Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Вычисление математических функций с использованием рядов»**

**Выполнил**:

студент группы 3824Б1ПМ1-1

Безносов А. С.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2025

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 8](#_Toc26962566)

[Результаты экспериментов 9](#_Toc26962567)

[Заключение 18](#_Toc26962568)

[Приложение 19](#_Toc26962569)

# Постановка задачи

Необходимо реализовать подсчет математических функций: через ряды Тейлора. Также необходимо сравнить значения функций методом подсчета через разные методы суммирования и через библиотечные функции в math.h.

# Метод решения

В этой работе будет рассматриваться три метода суммирования рядов - прямое, обратное и попарное. Для подсчета членов ряда используются рекуррентные формулы, которые позволяют не вычислять факториал в знаменателе, упрощая вычисления.

: ,

: ,

: ,

: ,

Для подсчета членов ряда для соответствующих математических функций были написаны функции вида ComputeTerms<имя функции>. На вход они принимают 3 аргумента: массив элементов ряда ArrTerms типа float, значение функции x типа float и число элементов ряда count типа int. В начале каждой из функции производится проверка входных данных. Если count ≤ 0, то функция завершает свою работу. Далее (только для тригонометрических функций) вычисляется остаток при делении x на 2π, что позволяет не вычислять значение функции при очень большом аргументе. Потом задается первый элемент ряда (нулевой элемент массива ArrTerms) и в цикле рекуррентно вычисляются остальные элементы, которые добавляются в массив. Функции ничего не возвращают.

Для подсчета ряда методом прямого суммирования использовалась функция DirectSummation. На вход принимается два аргумента: массив элементов ряда ArrTerms типа float и число элементов ряда count типа int. Далее в цикле к итоговой сумме SumRes прибавляется элементы из ArrTerms. Функция возвращает SumRes. Реализация обратного суммирования аналогична, отличие лишь в том, что к итоговой сумме элементы прибавляются в обратном порядке. Для попарного суммирования были написаны две функции. Первая ReverseSummation принимает 3 аргумента: массив элементов ряда ArrTerms типа float и индексы начала start и конца end массива типа int. Далее производится проверка базового случая – если start совпадает с end, то функция возвращает элемент массива с индексом start. Вычисляется середина массива mid. Сама функция рекурсивно делит массив на две части и суммирует их. Вторая функция ReverseSummatinRes ничего не делает, она лишь возвращает значение функции ReverseSummation при end = count – 1, что позволяет не выходить за пределы массива.

# Руководство пользователя

Для работы с программой необходимо сначала выбрать функцию (рис. 1). Далее выбирается метод суммирования (рис. 2). Потом вводится значение аргумента функции (рис. 3) и количество членов ряда (рис. 4). После ввода данных, на экран выведется итоговое значение функции, которое было подсчитано с помощью моей реализации, и реализации через библиотеку math.h, а также разность по модулю между величинами (рис. 5).

