main() {

int a = 50;

const int b = 100;

const int step = 5;

cout << setw(10) << "x" << setw(30) << "y" << endl;

for (int x = a; x <= b; x += step) {

cout << setw(10) << x/100.0 << setw(30) << Function(x/100.0) << endl;

}

system("pause");

return 0;

}

double Function(const double x) {

return x + cos(pow(x, 0.52) + 2);

}

# **Решение тестового примера на С++**

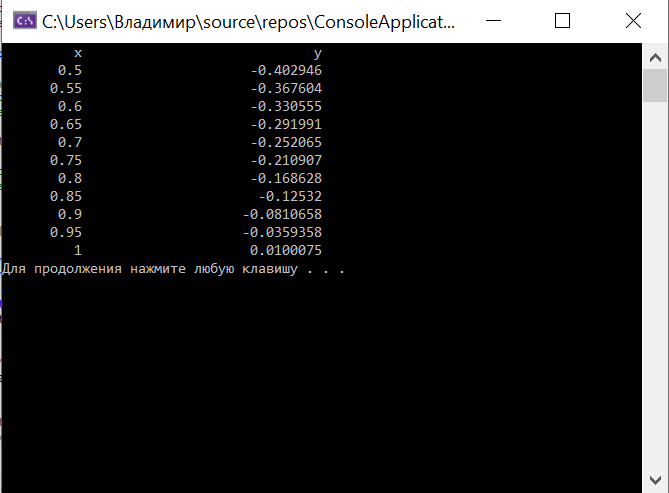
****

Рисунок 2 – Решение тестового примера в Excel

# **Решение тестового примера в Excel**

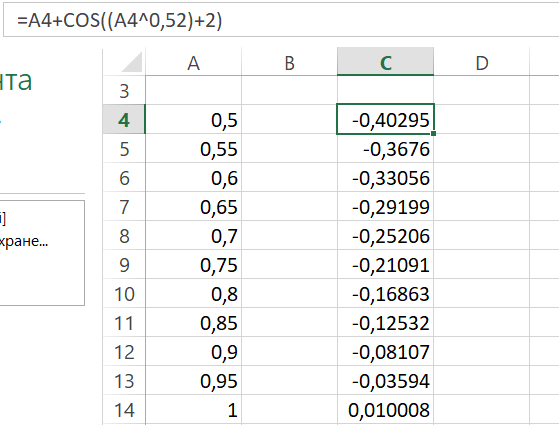
****

Рисунок 3 – Решение тестового примера в Excel

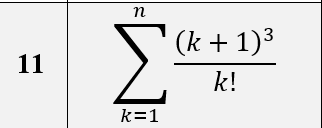
# **Задание 3-2(а)**

# **Формулировка задания**

Составьте две программы:

