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»**

**Выполнила**:

студентка группы 3823Б1ПМ1-1

Акчурина А.Н.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2024

**Оглавление**

[Постановка задачи 3](#_Toc161676556)

[Метод решения 4](#_Toc161676557)

[Руководство пользователя 5](#_Toc161676558)

[Описание программной реализации 6](#_Toc161676559)

[Подтверждение корректности 8](#_Toc161676560)

[Результаты экспериментов 9](#_Toc161676561)

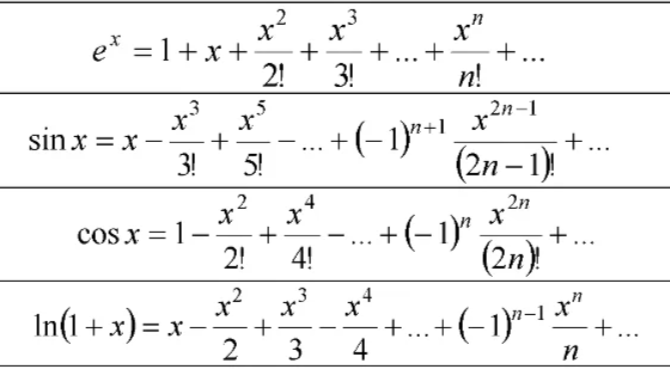
[Заключение 18](#_Toc161676562)

[Список литературы 19](#_Toc161676563)

[Приложение 20](#_Toc161676564)

# Постановка задачи

Требовалось написать на языке С программу, реализующую подсчет значений через ряд Маклорена для таких функций как синус, косинус, натуральный логарифм и экспонента.



Также необходимо выполнить расчет этих функций тремя разными видами суммирования: прямым, попарным и обратным .В конце выполнить сравнение результатов выполнения подсчетов суммы каждой функции для нахождения более корректного алгоритма.

# Метод решения

Для решения поставленной задачи, необходимо реализовать все 3 вида суммирования. Для выполнения всех 3-х видов суммирования была найдена формула расчета промежуточного члена ряда(pi). Таким образом для избежания переполнения при подсчете способом с применением факториалов, каждый последующий член последовательности считался через предыдущего домножением на промежуточный член:

* Pi(exponent) = x/i
* Pi(sin) = - x^2 / ( 2 \* i \* ( 2 \* i + 1))
* Pi(cos) = - x^2 / (( 2\* i \* (2 \* I – 1))
* Pi(ln) = - x \* i / ( i + 1)

где i – номер члена ряда, x – вводимый пользователем аргумент в радианах.

**Виды алгоритмов сложения**:

1. **Прямое суммирование:** Реализуется сложением слагаемых от начала к концу. Таким образом, происходит суммирование сначала больших, затем меньших членов ряда.(Последним членом суммы будет последний член ряда).
2. **Попарное суммирование:** Реализуется сложением не каждого члена ряда по отдельности, а парами. Таким образом, попарное суммирование складывает 1-й и 2-й, 3-й и 4-й, … , n-1-й и n-й члены.
3. **Обратное суммирование:** Реализуется сложением слагаемых от конца к началу. Таким образом происходит суммирование сначала больших, а затем меньших членов ряда. (Последним членом суммы будет 1-й член ряда).

В конце программа выводит все полученный значения на консоль для последующей проверки пользователем на корректность.

# Руководство пользователя

При запуске программы пользователю дается возможность ввести значение «х» современной арабской записью (цифрами) слева направо. Именно по этому аргументу впоследствии будут вычисляться значения четырех функций : sin, cos, exp, и ln. На вывод программы будут подаваться значения этих функций вычисленных тремя алгоритмами: прямым, попарным и обратным суммированием.

# Описание программной реализации

Программа начинает свою работу с функции main(). Она определяет диапазон членов ряда Маклорена (до 15), а также запрашивает у пользователя ввод значения “x”. После считывания ответа пользователя она реализует вызов последующих функций для выполнения разных алгоритмов суммирования. Затем выводит их значения на консоль для дальнейшей проверки на корректность.