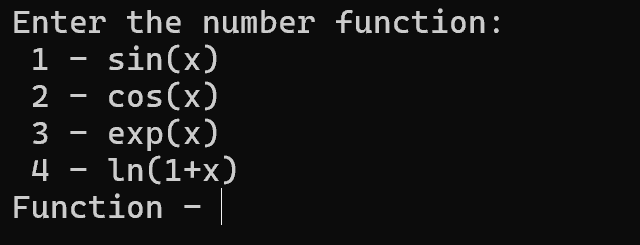


Рис. 1 Выбор функции

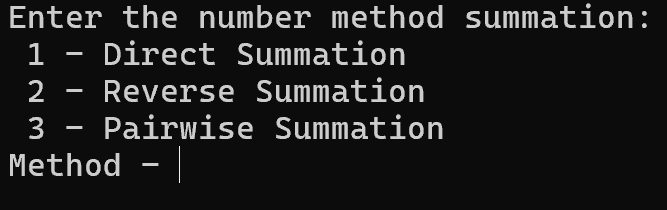


Рис. 2 Выбор метода суммирования



Рис. 3 Ввод значения аргумента функции



Рис. 4 Ввод числа членов ряда

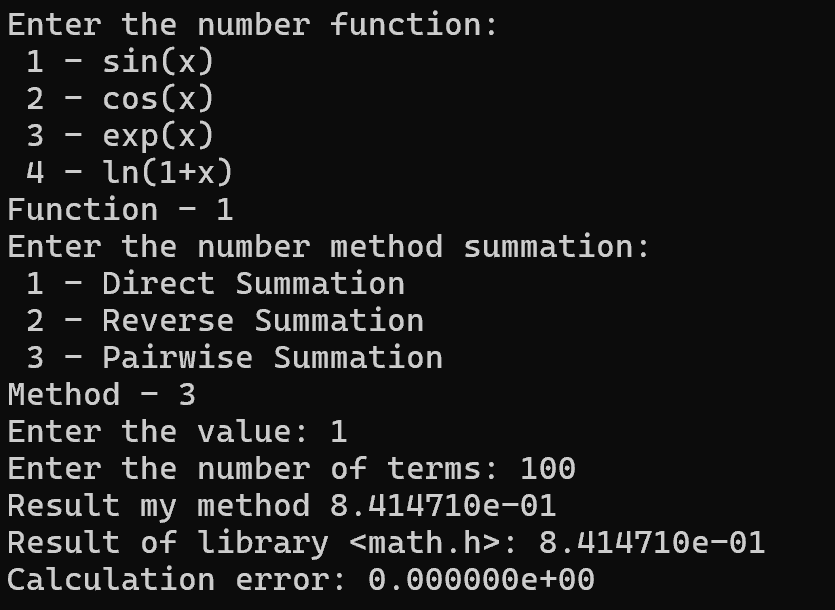


Рис. 5 Вывод

# Описание программной реализации

Программа содержит в себе 3 файла. В файле “main.cpp” содержится основная функция main, где объявлены переменные и вызов функции для выведения результатов. Также в main написан интерфейс программы. В файле “Header.h” написаны прототипы функций. Файл “Source.cpp” включает в себя все функции, отвечающие за подсчет и вывод результатов.

Для работы вспомогательных функций определяются два указателя на функции. Первый представляет собой функцию для суммирования членов ряда, а второй – для вычисления членов ряда.

Функция ComputeTaylorRow вычисляет сумму ряда. Она принимает 4 параметра: значение x типа float, количество членов ряда count типа int, указатели на функции ComputeTerms и SummationMethod типа ComputeTermsFunction и SummationFunction соответственно. Внутри функции создается массив для хранения членов ряда. Затем вызывается функция ComputeTerms, которая заполняет массив членами ряда соответствующей математической функции. После вызывается функция SummationMethod, суммирующая элементы массива определенными образом. Функция возвращает итоговую сумму.

Функция CompareAndOutputResults сравнивает результаты, полученные от функции ComputeTaylorRow и функций из math.h. Принимает на вход также 4 параметра: вид функции FunctionType типа int, метод суммирования SummationMethod типа int, значение x типа float и число членов ряда count типа int. В функции определены три массива: ComputeTermsFunctions типа ComputeTermsFunction, хранящий функции для вычисления членов ряда, SummationMethods типа SummationFunction, хранящий функции для суммирования различными методами, и MathLibFunctions типа float, хранящий функции из math.h. Далее переменной MyResult присваивается значение функции ComputeTaylorRow, которая вычисляет сумму ряда в зависимости от параметров, введенных пользователем. Затем переменной MathLibResult присваивается значение функции MathLibFunction. Значение обоих переменных и их разность по модулю выводится на экран.

Прототипы функций, реализованных в программе:

void ComputeTermsSin(float\* ArrTerms, int count, float x);

void ComputeTermsCos(float\* ArrTerms, int count, float x);

void ComputeTermsExp(float\* ArrTerms, int count, float x);

void ComputeTermsLn(float\* ArrTerms, int count, float x);

float DirectSummation(float\* ArrTerms, int count);

float ReverseSummation(float\* ArrTerms, int count);

float PairwiseSummation(float\* ArrTerms, int start, int end);

float PairwiseSummationRes(float\* ArrTerms, int count);

typedef float (\*SummationFunction)(float\* x, int count);

typedef void (\*ComputeTermsFunction)(float\* terms, int count, float x);

float ComputeSeries(float x, int count, ComputeTermsFunction ComputeTerms, SummationFunction SummationMethod);

void CompareAndOutputResults(int functionType, int summationMethod, float x, int count);

# Подтверждение корректности

Для подтверждения корректности в программе использовались математические функции из библиотеки math.h. В результате работы функции CompareAndOutputResults выводится значение функции через мою и библиотечные реализации, а также разность по модулю между этими величинами.

