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

Черкас Н.В.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2021

Оглавление

[Задание 3–1 3](#_Toc85287969)

[Формулировка задания 3](#_Toc85287970)

[Блок-схема алгоритма 4](#_Toc85287971)

[Программа на языке С++ 5](#_Toc85287972)

[Решение тестовых примеров на С++ 6](#_Toc85287973)

[Решение тестовых примеров в Excel 7](#_Toc85287974)

[Задание 3–2 8](#_Toc85287975)

[Формулировка задания 8](#_Toc85287976)

[Блок-схема алгоритма 9](#_Toc85287977)

[Программа на языке С++ 10](#_Toc85287978)

[Решение тестовых примеров на С++ 12](#_Toc85287979)

[Решение тестовых примеров в Excel 13](#_Toc85287980)

[Задание 3–3 14](#_Toc85287982)

[Формулировка задания 14](#_Toc85287983)

[Блок-схема алгоритма 15](#_Toc85287984)

[Программа на языке С++ 16](#_Toc85287985)

[Решение тестовых примеров на С++ 18](#_Toc85287986)

[Решение тестовых примеров в Excel 1](#_Toc85287987)**9**

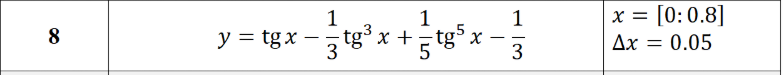
**Задание 3-1**

**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.

При решении данного задания в MatLab необходимо построить график!

**Таблица 1**

****

**Блок-схема алгоритма**

Начало

x, step

x>=0.8

**True**

**False**

y = tg(x) - 1 / 3 \* tg(x)^3 + 0.2 \* tg(x)^5 - 1 / 3

x=x + step

y

Конец

Рисунок 1 – Блок-схема алгоритма

**Программа на языке C++**

|  |  |
| --- | --- |
|  | #include <cmath> |
|  | #include <iostream> |
|  | using namespace std; |
|  |  |
|  | /\*\* |
|  | \* \brief Проверка косинуса на равенство c 0. |
|  | \* \param x - аргумент функции. |
|  | \* \return - возвращает выполнение или невыполнение условия. |
|  | \*\*/ |
|  | bool isCalculated(const double x); |
|  |  |
|  | /\*\* |
|  | \* \brief Расчет функции. |
|  | \* \param x - аргумент функции. |
|  | \* \return Возвращает значение функции y. |
|  | \*\*/ |
|  | double getY(const double x); |
|  | /\*\* |
|  | \* \brief Точка входа в программу. |
|  | \* \return Возвращает 0 в случае успешного выполнения. |
|  | \*/ |
|  | int main() |
|  | { |
|  | const double x = 0; |
|  | const double step = 0.05; |
|  | const double border = 0.8; |
|  | const double firstValue = 0.0; |
|  | for (double x = firstValue; x <= border; x = x + step) |
|  | { |
|  | if (!isCalculated(x)) |
|  | cout << "y = " << getY(x) << endl; |
|  | else cout<< "err"<< endl; |
|  | } |
|  | return 0; |
|  | } |
|  |  |
|  | bool isCalculated(const double x){ |
|  | const double epsilon = 0.001; |
|  | return abs(cos(x)) <= epsilon; |
|  | } |
|  |  |
|  | double getY(const double x) { |
|  | return tan(x) - 1.0 / 3.0 \* pow(tan(x), 3 ) + 0.2 \* pow(tan(x), 5 ) - 1.0 / 3.0;} |