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

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

****

## **Вычисление рекуррентного члена**

# **Блок-схема алгоритма**

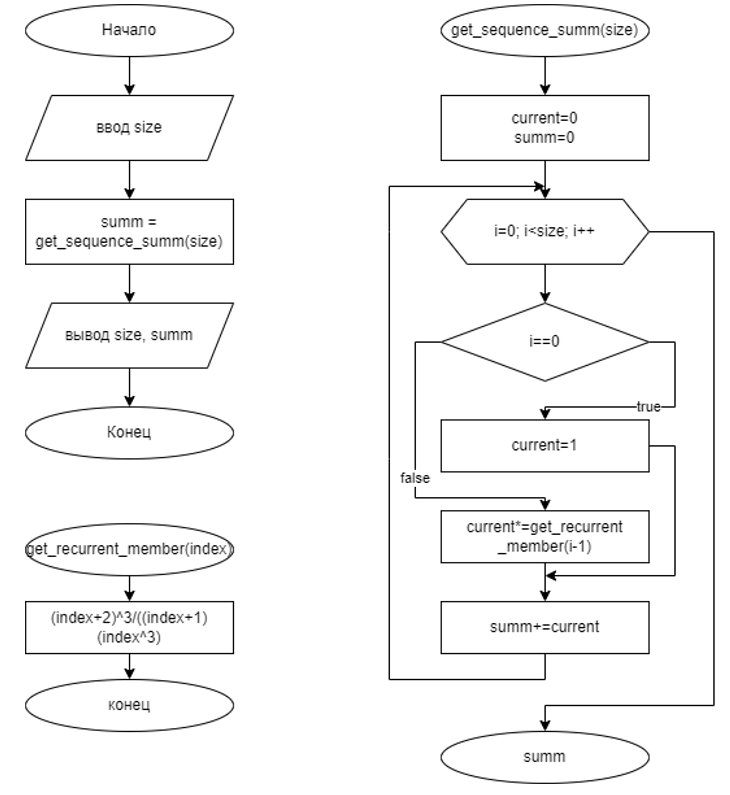
****

Рисунок 4 – Блок-схема алгоритма

# **Программа на языке С++**

#include <iostream>

/\*\*

\* \brief Пользовательский ввод количества членов последовательности.

\* \param message Мотивационное сообщение.

\* \return Размер последовательности.

\* \exception std::invalid\_argument В случае, если размер последовательности меньше 1.

\*/

size\_t get\_sequence\_size(const std::string& message);

/\*\*

\* \brief Рассчитывает сумму последовательность с количеством членов (\a size).

\* \param size Количество членов последовательности.

\* \return Сумма последовательности.

\*/

double get\_sequence\_summ(const size\_t size);

/\*\*

\* \brief Рассчитывает значение рекуррентного члена в зависимости от индекса (\a index).

\* \param index Индекс, номер итерации.

\* \return Значение рекуррентного члена.

\*/

double get\_recurrent\_member(const size\_t index) noexcept;

/\*\*

\* \brief Точка входа в программу.

\* \return 0 – в случае успеха,<br/>1 – в противном случае.

\*/

int main() noexcept

{

try

{

setlocale(LC\_ALL, "Russian");

const auto size = get\_sequence\_size("Введите количество членов последовательности ");

const double summ = get\_sequence\_summ(size);

std::cout << "Сумма последовательности из " << size << " членов равна: " << summ << std::endl;

return 0;

}

catch (const std::exception& e)

{

std::cerr << e.what() << std::endl;

return 1;

}

}

double get\_sequence\_summ(const size\_t size) {

double current = 0;

double summ = 0;

for (size\_t i = 0; i < size; i++) {

if (i == 0) {

current = 1;

}

else {

current \*= get\_recurrent\_member(i - 1);

}

summ += current;

}

return summ;

}

size\_t get\_sequence\_size(const std::string& message)

{

std::cout << message;

int size = 0;

std::cin >> size;

if (size < 1)

{

throw std::invalid\_argument("Значение должно быть больше 0!");

}

return size;

}

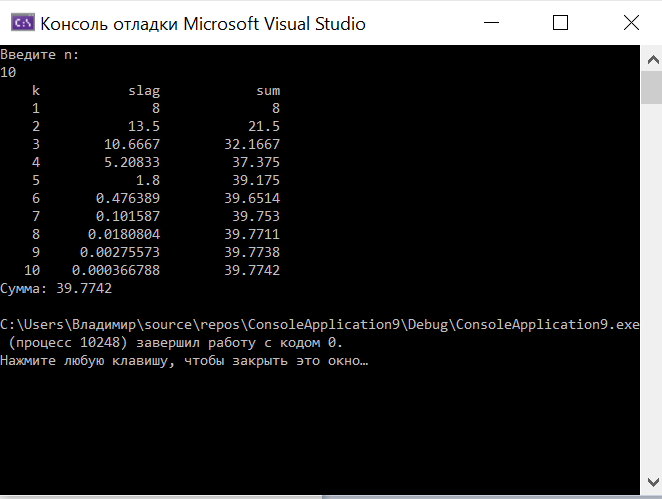
double get\_recurrent\_member(const size\_t index) noexcept

