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 №3.1

По дисциплине«Процедурное программирование»

Выполнил: ст. гр.ТКИ – 111

Кох Е.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2021

**Оглавление**

[1. Условие задачи 3](#_Toc88677522)

[2. Блок-схема алгоритма 4](#_Toc88677524)

[3. Текст программы на языке C++ 5](#_Toc88677525)

[4. Результаты работы программы 6](#_Toc88677526)

1. Условие 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.

1. Блок-схема алгоритма

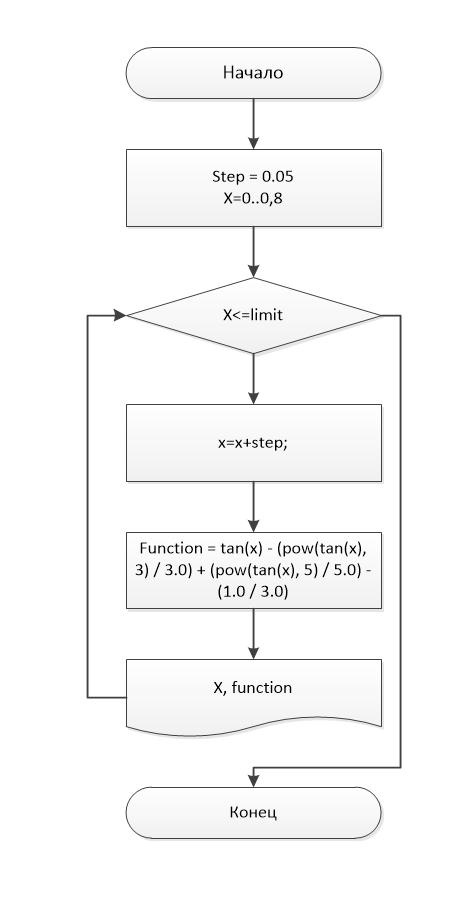


Рисунок1 -Блок-схема3.1

1. Текст программы на языке C++

|  |
| --- |
| #define \_USE\_MATH\_DEFINES  #include <iostream>  #include <cmath>  #include <iomanip>  using namespace std;  /\*\*  \* \brief Возможность расчета функции в точке.  \* \param x Входной параметр - точка x.  \* \return true, если функция определена в заданной точке x.  \*/  bool IsCalculated(const double x);  /\*\*  \*\brief Проверка аргумента.  \* \param x Входной параметр - точка x.  \* \return true, если аргумент входит в одз.  \*/  double Function(const double x);  /\*\*  \* \brief Точка входа в программу.  \* \return Код ошибки (0 - успех).  \*/  int main()  {  const auto left = 0.0;  const auto right = 0.8; // Границы промежутка  const auto step = 0.05; //Шаг функции  auto x = left;  cout << " x" << setw(15) << "y\n";  while ((x < right) || (abs(x - right) < step))  {  if (IsCalculated(x)  {  const auto y = Function(x);  cout << setw(10) << setprecision(2) << x << setw(15)  << setprecision(5) << y << '\n';  }  else  {  cout << setw(10) << setprecision(2) << x << setw(15)  << "error \n";  }  x += step;  }  return 0;  }  double Function(const double x)  {  return tan(x) - (pow(tan(x), 3) / 3.0) + (pow(tan(x), 5) / 5.0) - (1.0 / 3.0); // Расчёт функции  }  bool IsCalculated(const double x)  {  return abs(cos(x)) >= std::numeric\_limits<double>::min();  } // Проверка, больше ли число минимума типа double, если да - функция работает |

1. Результаты работы программы



Рисунок 2 -Вывод программы3.1