**Решение тестовых примеров на C++**

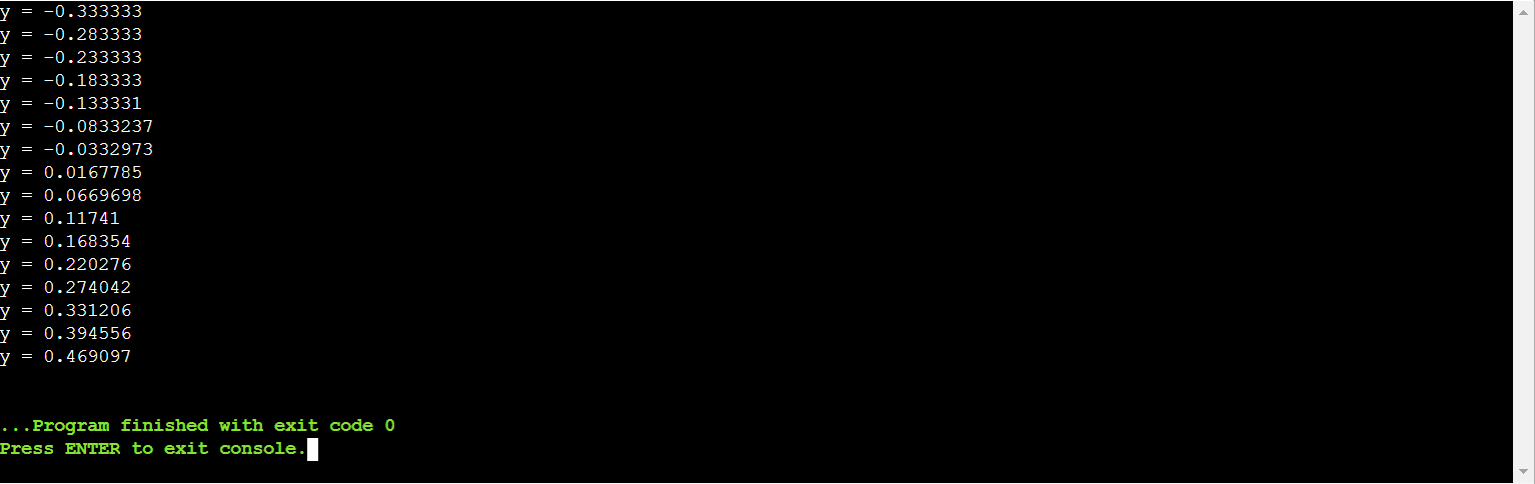
****

Рисунок 2 – Решение тестовых примеров на языке C++

**Решение тестовых примеров в Excel**

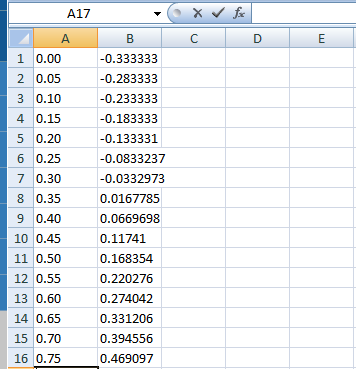
****

Рисунок 3 – Решение тестовых примеров в Excel

**Задание 3-2**

**Формулировка задания**

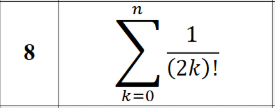
Составьте две программы:

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.

**Таблица 2**

****

**Блок-схема алгоритма**

Начало

k, n, e

k < n

**False**

cur = 0

k = 0

**True**

getCurrent = 1.0 / (2.0 \* k + 1)

cur = cur + getCurrent

sum1 = sum1 + cur

k = k +1

abs(cur)<e

**False**

**True**

getCurrent = 1.0 / (2.0 \* k + 1)

