**Шумилов Лев, РИС-20-1б, Вариант №25**

**Отчёт по практической работе №3 «Вычисление функция с использованием их разложения в степенной ряд»**

**Таблица с задачами:**

|  |  |  |
| --- | --- | --- |
| Базовая функция | Ряд Тейлора | Доп. условия |
|  |  | *N = 15,*  *0.1 <= x <= 1* |

**Задание** состоит их четырёх задач:

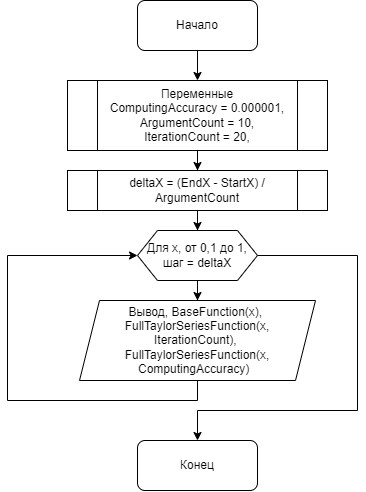
1. Реализация метода, выполняющего вычисление значения базовой функции;
2. Реализация метода, выполняющего вычисление значение функции Тейлора с рекуррентным соотношением, через количество итераций;
3. Реализация метода, выполняющего вычисление значение функции Тейлора с рекуррентным соотношением, через заданную точность вычислений;
4. Основная программа, которая сравнивает результаты работы этих функций;

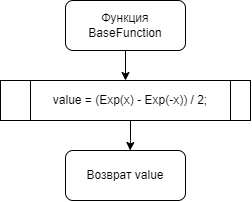
**Анализ задачи:**

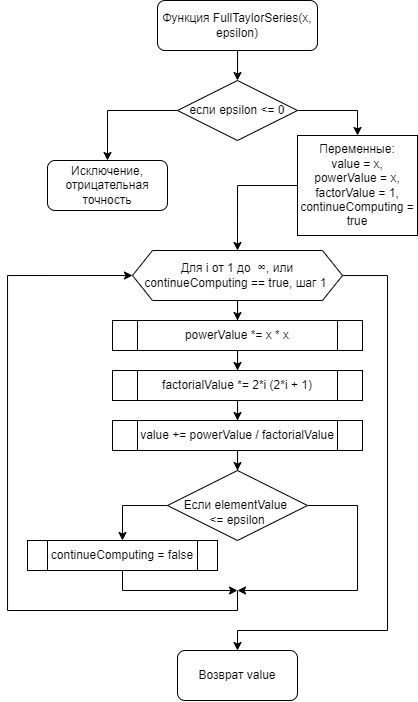
Исходя из условий, программа должна в цикле перебирать значения x, и подставлять их в реализованные функции. После того, как значения всех функций были вычислены, полученные результаты должны отправляться на печать.

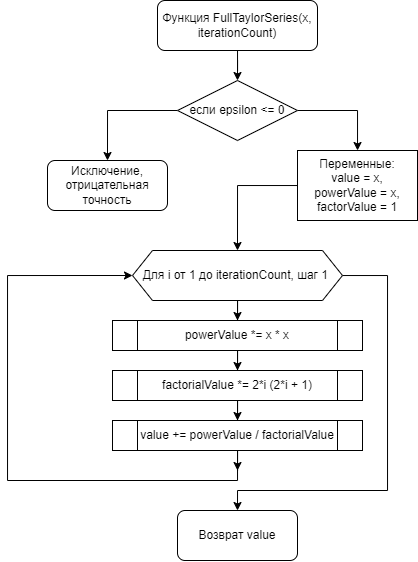
Главной особенностью рекуррентного соотношения, является то, что значение текущего элемента зависит от значения предыдущего, т.е. чтобы получить 2 в 4 степени, необходимо иметь 2 в 3 степени и умножить его на два, а не пересчитывать 2 в 3 заново с единицы.

Именно за счёт данной особенности – отсутствия необходимости пересчитывать значения заново, сложность программы сокращается с О(n^2) до O(n), что на большом количестве итераций может быть значительно.









Исходный код программы на языке C#:

using System;

using static System.Console;

using static System.Math;

namespace LaboratoryWorkNo3

{

class Program

{

static double BaseFunction(double x)

{

double value = (Exp(x) - Exp(-x)) / 2;

return value;

}

static double FullTaylorSeriesFunction(double x, double epsilon)

{

if (epsilon <= 0)

throw new ArgumentException("Точность не может быть <= 0");

double value = x;

double powerValue = x;

long factorialValue = 1;

var continueCompitung = true;

for (int i = 1; continueCompitung; i++)

{

powerValue \*= x \* x;

factorialValue \*= 2 \* i \* (2 \* i + 1);

if (factorialValue == 0)

factorialValue = long.MaxValue;

double elementValue = powerValue / factorialValue;

value += elementValue;

if (elementValue <= epsilon)

continueCompitung = false;

}

return value;

}

static double FullTaylorSeriesFunction(double x, int iterationCount)

{

if (iterationCount <= 0)

throw new ArgumentException("Кол-во итераций не может быть <= 0");

double value = x;

double powerValue = x;

long factorialValue = 1;

for (int i = 1; i <= iterationCount; i++)

{

powerValue \*= x \* x;

factorialValue \*= 2 \* i \* (2 \* i + 1);

if (factorialValue == 0)

factorialValue = long.MaxValue;

double elementValue = powerValue / factorialValue;

value += elementValue;

}

return value;

}

static void Main(string[] args)

{

const double StartX = 0.1;

const double EndX = 1;

const double ComputingAccuracy = 0.000001;

const int ArgumentCount = 10;

const int IterationCount = 20;

double deltaX = (EndX - StartX) / ArgumentCount;

WriteLine("Сокращения:\n" +

$"Base\_Fun - Базовая функция,\n" +

$"TS\_Iters - Ряд тейлора, кол-во итераций = {IterationCount},\n" +

$"TS\_Epsil - Ряд тейлора, точность = {ComputingAccuracy},\n" +

$"Интервал - [{StartX}; {EndX}]\n");

for (double x = StartX; x <= EndX; x += deltaX)

WriteLine($" X: {x:N4} => " +

$"Base\_Fun: {BaseFunction(x):N8}; " +

$"TS\_Iters: {FullTaylorSeriesFunction(x, IterationCount):N8} " +

$"TS\_Epsil: {FullTaylorSeriesFunction(x, ComputingAccuracy):N8};" );

}

}

}