{

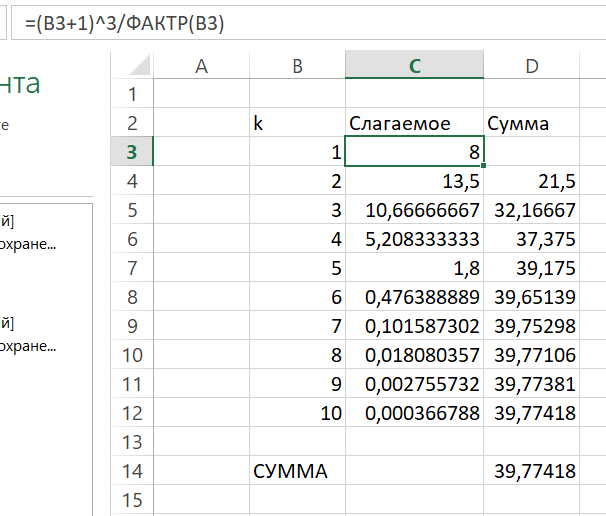
return (pow(index+2,3)/((index+1)pow(index,3));

}

# **Решение тестового примера на C++**

****

# **Решение тестового примера в Excel**

****

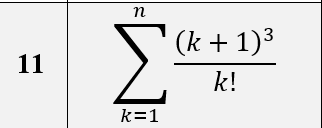
# **Задание 3-2(б)**

# **Формулировка задания**

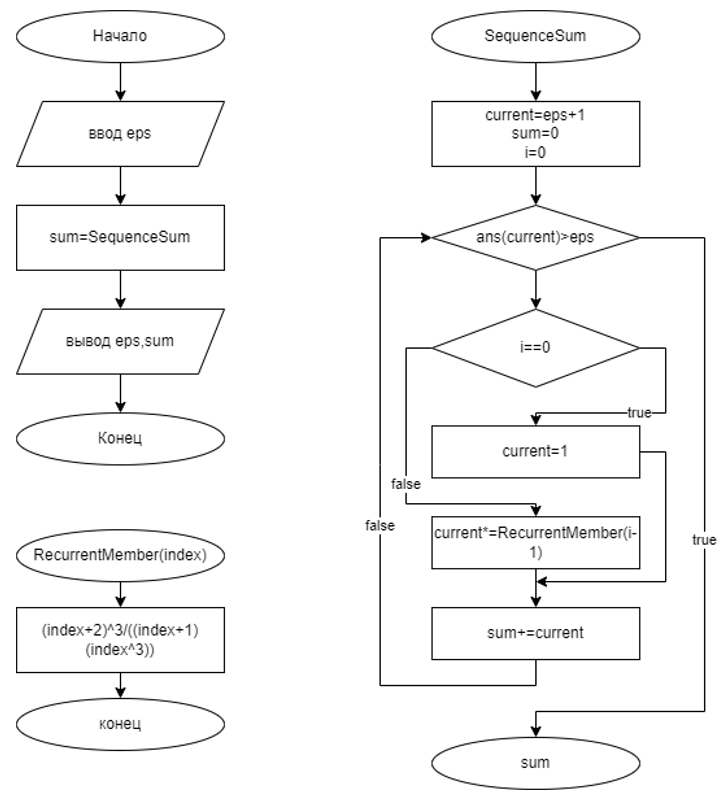
Составьте две программы:

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.

****

# **Блок-схема алгоритма**

****

# **Программа на языке C++**

#include <iostream>

/\*\*

\* \brief Пользовательский ввод необходимой точности вычисления.

\* \param message Мотивационное сообщение.

\* \return Точность вычисления.

\* \exception std::invalid\_argument В случае, если размер последовательности меньше 1.

\*/

double Epsilon(const std::string& message);

/\*\*

\* \brief Рассчитывает сумму последовательность с точностью (\a eps).

\* \param eps Точность вычисления.

\* \return Сумма последовательности.

\*/

double SequenceSum(const double eps);

/\*\*

\* \brief Рассчитывает значение рекуррентного члена в зависимости от индекса (\a index).

\* \param index Индекс, номер итерации.

\* \return Значение рекуррентного члена.

\*/

double RecurrentMember(const size\_t index) noexcept;

/\*\*

\* \brief Точка входа в программу.

\* \return 0 – в случае успеха,<br/>1 – в противном случае.

