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 функций»**

**Выполнил**:

студент группы 3821Б1ПМ2

Деревянкин Кирилл Евгеньевич

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 7](#_Toc26962566)

[Результаты экспериментов 8](#_Toc26962567)

[Заключение 9](#_Toc26962568)

[Приложение 10](#_Toc26962569)

# Постановка задачи

Требовалось реализовать вычисление значения функции sin(x), cos(x), e^x, ln(1+x) с помощью ряда Маклорена на языке программирования C для типа данных double. Нужно реализовать методы прямого, попарного и обратного суммирования, описать реализацию и алгоритмы работы программы. Необходимо подтвердить корректность реализации вычисления данных функций. Провести эксперименты по замеру точности различных методов суммирования.

# Метод решения

Для подсчёта данных функций я использовал ряды Маклорена:

Синус



Косинус



Экспонента



Логарифм натуральный (от -1 до 1)



# Руководство пользователя

Чтобы найти значение, в начале программы пользователю нужно выбрать функцию. Для этого нужно выбрать цифру от 1 до 4, где 1 – sin(x), 2 – cos(x), 3 – exp(x), 4 – ln(1+x). Затем пользователь пишет с клавиатуры значение x. Программа выводит на экран значение x, прямую, обратную и попарную суммы, а также их абсолютные и относительные ошибки.

# Описание программной реализации

В ходе своей работы я использовал библиотеки <stdio.h>, <locale.h>, <math.h>. Чтобы пользователь мог выбрать функцию, я использовал оператор switch. Для всех суммирований был реализован цикл for с числом 20. Также использовал функцию double factorial(double number) для нахождения факториала числа.