cur = cur \* getCurrent

sum2 = sum2+ current

k = k +1

sum1 sum2

Конец

Рисунок 4 – Блок-схема алгоритма

**Программа на языке C++**

|  |  |
| --- | --- |
|  | #include <iostream> |
|  |  |
|  | using namespace std; |
|  |  |
|  | /\*\* |
|  | \* \brief функция определения текущего элемента ряда. |
|  | \* param k - переменная для вычисления элемента ряда. |
|  | \* \return значение текущего элемент ряда. |
|  | \*/ |
|  | double getCurrent(const int k); |
|  |  |
|  | /\*\* |
|  | \* \brief функция вычисления суммы первых n элементов ряда. |
|  | \* param n - количество элементов ряда. |
|  | \* param k - переменная для вычисления элемента ряда. |
|  | \* \return сумма первых n элементов ряда. |
|  | \*/ |
|  | double getSumm1(int n, int k); |
|  |  |
|  | /\*\* |
|  | \* \brief вычисление суммы членов ряда не меньших числа e. |
|  | \* param e - константа по условию. |
|  | \* param k - переменная для вычисление элемента ряда. |
|  | \* \return сумма членов ряда не меньших числа e. |
|  | \*/ |
|  | double getSumm2(double e, int k); |
|  |  |
|  | /\*\* |
|  | \* \brief точка входа в программу. |
|  | \* return 0 в случае успеха. |
|  | \*/ |
|  | int main() |
|  | { |
|  | int n; |
|  | double e = 0.0; |
|  | int k = 0; |
|  | cout<<"Введите количество членов ряда"<< endl; |
|  | cin >> n; |
|  | cout<< "Введите e "; |
|  | cin>>e; |
|  | cout<< getSumm1(n, k)<< endl; |
|  | cout<< getSumm2(e, k); |
|  | return 0; |
|  | } |
|  |  |
|  | double getCurrent(const int k){ |
|  | return 1.0 / (2.0 \* k + 1); |
|  | } |
|  |  |
|  | double getSumm1(int n,int k){ |
|  | double sum; |
|  | for (int i = 1; i <= n; i++){ |
|  | sum = sum + getCurrent(k); |
|  | k=k+1; |
|  | } |
|  | return sum; |
|  | } |
|  |  |
|  | double getSumm2(double e, int k){ |
|  | double cur, sum; |
|  | cur = getCurrent(k); |
|  | while (cur > e){ |
|  | sum = sum + cur; |
|  | k++; |
|  | cur = getCurrent(k); |
|  | } |
|  | return sum; |
|  | } |

**Решение тестовых примеров на C++**

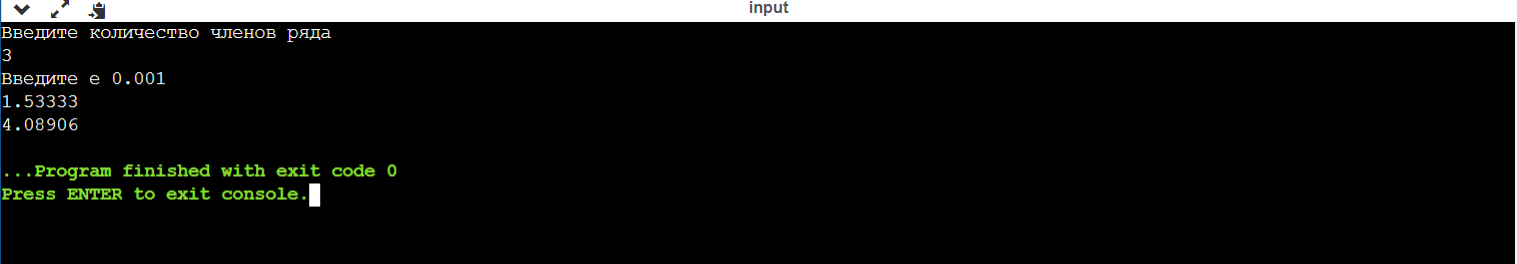
****

Рисунок 5 – Решение тестовых примеров на языке C++

**Решение тестовых примеров в Excel**

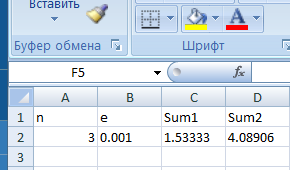
****

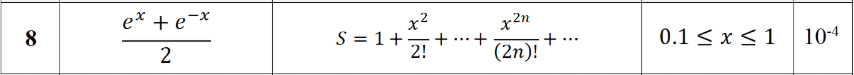
Рисунок 6 – Решение тестовых примеров в Excel