# Результаты экспериментов

Для проведения эксперимента рассмотрим погрешность между нашей и библиотечной реализациями значений функций в конкретной точке. Для функций возьмем , а для функции возьмем , так как при больших значениях (конкретно при ) ряд будет расходится. Я рассмотрел все необходимые способы суммирования с разным числом слагаемых.

Все необходимые данные представлены в таблицах и на графиках. Ось абсцисс отвечает за число слагаемых, а ось ординат – за погрешность.

1. **Прямое суммированиеe**

Таблица пао

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 4,260835647583000 |
| 2 | 4,296950340270990 |
| 3 | 1,614216327667230 |
| 4 | 0,330094575881958 |
| 5 | 0,042962312698364 |
| 6 | 0,003889262676239 |
| 7 | 0,000259697437286 |
| 8 | 0,000013232231140 |
| 9 | 0,000000655651093 |
| 10 | 0,000000119209290 |
| 11 | 0,000000119209290 |
| 12 | 0,000000119209290 |
| 13 | 0,000000119209290 |
| 14 | 0,000000119209290 |
| 15 | 0,000000119209290 |
| 16 | 0,000000119209290 |
| 17 | 0,000000119209290 |
| 18 | 0,000000119209290 |
| 19 | 0,000000119209290 |
| 20 | 0,000000119209290 |

Рис. 1.1 данные для sin(x)

На рис. 1.1 представлен график зависимости погрешности от числа слагаемых для функции . Видно, что уже при 5 слагаемых точность очень высока и практически не отличается от “эталонной”.

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 1,839071512222290 |
| 2 | 5,068284034729000 |
| 3 | 2,883642911911010 |
| 4 | 0,778142452239990 |
| 5 | 0,125187993049622 |
| 6 | 0,013470292091370 |
| 7 | 0,001041233539581 |
| 8 | 0,000060260295868 |
| 9 | 0,000003159046173 |
| 10 | 0,000000298023224 |
| 11 | 0,000000417232513 |
| 12 | 0,000000417232513 |
| 13 | 0,000000417232513 |
| 14 | 0,000000417232513 |
| 15 | 0,000000417232513 |
| 16 | 0,000000417232513 |
| 17 | 0,000000417232513 |
| 18 | 0,000000417232513 |
| 19 | 0,000000417232513 |
| 20 | 0,000000417232513 |

Рис. 2.1 данные для cos(x)

Также, как и с синусом, ряд с косинусом очень быстро сходится (см. рис. 2.1).

Рис. 3.3 данные для exp(x)

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 22025,4648437500 |
| 2 | 22015,4648437500 |
| 3 | 21965,4648437500 |
| 4 | 21798,7988281250 |
| 5 | 21382,1308593750 |
| 6 | 20548,7988281250 |
| 7 | 19159,9101562500 |
| 8 | 17175,7832031250 |
| 9 | 14695,6250000000 |
| 10 | 11939,8935546875 |
| 11 | 9184,1621093750 |
| 12 | 6678,9511718750 |
| 13 | 4591,2753906250 |
| 14 | 2985,3710937500 |
| 15 | 1838,2968750000 |
| 16 | 1073,5800781250 |
| 17 | 595,6328125000 |
| 18 | 314,4863281250 |
| 19 | 158,2949218750 |
| 20 | 76,0878906250 |

С экспонентой ситуация несколько иная. Даже ряд с 20 слагаемыми далек от истинного значения. На рис. 3.1 видно, что кривая очень плавно опускается и прижимается к оси абсцисс, по сравнению с двумя другими функциями, которые мы уже рассмотрели.

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,306852817535400 |
| 2 | 0,193147182464600 |
| 3 | 0,140186190605164 |
| 4 | 0,109813809394836 |
| 5 | 0,090186178684235 |
| 6 | 0,076480507850647 |
| 7 | 0,066376626491547 |
| 8 | 0,058623373508453 |
| 9 | 0,052487730979919 |
| 10 | 0,047512292861938 |
| 11 | 0,043396830558777 |
| 12 | 0,039936482906342 |
| 13 | 0,036986589431763 |
| 14 | 0,034442007541656 |
| 15 | 0,032224655151367 |
| 16 | 0,030275344848633 |
| 17 | 0,028548181056976 |
| 18 | 0,027007400989532 |
| 19 | 0,025624155998230 |
| 20 | 0,024375855922699 |

Рис. 4.1 данные для ln(1+x)

График для логарифма также плавно опускается (см. рис. 4.1). Точность достаточно высокая.

