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

Родин К. П.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2021

Оглавление

[**Задание 3–1** 3](#_Toc91010045)

[**Условие задания** 3](#_Toc91010046)

[**Блок-схема алгоритма** 4](#_Toc91010047)

[**Программа на языке С++** 5](#_Toc91010048)

[**Решение тестовых примеров на С++** 6](#_Toc91010049)

[**Решение тестовых примеров в Excel** 7](#_Toc91010050)

[**Задание 3–2** 8](#_Toc91010051)

[**Условие задания** 8](#_Toc91010052)

[**Блок-схема алгоритма** 9](#_Toc91010053)

[**Программа на языке С++** 10](#_Toc91010054)

[**Решение тестовых примеров на С++** 12](#_Toc91010055)

[**Решение тестовых примеров в Excel** 13](#_Toc91010056)

# 

# **Задание 3–1**

## **Условие 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

|  |  |  |
| --- | --- | --- |
| Вариант | Функция | Константы |
| 1 |  |  |

## **Блок-схема алгоритма**

y

x = x + step

x > 3

Начало

x, step

Конец

True

False

Рисунок 1 – Блок-схема 3-1

## **Программа на языке С++**

#define \_USE\_MATH\_DEFINES

#include <cmath>

#include <iostream>

#include <iomanip>

/\*\*

\* \brief Рассчитывает значение y при заданном x.

\* \param x Значение x

\* \return Значение y при заданном x

\*/

double Count(double x);

/\*\*

\* \brief Проверка y при заданном х

\* \param x Значение x

\* \return true, y существует при заданном х

\*/

bool isCalculated(double x);

/\*\*

\* \brief Точка входа в программу

\* \return (Код 0) Успех

\*/

int main()

{

const double x\_0 = 2;

const double x\_1 = 3.1;

const double step = 0.1;

double x = x\_0;

while (x < x\_1)

{

if (isCalculated(x))

{

const double y = Count(x);

std::cout << "x = " << std::setw(10) << std::left << std::setprecision(4) << x << " y = " << y << "\n";

}

else

{

std::cout << "При x = " << x << " y не существует" << "\n";

}

x += step;

}

return 0;

}

double Count(const double x) {

return 3 \* sin(sqrt(x)) + 0.39 \* x - 3.8;

}

bool isCalculated(const double x) {

return x >= 0;

}

## **Решение тестовых примеров на С++**

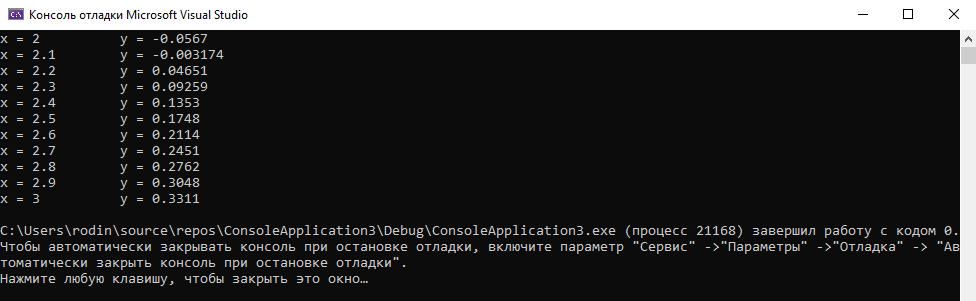


Рисунок 2 – Решение тестового примера для задания 3-1 на С++

## **Решение тестовых примеров в Excel**

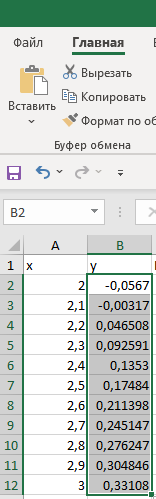


Рисунок 3 - Решение тестового примера задания 3-1 в Excel

# **Задание 3–2**

## **Условие 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

|  |  |
| --- | --- |
| Номер варианта | Ряд |
| 1 |  |

## **Блок-схема алгоритма**

Начало

Size

Epsilon

Index

Current  
Sum

SumE

False

Index < Size

Current = 0

Index = 0

True

RecurrentNumber = -1.0 / ((index + 3) \* (index + 4))

current = current \* RecurrentNumber

Sum = sum + current

Index = Index +1

abs(current)< eps

False

RecurrentNumber = -1.0 / ((index + 3) \* (index + 4))

current = current \* RecurrentNumber

SumE = sumE + current

Index = Index +1

True

Sum SumE

Конец

Рисунок 4 – Блок-схема 3-2

## **Программа на языке С++**

#include <iostream>

using namespace std;

/\*\*

\* \brief Пользовательский ввод необходимой точности вычисления.