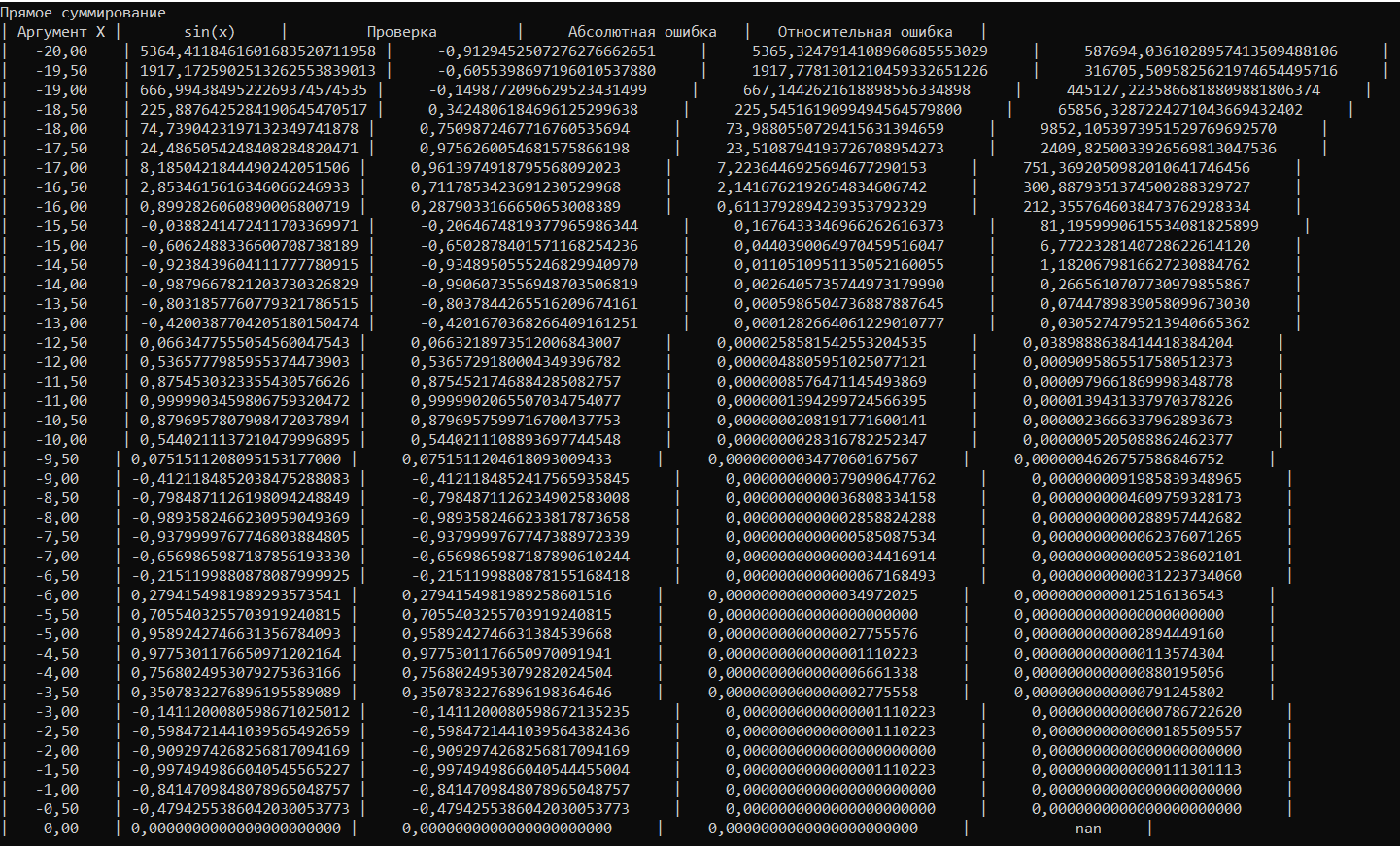
# Подтверждение корректности

Для подтверждения корректности программы я использовал стандартные математические функции (sin(x), cos(x), exp(x), log(1+x)) из библиотеки <math.h>.

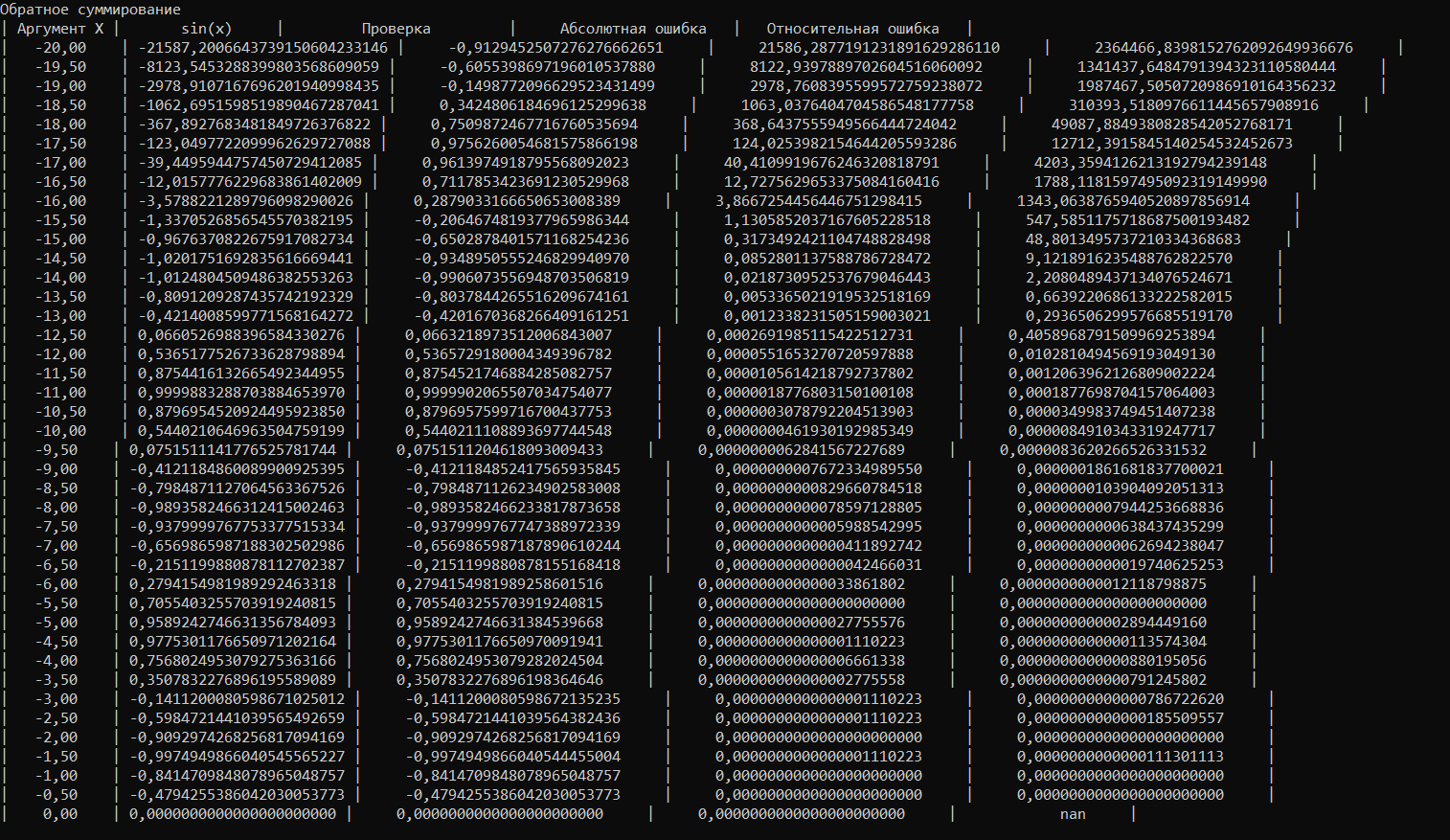
# Результаты экспериментов

Для sin(x), cos(x), e^x я использовал интервал от -20 до 0 с шагом 0,5, а для ln(1+x) – от -1 до 0,95 с шагом 0,05.

