Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Основные конструкции языка C#

тема

Вариант 11

|  |  |  |
| --- | --- | --- |
| Преподаватель |  | А. А. Чикизов |
|  | подпись, дата | инициалы, фамилия |
| Студент КИ18-16б 031831229 |  | В. А. Прекель |
| номер группы, зачетной книжки | подпись, дата | инициалы, фамилия |

Красноярск 2020

# 1 Задание

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.

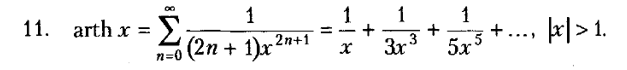


Рисунок 1 – Задание, вариант 11

# 2 Исходный код основного алгоритма

Листинг 1 – Lab01/CSharpLabs.Lab01.Core/InverseHyperbolicCotangent/ ArcothAvx.cs

using System;  
using System.Runtime.Intrinsics;  
using System.Runtime.Intrinsics.X86;  
  
namespace CSharpLabs.Lab01.Core.InverseHyperbolicCotangent  
{  
 public class ArcothAvx : AbstractArcoth  
 {  
 protected override unsafe double CalculateImpl(double x, double stepThreshold, int maxN)  
 {  
 if (!Avx.IsSupported)  
 {  
 Status = TaylorSeriesStatus.**NotSupported**;  
 return Double.**NaN**;  
 }  
  
 const int **vectorSize** = 256 / 8 / sizeof(double);  
  
 *// v8888 <- (8, 8, 8, 8)* var value8 = 8.0;  
 var v8888 = Avx.BroadcastScalarToVector256(&value8);  
  
 *// xPow8 <- (x^8, x^8, x^8, x^8)* var xPow8 = Avx.BroadcastScalarToVector256(&x);  
 xPow8 = Avx.Multiply(xPow8, xPow8);  
 xPow8 = Avx.Multiply(xPow8, xPow8);  
 xPow8 = Avx.Multiply(xPow8, xPow8);  
  
 *// up <- (x^(-1), x^(-3), x^(-5), x^(-7))* var upSa = stackalloc double[**vectorSize**];  
 var xDiv2iPlus1 = 1 / x;  
 for (var i = 0; i < **vectorSize**; i++)  
 {  
 upSa[i] = xDiv2iPlus1;  
 xDiv2iPlus1 /= x \* x;  
 }  
  
 var up = Avx.LoadVector256(upSa);  
  
 *// down <- (1, 3, 5, 7)* var downSa = stackalloc double[**vectorSize**] {1, 3, 5, 7};  
 var down = Avx.LoadVector256(downSa);  
  
 *// sum <- (0, 0, 0, 0)* var sum = Vector256<double>.Zero;  
  
 N = 0;  
 while (N < maxN)  
 {  
 *// div <- up / down* var div = Avx.Divide(up, down);  
 *// sum <- sum + div* sum = Avx.Add(sum, div);  
 *// div = (x1, x2, x3, last)* var last = div.GetElement(**vectorSize** - 1);  
 N += **vectorSize**;  
 if (Math.Abs(last) < stepThreshold)  
 {  
 break;  
 }  
  
 *// up <- up / (x^8, x^8, x^8, x^8)* up = Avx.Divide(up, xPow8);  
 *// down <- down + (8, 8, 8, 8)* down = Avx.Add(down, v8888);  
 }  
  
 var resultSa = stackalloc double[**vectorSize**];  
 Avx.Store(resultSa, sum);  
  
 Status = N >= maxN ? TaylorSeriesStatus.**TooManyIterations** : TaylorSeriesStatus.**Success**;  
  
 return resultSa[0] + resultSa[1] + resultSa[2] + resultSa[3];  
 }  
 }  
}

# 3 Результат

Листинг 2 – Запуск программы

Вычисление обратного гиперболического котангенса  
(arcth aka arcoth aka Inverse Hyperbolic Cotangent)  
с помощью ряда Тейлора  
  
Введите Xнач: -3  
Введите Xкон: 3  
Введите dx: 0,5  
Введите порог (в несколько раз меньше eps): 0,000001  
Доступные вычислители:  
1 ArcothAvx  
2 ArcothLinq  
3 ArcothNaive  
4 ArcothOptimized  
Введите номер вычислителя [ArcothAvx]: 1  
  
 x | f(x) | Σ(x) | n | Status  
 -3,0000000000 | -0,3465735903 | -0,3465735898 | 8 | Success  
 -2,5000000000 | -0,4236489302 | -0,4236489184 | 8 | Success  
 -2,0000000000 | -0,5493061443 | -0,5493061428 | 12 | Success  
 -1,5000000000 | -0,8047189562 | -0,8047188755 | 16 | Success  
 -1,0000000000 | -∞ | не число | -1 | NotInDomain  
 -0,5000000000 | не число | не число | -1 | NotInDomain  
 0,0000000000 | не число | не число | -1 | NotInDomain  
 0,5000000000 | не число | не число | -1 | NotInDomain  
 1,0000000000 | ∞ | не число | -1 | NotInDomain  
 1,5000000000 | 0,8047189562 | 0,8047188755 | 16 | Success  
 2,0000000000 | 0,5493061443 | 0,5493061428 | 12 | Success  
 2,5000000000 | 0,4236489302 | 0,4236489184 | 8 | Success  
  
Process finished with exit code 0.