**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). 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 3**

****

**Блок-схема алгоритма**

Начало

e, h, x

x<1

**False**

**True**

Function = (exp(x) + exp(-x)) / 2.0

previous = current

current = previous \* (x^ 2 \* n) / (2 \* n + 1)

sum = sum + current

n= n + 1

x, function, sum

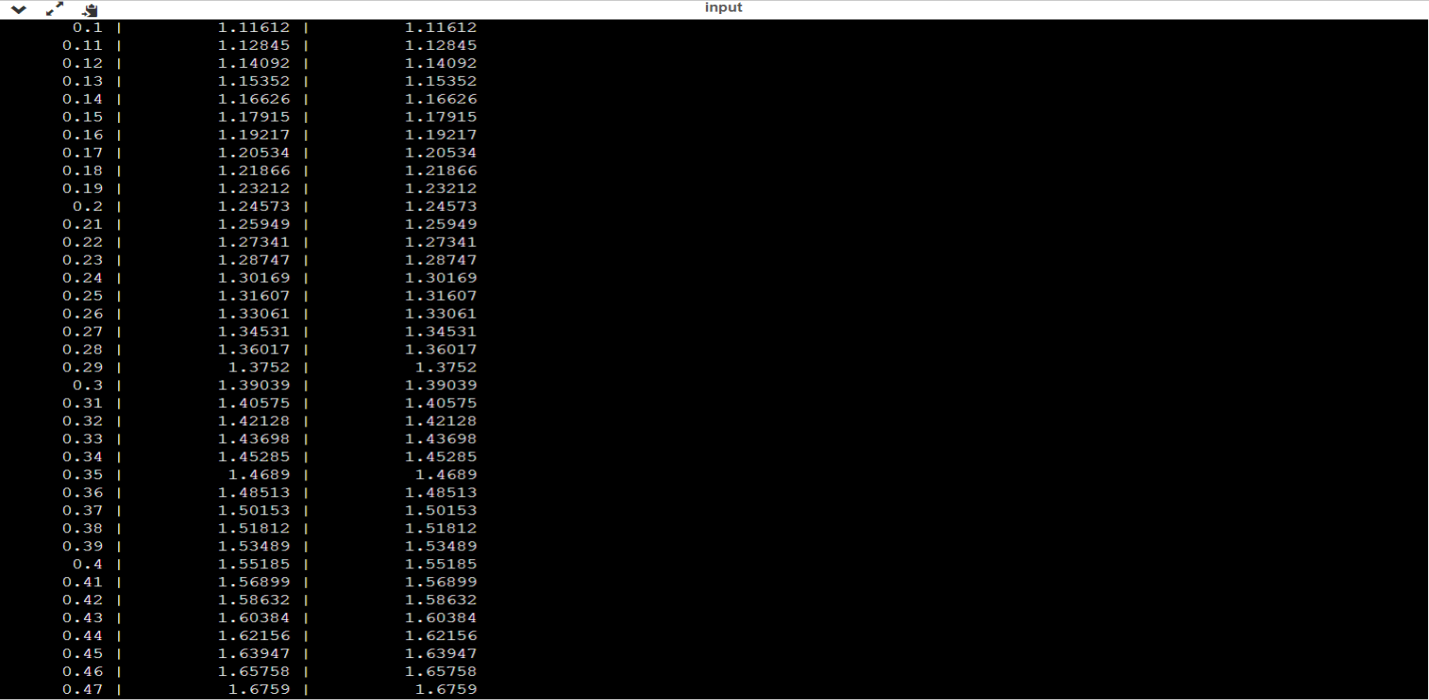
Конец

Рисунок 7 – Блок-схема алгоритма

**Программа на языке C++**

|  |  |
| --- | --- |
|  | #include <iomanip> |
|  | #include <iostream> |
|  | #include <cmath> |
|  |  |
|  | using namespace std; |
|  |  |
|  | /\*\* |
|  | \* \brief Расчет функции exp в степени x. |
|  | \* \param x Показатель степени. |
|  | \* \return Значение exp в степени x. |
|  | \*/ |
|  | double GetEPowerX(const double x); |
|  |  |
|  | /\*\* |
|  | \* \brief Вычисление рекуррентного члена ряда. |
|  | \* \param x Параметр функции, аппроксимируемой рядом. |
|  | \* \param n Индекс члена ряда. |
|  | \* \return Значение рекуррентного члена. |
|  | \*/ |
|  | double GetRecurrent(const double x, const unsigned int n); |
|  |  |
|  | /\*\* |
|  | \* \brief Вычисление суммы ряда от x, до того момента пока |x\_i - x\_{x-1}| >= e |
|  | \* \param x Показатель степени. |
|  | \* \param e Погрешность вычислений. |
|  | \* \return Значение 3 в степени x. |
|  | \*/ |
|  | double GetSumOfSeries(const double x, const double e); |
|  |  |
|  | /\*\* |
|  | \* \brief Форматный вывод в поток вывода out. |
|  | \* \param argument Аргумент табулируемой функции. |
|  | \* \param function Значение табулируемой функции. |
|  | \* \param series Аппроксимация табулируемой функции. |