\*/

int main() noexcept

{

try

{

setlocale(LC\_ALL, "Russian");

const auto eps = Epsilon("Введите точность вычислений ");

const double Sum = SequenceSum(eps);

std::cout << "Сумма последовательности c точностью " << eps << " равна: " << Sum << std::endl;

return 0;

}

catch (const std::exception& e)

{

std::cerr << e.what() << std::endl;

return 1;

}

}

double SequenceSum(const double eps) {

size\_t i = 0;

double Sum = 0;

double Current = eps + 1;

while (abs(Current) >= eps) {

if (i == 0) {

Current = 1;

}

else {

Current \*= RecurrentMember(i - 1);

}

Sum += Current;

i++;

}

return Sum;

}

double Epsilon(const std::string& message)

{

std::cout << message;

double eps = 0;

std::cin >> eps;

if (eps < 0)

{

throw std::invalid\_argument("Значение должно быть больше 0!");

}

return eps;

}

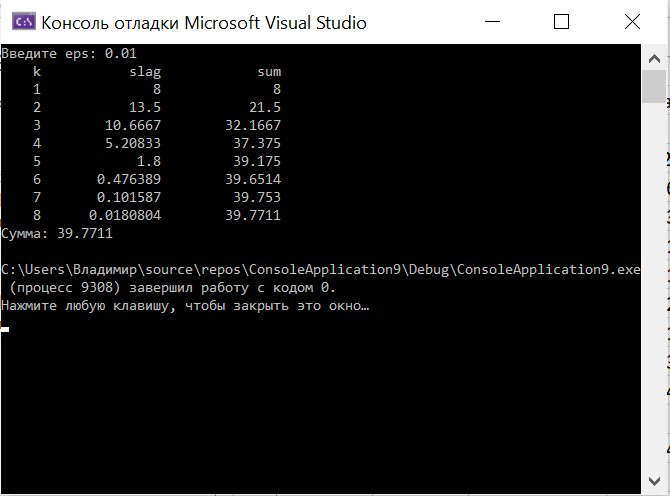
double RecurrentMember(const size\_t index) noexcept

{

return (pow(index + 2, 3) / ((index + 1)pow(index, 3));

}

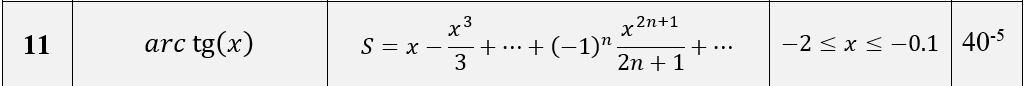
# **Решение тестового примера C++**

****

# **Задание 3-3**

# **Формулировка задания**

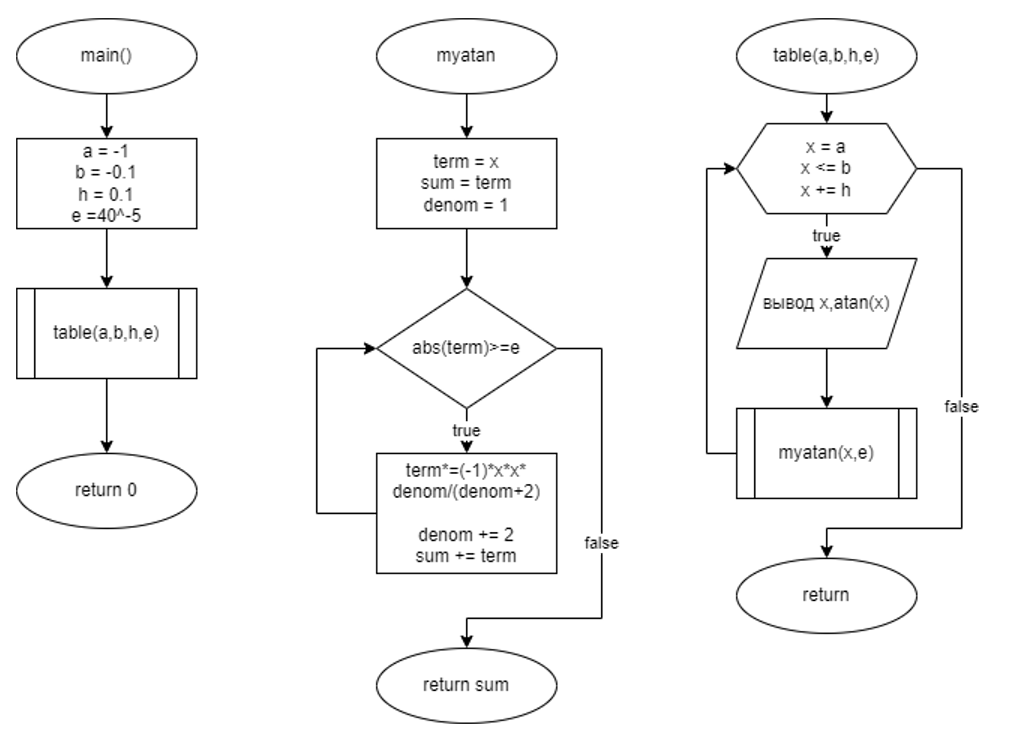
Протабулировать заданную функцию и сумму функционального ряда разложения этой функции на интервале [*a,b*] и с шагом *h* (шаг и интервал задается в константах). Функциональный ряд вычисляется по соответствующей рекуррентной формуле с заданной точностью *ɛ*. В результате показать три столбца: значение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