**Функции, вызывающиеся в main:**

1. **float ln(float x, int N) –** эта функция выполняет прямое суммирование натурального логарифма. Изначально переменной res присваивается нулевое значение, а переменной term значение «х». После этого идет выполнение цикла, в котором реализуется счетчик(все значения запоминаются переменной res). Переменная term на каждом шаге меняет свой знак, соответственно и свое значение, домножаясь на промежуточную переменную. По окончанию выводится значение счетчика(res).
2. **float ln\_end(float x, float (\*sum)(float, float, float(\*)(float, int), int)) –** эта функция работает с указателем на попарное суммирование и обратное.
3. **float ln\_member(float x, int i) -** эта функция работает с указателем на попарное суммирование и обратное.
4. **float my\_sin(float x, int N) –** эта функция выполняет прямое суммирование натурального логарифма. Изначально переменной res присваивается нулевое значение, а переменной term значение «х». После этого идет выполнение цикла, в котором реализуется счетчик(все значения запоминаются переменной res). Если значение переменной term четное, то последующее значение будет вычитаться, в ином случае оно суммируется. Переменная term на каждом шаге меняет свое значение, домножаясь на промежуточную переменную. По окончанию выводится значение счетчика(res).
5. **float sin\_end(float x, float (\*sum)(float, float, float(\*)(float, int), int)) -** эта функция работает с указателем на попарное суммирование и обратное.
6. **float sin\_member(float x, int i) –** функция реализующая подсчет члена ряда.
7. **Float exponent(float x, int N)** – эта функция выполняет прямое суммирование для экспоненты. Изначально переменной res присваивается нулевое значение, а переменной term единица. После этого идет выполнение цикла, в котором реализуется счетчик(все значения запоминаются переменной res). Переменная term на каждом шаге меняет свое значение, домножаясь на промежуточную переменную. По окончанию выводится значение счетчика(res).
8. **float exp\_end(float x, float (\*sum)(float, float, float(\*)(float, int), int)) -** эта функция работает с указателем на попарное суммирование и обратное.
9. **float exp\_member(float x, int i)** – эта функция реализующая подсчет члена ряда.
10. **float my\_cos(float x, int N) -** эта функция выполняет прямое суммирование натурального логарифма. Изначально переменной res присваивается нулевое значение, а переменной term значение «х». После этого идет выполнение цикла, в котором реализуется счетчик(все значения запоминаются переменной res). Если значение переменной term четное, то последующее значение будет вычитаться, в ином случае оно суммируется. Переменная term на каждом шаге меняет свое значение, домножаясь на промежуточную переменную. По окончанию выводится значение счетчика(res).
11. **float cos\_end(float x, float (\*sum)(float, float, float(\*)(float, int), int)) -** эта функция работает с указателем на попарное суммирование и обратное.
12. **float cos\_member(float x, int i)** – эта функция реализующая подсчет члена ряда.
13. **float inverse\_sum(float a0, float x, float(\*funk)(float, int), int N) –** эта функция реализует алгоритм обратного суммирования.
14. **float par\_sum(float a0, float x, float(\*funk)(float, int), int N) –** эта функция реализует алгоритм попарного суммирования.

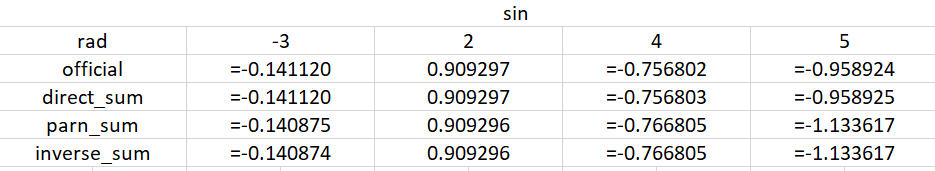
# Подтверждение корректности

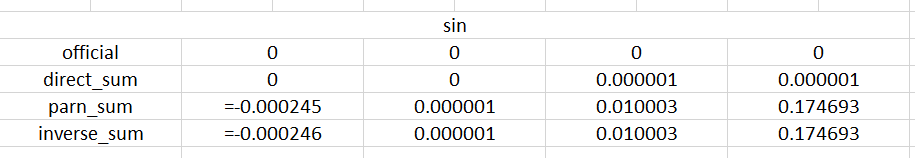
Для подтверждения корректности на консоль выводятся поочередно сначала значения прямого, обратного и попарного суммирования синуса, затем косинуса, экспоненты и натурального логарифма. Таким образом пользователь имеет возможность сравнить полученные значения при реализации этих алгоритмов с официальными данными для введенного аргумента.

# Результаты экспериментов

Эксперименты проводились следующим образом: сначала происходило суммирование различными способами (прямое, обратное и попарное), затем все это выводилось на консоль. На последнем шаге происходила проверка на корректность, которая заключалась в том, чтобы сравнить официальные данные синуса, косинуса, экспоненты и натурального логарифма с полученными в ходе эксперимента. Таким образом, для наибольшего удобства все данные выводились на консоль в виде столбца, благодаря чему более ярко были выражены несоответствия в значениях суммирования.