|  | \*/ |
|  | void FormatOutput(const double argument, const double function, const double series, ostream& out = cout); |
|  |  |
|  | /\*\* |
|  | \* \brief Точка входа в программу. |
|  | \* \return Код ошибки (0 - успех). |
|  | \*/ |
|  | int main() |
|  | { |
|  | const auto leftBound = 0.1; |
|  | const auto rightBound = 1.0; |
|  | const auto e = 0.0001; |
|  | const auto h = 0.01; |
|  |  |
|  | auto x = leftBound; |
|  | while (x <= rightBound + e) |
|  | { |
|  | const auto function = GetEPowerX(x); |
|  | const auto series = GetSumOfSeries(x, e); |
|  |  |
|  | FormatOutput(x, function, series); |
|  |  |
|  | x += h; |
|  | } |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  | double GetEPowerX(const double x) |
|  | { |
|  | return (exp(x) + exp(-x)) / 2.0; |
|  | } |
|  |  |
|  | double GetRecurrent(const double x, const unsigned int n) |
|  | { |
|  | return pow(x, 2 \* n) / (2 \* n + 1); |
|  | } |
|  |  |
|  | double GetSumOfSeries(const double x, const double e) |
|  | { |
|  | auto previous = 0.0; |
|  | auto current = 1.0; |
|  | auto sum = current; |
|  | unsigned int n = 0; |
|  |  |
|  | do |
|  | { |
|  | previous = current; |
|  | current = previous \* GetRecurrent(x, n); |
|  | sum += current; |
|  | n++; |
|  | } while (abs(previous - current) >= e); |
|  |  |
|  | return sum; |
|  | } |
|  |  |
|  | void FormatOutput(const double argument, const double function, const double series, std::ostream&) |
|  | { |
|  | cout << setw(10) << argument << " | "; |
|  | cout << setw(15) << function << " | "; |
|  | cout << setw(15) << series << "\n"; |
|  | } |