1. **Обратное суммирование**

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 4,260835647583000 |
| 2 | 4,296950340270990 |
| 3 | 1,614216327667230 |
| 4 | 0,330094337463379 |
| 5 | 0,042962074279785 |
| 6 | 0,003889083862305 |
| 7 | 0,000259399414062 |
| 8 | 0,000013351440430 |
| 9 | 0,000000476837158 |
| 10 | 0,000000000000000 |
| 11 | 0,000000000000000 |
| 12 | 0,000000000000000 |
| 13 | 0,000000000000000 |
| 14 | 0,000000000000000 |
| 15 | 0,000000000000000 |
| 16 | 0,000000000000000 |
| 17 | 0,000000000000000 |
| 18 | 0,000000000000000 |
| 19 | 0,000000000000000 |
| 20 | 0,000000000000000 |

Рис. 1.2 данные для sin(x)

На рис. 1.2 видно, что при 10 слагаемых значение ряда абсолютно не отличается от истинного значения.

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 1,839071512222290 |
| 2 | 5,068284034729000 |
| 3 | 2,883642911911010 |
| 4 | 0,778142690658569 |
| 5 | 0,125188112258911 |
| 6 | 0,013470411300659 |
| 7 | 0,001041173934937 |
| 8 | 0,000060319900513 |
| 9 | 0,000003099441528 |
| 10 | 0,000000238418579 |
| 11 | 0,000000238418579 |
| 12 | 0,000000238418579 |
| 13 | 0,000000238418579 |
| 14 | 0,000000238418579 |
| 15 | 0,000000238418579 |
| 16 | 0,000000238418579 |
| 17 | 0,000000238418579 |
| 18 | 0,000000238418579 |
| 19 | 0,000000238418579 |
| 20 | 0,000000238418579 |

Точность такая же высокая, ряд с 8 слагаемыми недалек от истины (см. рис. 2.2).

Рис. 3.2 данные для exp(x)

Рис. 2.2 данные для cos(x)

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 22025,4648437500 |
| 2 | 22015,4648437500 |
| 3 | 21965,4648437500 |
| 4 | 21798,7988281250 |
| 5 | 21382,1308593750 |
| 6 | 20548,7988281250 |
| 7 | 19159,9101562500 |
| 8 | 17175,7832031250 |
| 9 | 14695,6250000000 |
| 10 | 11939,8925781250 |
| 11 | 9184,1601562500 |
| 12 | 6678,9501953125 |
| 13 | 4591,2753906250 |
| 14 | 2985,3710937500 |
| 15 | 1838,2968750000 |
| 16 | 1073,5800781250 |
| 17 | 595,6328125000 |
| 18 | 314,4863281250 |
| 19 | 158,2949218750 |
| 20 | 76,0878906250 |

Видно, что точность никак не увеличилась, тенденция плавного уменьшения сохраняется (см. рис. 3.2).

Рис. 4.2 данные для ln(1+x)

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,306852817535400 |
| 2 | 0,193147182464600 |
| 3 | 0,140186190605164 |
| 4 | 0,109813809394836 |
| 5 | 0,090186178684235 |
| 6 | 0,076480507850647 |
| 7 | 0,066376626491547 |
| 8 | 0,058623373508453 |
| 9 | 0,052487730979919 |
| 10 | 0,047512233257294 |
| 11 | 0,043396830558777 |
| 12 | 0,039936482906342 |
| 13 | 0,036986589431763 |
| 14 | 0,034441947937012 |
| 15 | 0,032224655151367 |
| 16 | 0,030275344848633 |
| 17 | 0,028548240661621 |
| 18 | 0,027007341384888 |
| 19 | 0,025624275207520 |
| 20 | 0,024375796318054 |

Ситуация качественно не изменилась (см. рис. 4.2).

1. **Попарное суммирование**

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 4,260835647583000 |
| 2 | 4,296950340270990 |
| 3 | 1,614216327667230 |
| 4 | 0,330094575881958 |
| 5 | 0,042962312698364 |
| 6 | 0,003889203071594 |
| 7 | 0,000259697437286 |
| 8 | 0,000013232231140 |
| 9 | 0,000000655651093 |
| 10 | 0,000000059604645 |
| 11 | 0,000000178813934 |
| 12 | 0,000000178813934 |
| 13 | 0,000000059604645 |
| 14 | 0,000000059604645 |
| 15 | 0,000000059604645 |
| 16 | 0,000000059604645 |
| 17 | 0,000000119209290 |
| 18 | 0,000000119209290 |
| 19 | 0,000000059604645 |
| 20 | 0,000000059604645 |

Рис. 1.3 данные для sin(x)

График имеет аналогичный двум предыдущим примерам вид (см. рис. 1.3).