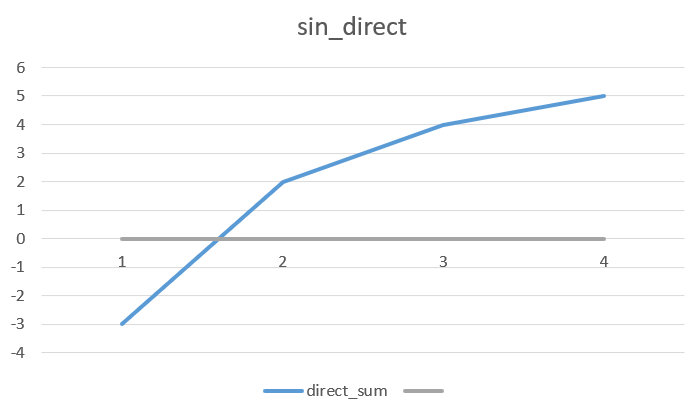
Для наибольшего удобства определения корректности результатов были сделаны таблицы значений и графики соответствий.



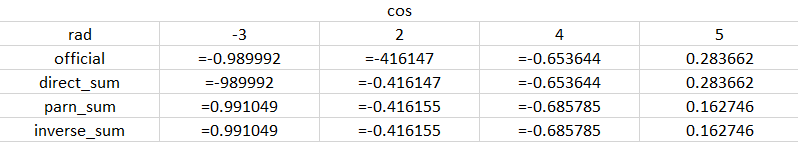


В данной таблице приведена разница в показаниях прямой, попарной и обратной сумм с официальными значениями (для наглядности).

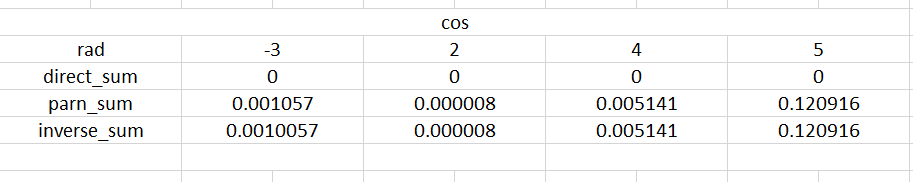
Из таблицы мы видим, что прямая сумма выводит значения, полностью совпадающие с официальными данными. Попарное и обратное же суммирование выводят на консоль значения, не совпадающие с официальными данными лишь в тысячных долях.



Сравнивая графики, можно заметить, что их отличия крайне незначительны и практически незаметны.

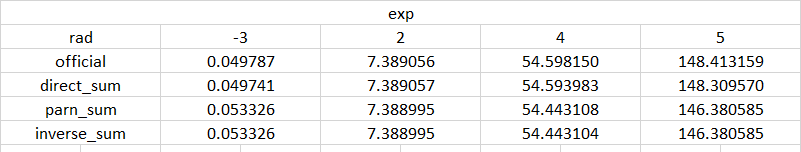


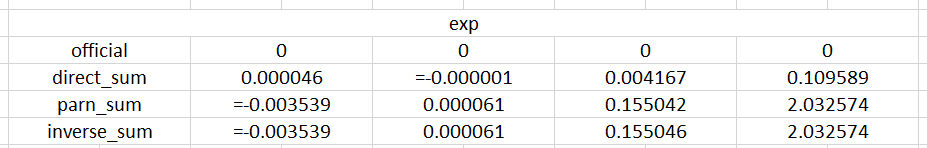
Из таблицы видно, что при прямом суммировании значение полностью совпадает с официальными данными этой функции. Попарное и обратное суммирования имеют значения отличающиеся на сотые доли.



В данной таблице приведена разница в показаниях прямой, попарной и обратной сумм с официальными значениями (для наглядности).

По графикам видно, что различия в значениях минимальны. Таким образом, их сложно определить.

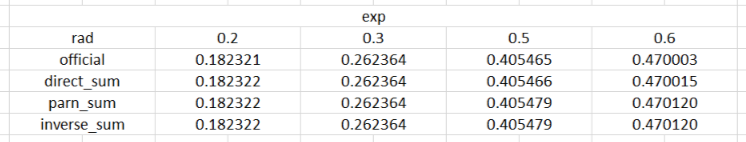




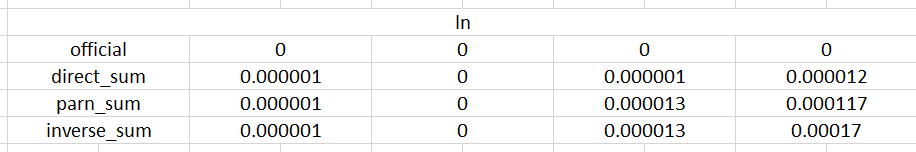
В данной таблице приведена разница в показаниях прямой, попарной и обратной сумм с официальными значениями (для наглядности).