**Решение тестовых примеров на C++**



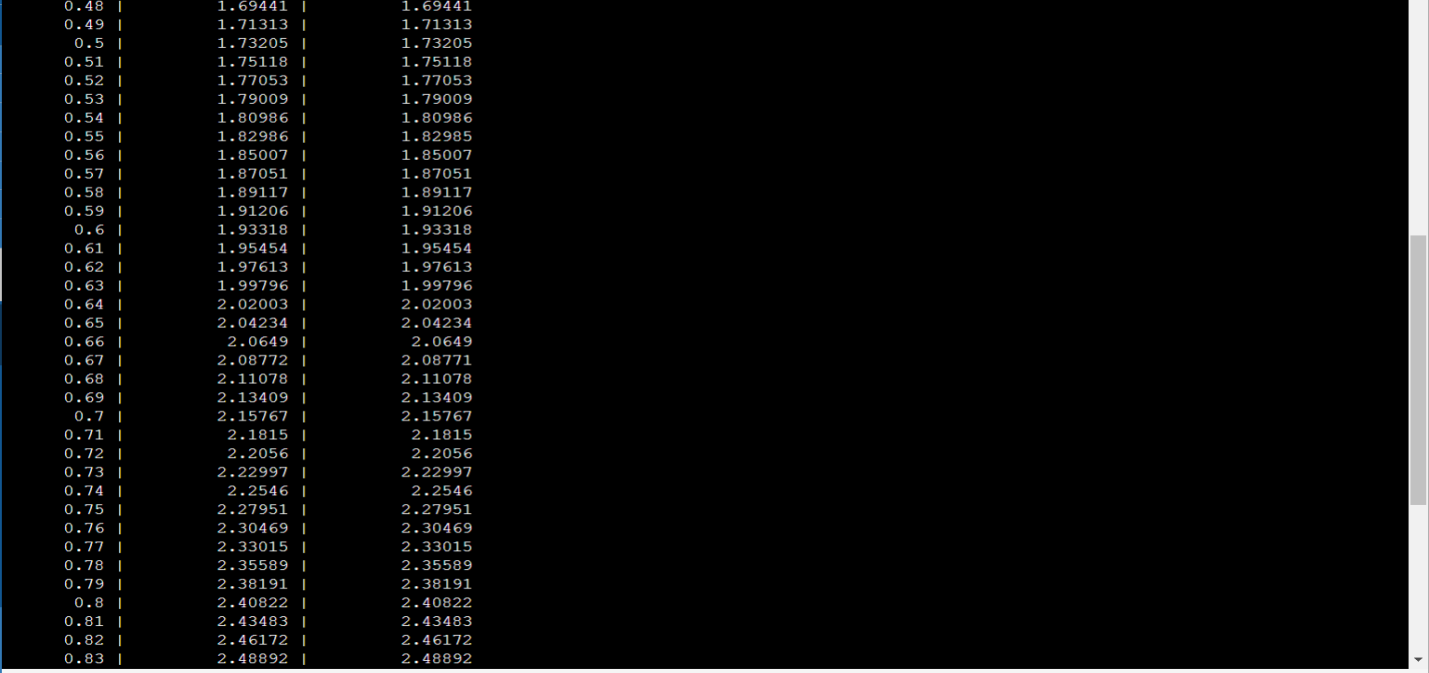
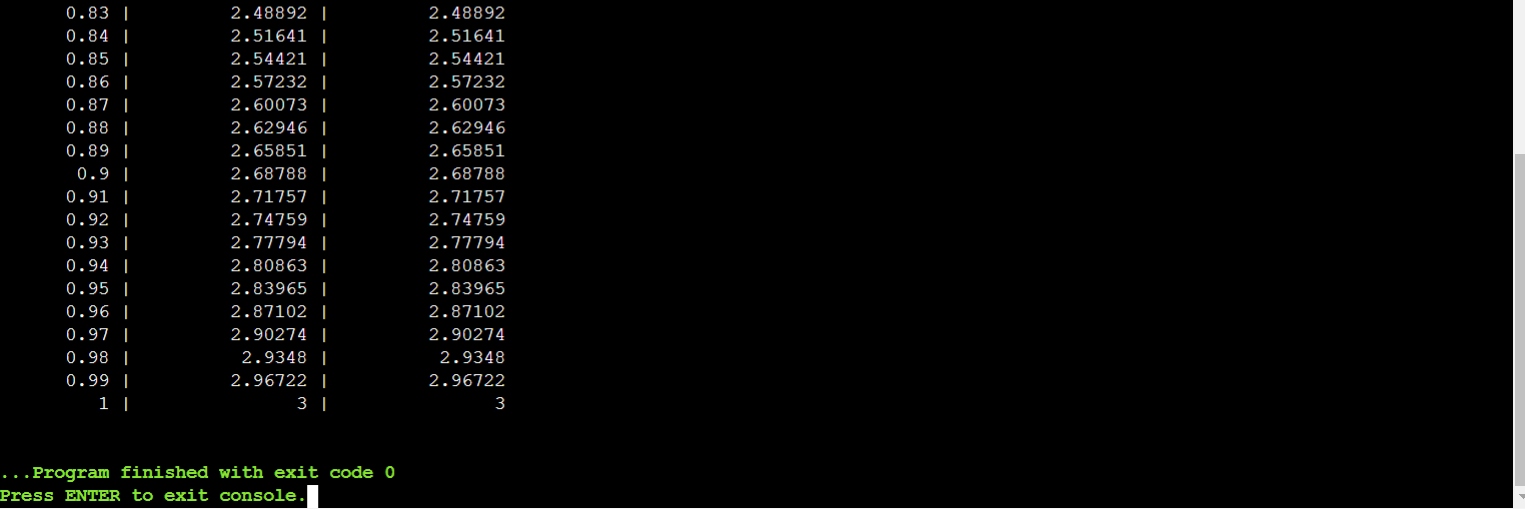
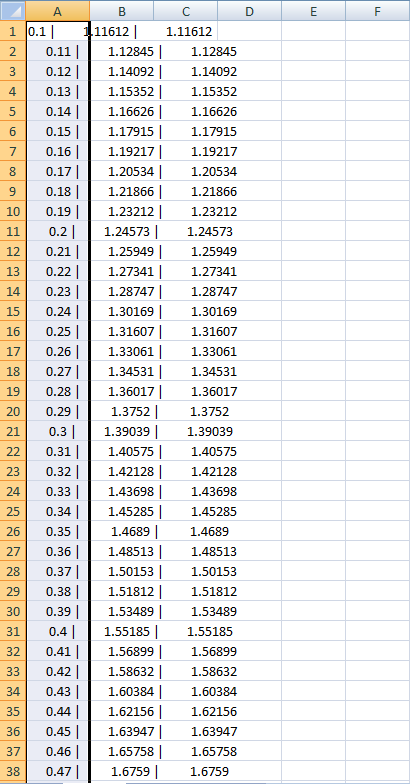
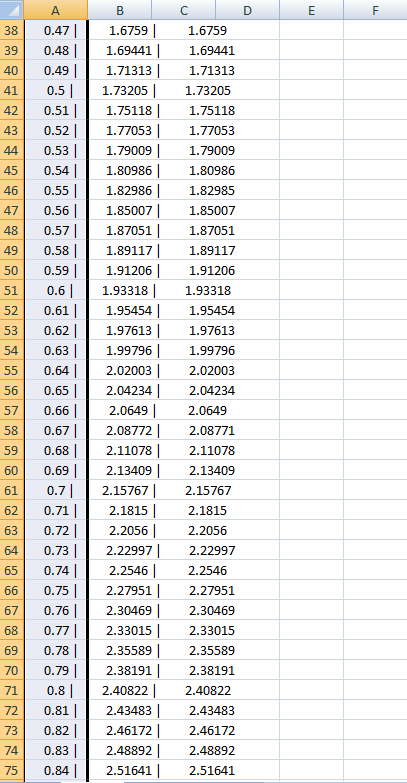
 

Рисунок 8 – Решение тестовых примеров в C++

**Решение тестовых примеров в Excel**

****

****

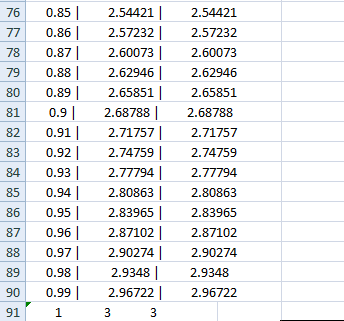
****

Рисунок 9 – Решение тестовых примеров в Excel