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 1,839071512222290 |
| 2 | 5,068284034729000 |
| 3 | 2,883642911911010 |
| 4 | 0,778142690658569 |
| 5 | 0,125188112258911 |
| 6 | 0,013470172882080 |
| 7 | 0,001040995121002 |
| 8 | 0,000060498714447 |
| 9 | 0,000003218650818 |
| 10 | 0,000000357627869 |
| 11 | 0,000000536441803 |
| 12 | 0,000000536441803 |
| 13 | 0,000000119209290 |
| 14 | 0,000000119209290 |
| 15 | 0,000000119209290 |
| 16 | 0,000000119209290 |
| 17 | 0,000000476837158 |
| 18 | 0,000000476837158 |
| 19 | 0,000000476837158 |
| 20 | 0,000000476837158 |

Рис. 2.3 данные для cos(x)

Из графика (см. рис. 2.3) и таблицы видно, что ряд с 8 слагаемыми показывает очень высокую точность.

Рис. 3.3 данные для exp(x)

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 22025,4648437500 |
| 2 | 22015,4648437500 |
| 3 | 21965,4648437500 |
| 4 | 21798,7988281250 |
| 5 | 21382,1308593750 |
| 6 | 20548,7988281250 |
| 7 | 19159,9101562500 |
| 8 | 17175,7832031250 |
| 9 | 14695,6250000000 |
| 10 | 11939,8935546875 |
| 11 | 9184,1621093750 |
| 12 | 6678,9511718750 |
| 13 | 4591,2753906250 |
| 14 | 2985,3710937500 |
| 15 | 1838,2968750000 |
| 16 | 1073,5800781250 |
| 17 | 595,6328125000 |
| 18 | 314,4863281250 |
| 19 | 158,2929687500 |
| 20 | 76,0878906250 |

Вид графика (см. рис. 3.3) практически не отличается от двух предыдущих примеров с экспонентой.

Рис. 4.3 данные для ln(1+x)

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,306852817535400 |
| 2 | 0,193147182464600 |
| 3 | 0,140186190605164 |
| 4 | 0,109813809394836 |
| 5 | 0,090186178684235 |
| 6 | 0,076480507850647 |
| 7 | 0,066376686096191 |
| 8 | 0,058623313903809 |
| 9 | 0,052487790584564 |
| 10 | 0,047512233257294 |
| 11 | 0,043396830558777 |
| 12 | 0,039936482906342 |
| 13 | 0,036986649036407 |
| 14 | 0,034441947937012 |
| 15 | 0,032224714756012 |
| 16 | 0,030275285243988 |
| 17 | 0,028548240661621 |
| 18 | 0,027007281780243 |
| 19 | 0,025624275207520 |
| 20 | 0,024375736713409 |

Аналогичная ситуация (см. рис. 4.3), как и с двумя предыдущими примерами.

1. **Вывод**

Проанализировав все методы суммирования, я пришел к выводу, что все отличия в них начинаются ближе к 7 знаку после запятой, что и характерно для типа данных float. Для синуса самый лучший метод – обратный. Для косинуса - обратный. Для экспоненты не получилось выявить явного фаворита, ведь данные идентичны. Для логарифма – попарный.

# Заключение

Мы реализовали подсчет необходимых математических функций через ряды Тейлора и сравнили значения функций через различные методы суммирования и через библиотечные функции в math.h.

# Приложение

#include <stdio.h>

#include <stdlib.h>

#define \_USE\_MATH\_DEFINES

#include <math.h>

#define PI M\_PI

#define COUNT\_FUNCTION\_TYPE 4

#define COUNT\_SUMMATION\_METHOD 3

//sin(x)

void ComputeTermsSin(float\* ArrTerms, int count, float x) {

if (count <= 0) {

return;

}

x = fmodf(x, 2.0f \* (float)PI);

ArrTerms[0] = x;

for (int n = 1; n < count; n++) {

ArrTerms[n] = ArrTerms[n - 1] \* (-x \* x) / ((2 \* n + 1) \* (2 \* n));

}

}

//cos(x)

void ComputeTermsCos(float\* ArrTerms, int count, float x) {

if (count <= 0) {

return;

}

x = fmodf(x, 2.0f \* (float)PI);