Из таблицы видно, что прямое суммирование выводит идентичные официальному значения. Попарное и обратное суммирования имеют одиноково отличные в сотых долях значения.

По графикам видно, что различия в значениях минимальны. Таким образом, их сложно определить.



Из таблицы мы видим, что о более малых значениях аргумента в попарном, прямом и обратом суммировании. Начиная с 0.5, появляются незначительные погрешности в десятитысячных долях.



В данной таблице приведена разница в показаниях прямой, попарной и обратной сумм с официальными значениями (для наглядности).

По графикам видно, что различия в значениях минимальны. Таким образом, их сложно определить.

# Заключение

По результатам проведенных экспериментов были сделаны следующие выводы:

* Все способы недостаточно точны из-за самого принципа многочлена Маклорена: чем дальше от , тем меньше точность. Из-за этого при достаточно больших некоторые способы оказывались формально точнее.
* Для тригонометрических функций наиболее корректным алгоритмом суммирования является прямая сумма, изменение в результатах в отличии от обратного и попарного суммирования практически отсутствуют. Для натурального логарифма обратное и попарное суммирование также были наименее точными. При расчете экспоненты более точными были результаты прямого суммирования. Обратное и попарное суммирования имели погрешности в тысячных и сотых долях, поэтому неточность измерения была наиболее наглядной именно при просмотре таблиц. Графики же практически не имели различий, или же некорректность вычислений была еле заметна.
* Таким образом можно сделать вывод, что с большей вероятностью реализация прямой суммы будет более корректным способом вычисления.

# Список литературы

1. Демидович Б. П. “Сборник задач и упражнений по математическому анализу. Учебное пособие для вузов”/ Б. П. Демидович. – М.: ООО «Издательство Астрель»: ООО «Издательство АСТ», 2005. (дата обращения – 20.02.2024).
2. *Ильин В. А., Садовничий В. А., Сендов Б. Х.* Математический анализ, ч. 1, изд. 3, ред. А. Н. Тихонов. М.: Проспект, 2004.

# Приложение

float exponent(float x, int N) {//прямая сумма

float res = 0, term = 1;

for (int i = 1; i < N; i++) {

res += term;

term \*= x / (float)i;

}

return res;

}

float exp\_member(float x, int i) {//нахождение значения отдельного члена экспоненты

return x / i;

}

float my\_sin(float x, int N) {//прямая сумма

float res = 0, term = x;

for (int i = 1; i < N; i++) {

if (i % 2 == 0) res -= term;

else res += term;

term \*= x \* x / (float)(2 \* i \* (2 \* i + 1));

}

return res;

}

float sin\_member(float x, int i) {//нахождение отдельного члена синуса

return -x \* x / (2 \* i \* (2 \* i + 1));

}

float my\_cos(float x, int N) {//прямая сумма

float res = 0, term = 1;

for (int i = 1; i < N; i++) {

if (i % 2 == 0) res -= term;

else res += term;

term \*= x \* x / (float)((2 \* i - 1) \* 2 \* i);

}

return res;

}

float cos\_member(float x, int i) {//нахождение отдельного члена косинуса

return -x \* x / (2 \* i \* (2 \* i - 1));

}

float ln(float x, int N) {//прямая сумма

float res = 0, term = x;

for (int i = 1; i < N; ++i) {

res += term;

if (i == 0) term \*= (-1) \* x / (float)(i + 1);

else term \*= (-1) \* x \* (float)(i) / (float)(i + 1);

}

return res;

}

float ln\_member(float x, int i) {//нахождение отдельного члена натурального логарифма

return -x \* i / (i + 1);

}

float inverse\_sum(float a0, float x, float(\*funk)(float, int), int N)//обратная сумма

{

float res, an, b;

an = a0;

res = 0;

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

{

b = funk(x, i);

if (b == 0)

{

N = i - 1;

break;

}

an \*= b;

}

for (int i = N; i >= 1; i--)

{

res += an;

an /= funk(x, i);

}

return res + a0;

}

float par\_sum(float a0, float x, float(\*funk)(float, int), int N)//попарная сумма

{

float res, a1, a2;

res = a1 = a0;

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

{

a1 \*= funk(x, i);

a2 = a1;

if (i + 1 <= N)

{

a1 \*= funk(x, ++i);

a2 += a1;

}

res += a2;

}

return res;

}