Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОМ ЗАДАНИИ**

Основные конструкции языка C#

тема

Преподаватель \_\_\_\_\_\_\_\_\_ А.А. Чикизов

подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 \_\_\_\_\_\_\_\_\_ Е.В. Железкин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

Содержание

[Содержание 2](#_Toc18956750)

[1 Цель работы 3](#_Toc18956751)

[2 Задача работы 3](#_Toc18956752)

[3 Ход работы 4](#_Toc18956753)

[4 Вывод 12](#_Toc18956754)

# Цель работы

Изучение основных конструкций языка C#.

# Задачи работы

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от Хнач до Хкон с шагом dx с точностью e. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.



# Ход работы

1. Был написал алгоритм, для нахождения значения арктангенса с помощью рядов Тейлора:

Листинг 1 – файл Program.cs

using System;  
  
namespace Lab1  
{  
 static class Program  
 {  
 private static double XStart { get; set; }  
 private static double XEnd { get; set; }  
 private static double Dx { get; set; }  
 private static double Precision { get; set; }  
   
 static void Main()  
 {  
 while (true)  
 {  
 try  
 {  
 System.Console.Write("Enter the start number: ");  
 XStart = double.Parse(System.Console.ReadLine()!);  
 System.Console.Write("Enter the end value: ");  
 XEnd = double.Parse(System.Console.ReadLine()!);  
 System.Console.Write("Enter dx: ");  
 Dx = double.Parse(System.Console.ReadLine()!);  
 System.Console.Write("Enter Precision: ");  
 Precision = double.Parse(System.Console.ReadLine()!);  
 break;  
 }  
 catch (Exception e)  
 {  
 System.Console.Error.WriteLine(e.Message);  
 }  
 }  
  
 System.Console.WriteLine($" {"x ", 10} | {"f(x) ", 20} | {"Sum(x) ", 20} | {"n ", 7} | {"Condition", 12} ");  
 for (var i = XStart; i <= XEnd; i += Dx)  
 {  
   
 Console.WriteLine($" {$"{i:F10}".TrimEnd('0').TrimEnd('.'), 10} | {$"{Math.Atan(i):F10}", 20} | " +  
 $"{$"{TaylorSeriesArccotangent.CalculateValue(i, Precision):F10}", 20} | " +  
 $"{$"{TaylorSeriesArccotangent.*I*}", 7} | " +  
 $"{TaylorSeriesArccotangent.*Condition*}");  
   
 }  
   
 }  
 }  
}

Листинг 2 – файл TaylorSeriesArccotangent.cs

using System;  
  
namespace Lab1  
{  
 public static class TaylorSeriesArccotangent  
 {  
   
 private const int *MaxN* = 100000;  
   
 private static double *\_result*;  
  
 public static string *Condition*;  
  
 public static int *I*;  
   
 private struct StepData  
 {  
 public double Top { get; set; }  
 public double Under { get; set; }  
 public bool Mark { get; set; }  
  
 public StepData(double top, double under, bool mark)  
 {  
 Top = top;  
 Under = under;  
 Mark = mark;  
 }  
 }  
  
 public static double CalculateValue(double x, double precision)  
 {  
 *\_result* = 0d;  
 var stepData = new StepData(x, 1d, true);  
 *I* = 0;  
  
 if (Math.Abs(x) >= 1)  
 {  
 *Condition* = "Неверное значение x!";  
 return *\_result*;  
 }  
  
 for (; *I* < *MaxN*; *I*++)  
 {  
 unsafe  
 {  
 var stepValue = CalculateCurrent(stepData);  
  
 if (Math.Abs(stepValue) < precision)  
 {  
 break;  
 }  
  
 *\_result* += stepValue;  
 *// Console.WriteLine($"{I}) {$"{\_result:F20}", 10}");* UpdateDataForNextStep(&stepData, x);  
 }  
 }  
  
 if (*I* == *MaxN*)  
 {  
 *Condition* = "Ошибка вычисления! (Превышено число итераций)";  
 *// throw new Exception("Задана слишком высокая точность!");* return 0d;  
 }  
  
 *Condition* = "Значение получено";  
 return *\_result*;  
 }  
  
 private static unsafe void UpdateDataForNextStep(StepData\* stepData, double x)  
 {  
 stepData->Top \*= x \* x;  
 stepData->Under += 2;  
 stepData->Mark = !stepData->Mark;  
 }  
  
 private static double CalculateCurrent(StepData stepData)  
 {  
 return stepData.Mark ? stepData.Top / stepData.Under : - stepData.Top / stepData.Under;  
 }  
   
 }  
}

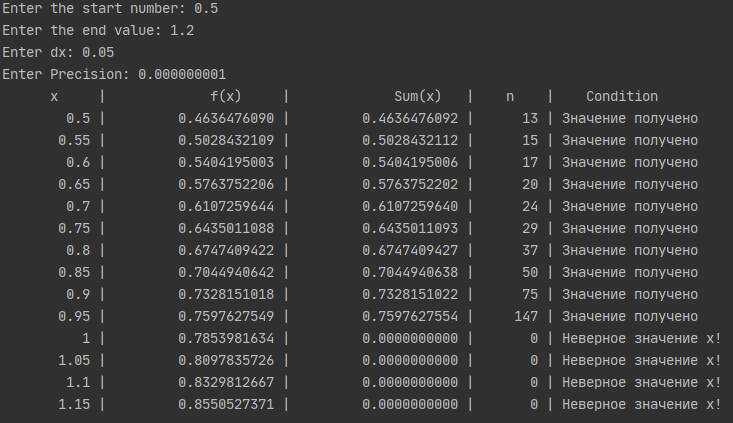


Рисунок 1 – Результат выполнения

# Вывод

В ходе данной лабораторной работы были изучены основные конструкции языка C#.