ArrTerms[0] = 1.0f;

for (int n = 1; n < count; n++) {

ArrTerms[n] = ArrTerms[n - 1] \* (-x \* x) / ((2 \* n) \* (2 \* n - 1));

}

}

//exp(x)

void ComputeTermsExp(float\* ArrTerms, int count, float x) {

if (count <= 0) {

return;

}

ArrTerms[0] = 1.0f;

for (int n = 1; n < count; n++) {

ArrTerms[n] = ArrTerms[n - 1] \* x / n;

}

}

//ln(1 + x)

void ComputeTermsLn(float\* ArrTerms, int count, float x) {

if (count <= 0) {

return;

}

ArrTerms[0] = x;

for (int n = 1; n < count; n++) {

ArrTerms[n] = ArrTerms[n - 1] \* (-x) \* n / (n + 1);

}

}

//ПРЯМОЕ СУММИРОВАНИЕ

float DirectSummation(float\* ArrTerms, int count) {

float SumRes = 0.0f;

for (int i = 0; i < count; i++) {

SumRes += ArrTerms[i];

}

return SumRes;

}

//ОБРАТНОЕ СУММИРОВАНИЕ

float ReverseSummation(float\* ArrTerms, int count) {

float SumRes = 0.0f;

for (int i = count - 1; i >= 0; i--) {

SumRes += ArrTerms[i];

}

return SumRes;

}

//ПОПАРНОЕ СУММИРОВАНИЕ

float PairwiseSummation(float\* ArrTerms, int start, int end) {

if (start == end) {

return ArrTerms[start];

}

int mid = (start + end) / 2;

return PairwiseSummation(ArrTerms, start, mid) + PairwiseSummation(ArrTerms, mid + 1, end);

}

float PairwiseSummationRes(float\* ArrTerms, int count) {

if (count <= 0) {

return 0.0f;

}

return PairwiseSummation(ArrTerms, 0, count - 1);

}

typedef float (\*SummationFunction)(float\* x, int count);

typedef void (\*ComputeTermsFunction)(float\* terms, int count, float x);

//ВЫЧИСЛЕНИЕ СУММЫ РЯДА

float ComputeTaylorRow(float x, int count, ComputeTermsFunction ComputeTerms, SummationFunction SummationMethod) {

float\* ArrTerms = (float\*)malloc(sizeof(float) \* count);

if (ArrTerms == NULL) {

return 0.0f;

}

ComputeTerms(ArrTerms, count, x);

float Res = SummationMethod(ArrTerms, count);

free(ArrTerms);

return Res;

}

//ВЫВОД РЕЗУЛЬТАТОВ

void CompareAndOutputResults(int FunctionType, int SummationMethod, float x, int count) {

if (FunctionType <= 0 || SummationMethod <= 0 || FunctionType > COUNT\_FUNCTION\_TYPE || SummationMethod > COUNT\_SUMMATION\_METHOD) {

printf("Invalid type of function or method of summation!");

return;

}

//МАССИВ ФУНКЦИЙ ДЛЯ ПОДСЧЕТА ЧЛЕНОВ РЯДА

ComputeTermsFunction ComputeTermsFunctions[] = {

NULL,

ComputeTermsSin, //sin(x)

ComputeTermsCos, //cos(x)

ComputeTermsExp, //exp(x)

ComputeTermsLn //ln(1 + x)

};

//МАССИВ ФУНКЦИЙ ДЛЯ СУММИРОВАНИЯ

SummationFunction SummationMethods[] = {

NULL,

DirectSummation, //ПРЯМОЕ СУММИРОВАНИЕ

ReverseSummation, //ОБРАТНОЕ СУММИРОВАНИЕ

PairwiseSummationRes //ПОПАРНОЕ СУММИРОВАНИЕ

};

//МАССИВ ФУНКЦИЙ ИЗ math.h

float (\*MathLibFunctions[])(float x) = {

NULL,

sinf, //sin(x)

cosf, //cos(x)

expf, //exp(x)

log1pf //ln(1 + x)

};

float MyResult = ComputeTaylorRow(x, count, ComputeTermsFunctions[FunctionType], SummationMethods[SummationMethod]);

float MathLibResult = MathLibFunctions[FunctionType](x);

printf("Result my method: %e\n", MyResult);

printf("Result of library <math.h>: %e\n", MathLibResult);

printf("Calculation error: %e\n", fabsf(MyResult - MathLibResult));

}