Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №2**

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ ВОЗМОЖНОСТИ 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)

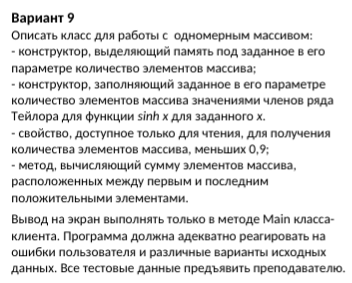
[3 Ход работы 3](#_Toc18956753)

[4 Вывод 6](#_Toc18956754)

# Цель работы

Изучение объектно-ориентированных возможностей языка C#.

# Задачи работы



# Ход работы

1. Был написал алгоритм, для нахождения значения гиперболического синуса с помощью рядов Макларена (Приложение А):

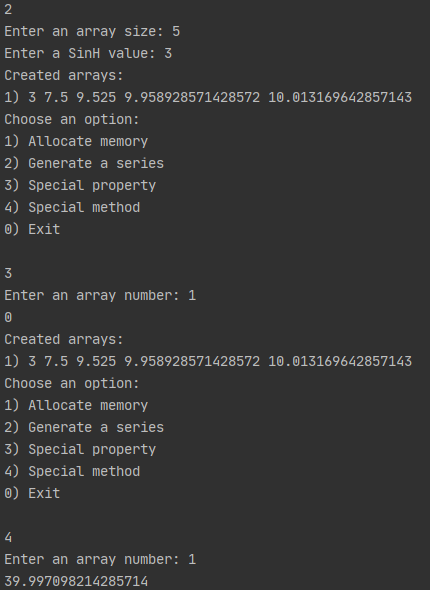


Рисунок 1 – Результат выполнения

# Вывод

В ходе данной лабораторной работы были изучены объектно-ориентированные возможности языка C#.

**ПРИЛОЖЕНИЕ А**

Листинг 1 – файл Program.cs

using System;  
using System.Collections.Generic;  
  
namespace Lab\_2  
{  
 class Program  
 {  
 static void Main()  
 {  
   
 var series = new List<ContainingArray>();  
  
 while (true)  
 {  
 try  
 {  
 if (series.Count > 0)  
 {  
 Console.Write("Created arrays: \n");  
 foreach (var item in series)  
 {  
 Console.Write($"{series.IndexOf(item) + 1}) ");  
 foreach (var value in item.SeriesArray)  
 {  
 Console.Write($"{value} ");   
 }  
 Console.WriteLine();  
 }  
 }  
  
 Console.Write("Choose an option:\n1) Allocate memory\n2) Generate a series\n" +  
 "3) Special property\n4) Special method\n0) Exit\n\n");  
 var option = int.Parse(Console.ReadLine()!);  
  
 switch (option)  
 {  
 case 1:  
 Console.Write("Enter an array size: ");  
 var tempSize = int.Parse(Console.ReadLine()!);  
 series.Add(new ContainingArray(tempSize));  
 break;  
 case 2:  
 Console.Write("Enter an array size: ");  
 tempSize = int.Parse(Console.ReadLine()!);  
 Console.Write("Enter a SinH value: ");  
 var tempX = double.Parse(Console.ReadLine()!);  
 series.Add(new ContainingArray(tempSize, tempX));  
 break;  
 case 3:  
 Console.Write("Enter an array number: ");  
 tempSize = int.Parse(Console.ReadLine()!);  
 var counter = 0;  
 ContainingArray currentEl = null;  
 foreach (var item in series)  
 {  
 if (counter < tempSize - 1)  
 counter++;  
 else  
 {  
 currentEl = item;  
 break;  
 }  
 }  
 Console.WriteLine(currentEl.GetSum);  
 break;  
 case 4:  
 Console.Write("Enter an array number: ");  
 tempSize = int.Parse(Console.ReadLine()!);  
 counter = 0;  
 currentEl = null;  
 foreach (var item in series)  
 {  
 if (counter < tempSize - 1)  
 counter++;  
 else  
 {  
 currentEl = item;  
 break;  
 }  
 }  
 Console.WriteLine(currentEl.CalculateSpecialSum());  
 break;  
 case 0:  
 return;  
 }  
 }  
 catch (Exception e)  
 {  
 Console.Error.WriteLine(e.Message);  
 }  
 }  
 }  
 }  
}

Листинг 2 – файл ContainingArray.cs

using System;  
using System.Linq;  
  
namespace Lab\_2  
{  
 public class ContainingArray  
 {  
 public readonly double[] SeriesArray;  
 private static long Factorial(long x)  
 {  
 return (x == 0) ? 1 : x \* Factorial(x - 1);  
 }  
   
 public double GetSum  
 {  
 get  
 {  
 return SeriesArray.Count(value => value <= 0.9);  
 *// var result = 0;  
 // foreach (var value in \_seriesArray)  
 // {  
 // if (value <= 0.9) result++;  
 // }  
 // return result;* }  
 }  
  
 public ContainingArray(int count)  
 {  
 SeriesArray = new double[count];  
 }  
  
 public ContainingArray(int count, double x)  
 {  
 var currentUp = x;  
 var currentDown = 1;  
 var result = 0d;  
   
 SeriesArray = new double[count];  
  
 for (var i = 0; i < count; i++)  
 {  
 result += currentUp / Factorial(currentDown);  
  
 try  
 {  
 SeriesArray[i] = result;  
 }  
 catch (Exception e)  
 {  
 Console.Out.WriteLine(e.Message);  
 }  
  
 currentUp \*= x \* x;  
 currentDown += 2;  
 }  
 }  
  
 public double CalculateSpecialSum()  
 {  
 var flag = 0;  
 var firstPos = -1;  
 var lastPos = -1;  
 var result = 0d;  
   
   
 for (var i = 0; i < SeriesArray.Length; i++)  
 {  
 if (flag < 2 && SeriesArray[i] > 0 && firstPos == -1)  
 {  
 firstPos = i;  
 flag++;  
 }  
  
 if (flag < 2 && SeriesArray[SeriesArray.Length - i - 1] > 0 && lastPos == -1)  
 {  
 lastPos = SeriesArray.Length - i - 1;  
 flag++;  
 }  
  
 if (flag == 2) break;  
 }  
  
 for (var i = firstPos; i <= lastPos; i++)  
 {  
 result += SeriesArray[i];  
 }  
  
 return result;  
 }  
   
 }  
}