\* \param message Мотивационное сообщение

\* \return Точность вычисления

\* \exception invalid\_argument В случае, если epsilon меньше 0

\*/

double getEpsilon(const string& message);

/\*\*

\* \brief Рассчитывает сумму последовательности

\* \param size Количество членов последовательности

\* \return Сумма последовательности

\*/

double getSum(const size\_t size);

/\*\*

\* \brief Рассчитывает сумму последовательность с точностью epsilon

\* \param eps Точность вычисления

\* \return Сумма последовательности с точностью epsilon

\*/

double getSumE(const double eps);

/\*\*

\* \brief Пользовательский ввод количества членов последовательности

\* \param message Мотивационное сообщение

\* \return Размер последовательности

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

\*/

size\_t getSequenceSize(const string& message);

/\*\*

\* \brief Рассчитывает значение рекуррентного члена в зависимости от индекса

\* \param index Индекс, номер итерации

\* \return Значение рекуррентного члена

\*/

double getRecurrentNumber(const size\_t index) noexcept;

/\*\*

\* \brief Точка входа в программу

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

\*/

int main() noexcept

{

setlocale(LC\_ALL, "Russian");

try

{

const auto size = getSequenceSize("Введите количество членов последовательности n = ");

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

const double sum = getSum(size);

const double sumE = getSumE(eps);

cout << "Итоговая сумма равна: " << sum << endl;

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

return 0;

}

catch (const exception& e)

{

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

return 1;

}

}

double getSum(const size\_t size) {

double current = 0;

double sum = 0;

for (int i = 0; i < size; i++) {

if (i == 0) {

current = 1;

}

else {

current \*= getRecurrentNumber(i - 1);

}

sum += current;

}

return sum;

}

double getSumE(const double eps) {

size\_t i = 0;

double sumE = 0;

double current = eps + 1;

while (abs(current) >= eps) {

if (i == 0) {

current = 1;

}

else {

current \*= getRecurrentNumber(i - 1);

}

sumE += current;

i++;

}

return sumE;

}

double getEpsilon(const string& message)

{

cout << message;

double eps = 0;

cin >> eps;

if (eps < 0)

{

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

}

return eps;

}

size\_t getSequenceSize(const string& message)

{

cout << message;

int size = 0;

cin >> size;

if (size < 1)

{

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

}

return size;

}

double getRecurrentNumber(const size\_t index) noexcept

{

return -1.0 / ((index + 3) \* (index + 4));

}

## **Решение тестовых примеров на С++**

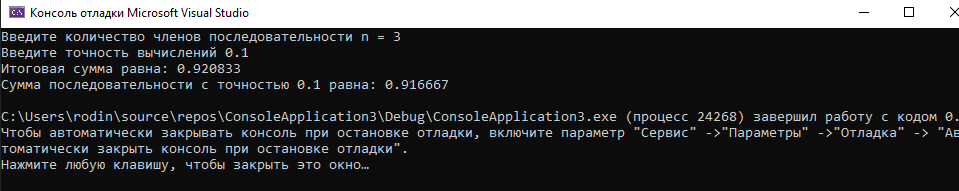


Рисунок 5 – Решение тестового примера для задания 3-2

## **Решение тестовых примеров в Excel**

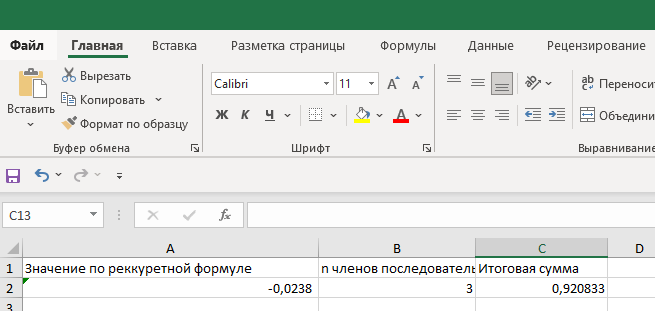


Рисунок – 6 Решение тестового примера для задания 3-2 в Excel