# **Вычисление рекуррентного члена**

denom - старый знаменатель

# **Блок-схема алгоритма**

****

# **Программа на языке С++**

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

/\*\*

\* \brief Pеализация функции вычисления арктангенса.

\* \param x - Аргумент функции.

\* \param e - Точность вычислений.

\* \return Значение суммы последовательности слагаемых.

\*/

double myatan(const double x, const double e);

/\*\*

\* \brief Вывод в консоль таблицы значений из трех столбцов.

\* \param a - Левая граница интервала.

\* \param b - Правая граница интервала.

\* \param h - Шаг изменения аргумента.

\* \param e - Точность вычислений.

\*/

void table(const double a, const double b, const double h, const double e);

int main() {

  const double a = -1;

  const double b = -0.1;

  const double h = 0.1;

  const double e = pow(40, -5);

  table(a, b, h, e);

  return 0;

}

double myatan(const double x, const double e)

{

  double term = x;

  double sum = term;

  double denom = 1;

  while (abs(term) >= e)

{

    term \*= (-1) \* x \* x \* denom / (denom + 2);

    denom += 2;

    sum += term;

  }

  return sum;

}

void table(const double a, const double b, const double h, const double e)

{

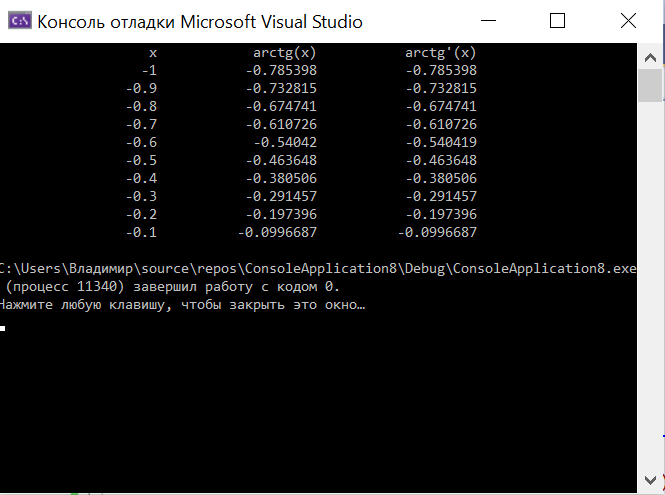
  cout << setw(20) << "x" << setw(20) << "arctg(x)" << setw(20) << "arctg'(x)" << endl;

  for (double x = a; x <= b; x += h)

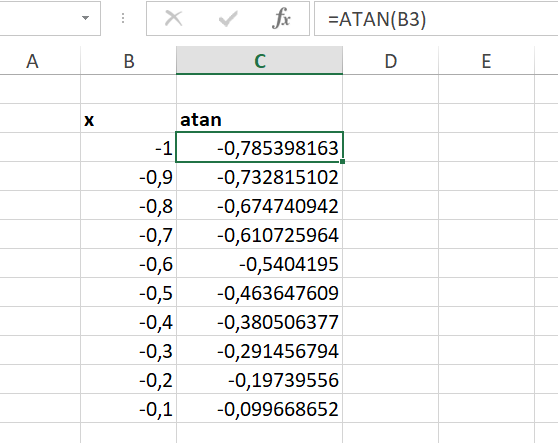
    cout << setw(20) << x << setw(20) << atan(x) << setw(20) << myatan(x, e) << endl;

}

# **Решение тестового примера на C++**

****

# **Решение тестового примера в Excel**

****