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 — подпись, дата — А. А. Чикизов (инициалы, фамилия)

Студент КИ18-16б 031831229 (номер группы, зачетной книжки) — подпись, дата — В. А. Прекель (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.

$$11.\quad \operatorname{arth} x = \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = \frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \ldots, \quad |x| > 1.$$

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 Хнач: -3
Введите Хкон: 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.
```

```
BenchmarkDotNet=v0.12.1, OS=Windows 10.0.19042
AMD Ryzen 3 1200, 1 CPU, 4 logical and 4 physical cores
.NET Core SDK=3.1.400
  [Host]     : .NET Core 3.1.6 (CoreCLR 4.700.20.26901, CoreFX 4.700.20.31603), X64 RyuJIT
  DefaultJob : .NET Core 3.1.6 (CoreCLR 4.700.20.26901, CoreFX 4.700.20.31603), X64 RyuJIT
```

Method	Calc	Mean	Error	StdDev
Benchmark1	CShar(...)thAvx [58]	121.1 ns	0.43 ns	0.38 ns
Benchmark2	CShar(...)thAvx [58]	522.8 ns	3.75 ns	3.51 ns
Benchmark1	CShar(...)hLinq [59]	4,950.5 ns	17.08 ns	15.98 ns
Benchmark2	CShar(...)hLinq [59]	24,553.9 ns	92.94 ns	86.94 ns
Benchmark1	CShar(...)Naive [60]	2,481.9 ns	10.68 ns	9.99 ns
Benchmark2	CShar(...)Naive [60]	12,504.7 ns	46.38 ns	43.38 ns
Benchmark1	CShar(...)mized [64]	323.0 ns	1.22 ns	1.14 ns
Benchmark2	CShar(...)mized [64]	1,562.0 ns	7.13 ns	6.32 ns

Рисунок 2 – Результаты бенчмарка AVX-реализации и прочих