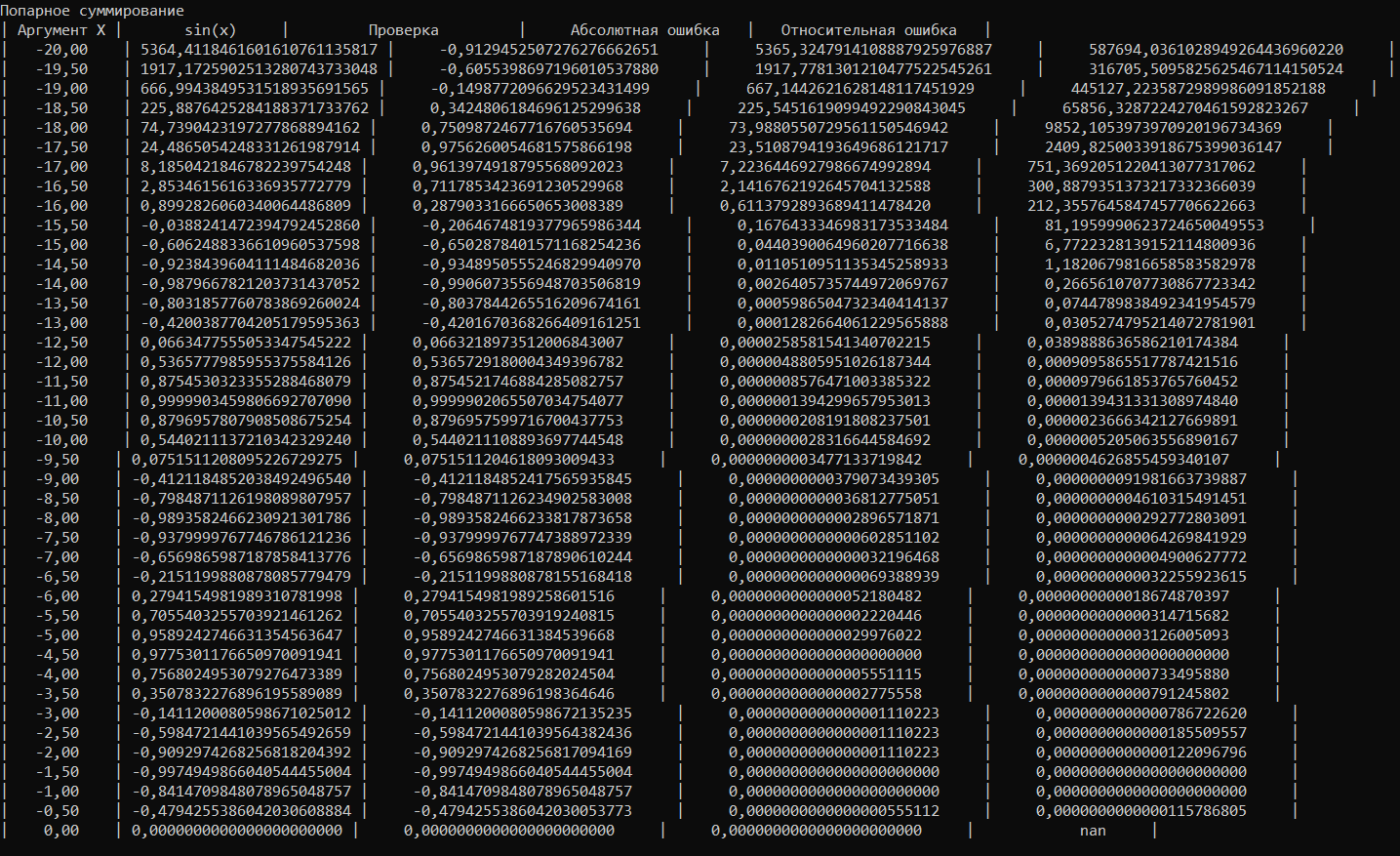
Прямое суммирование для sin(x).



Обратное суммирование для sin(x).

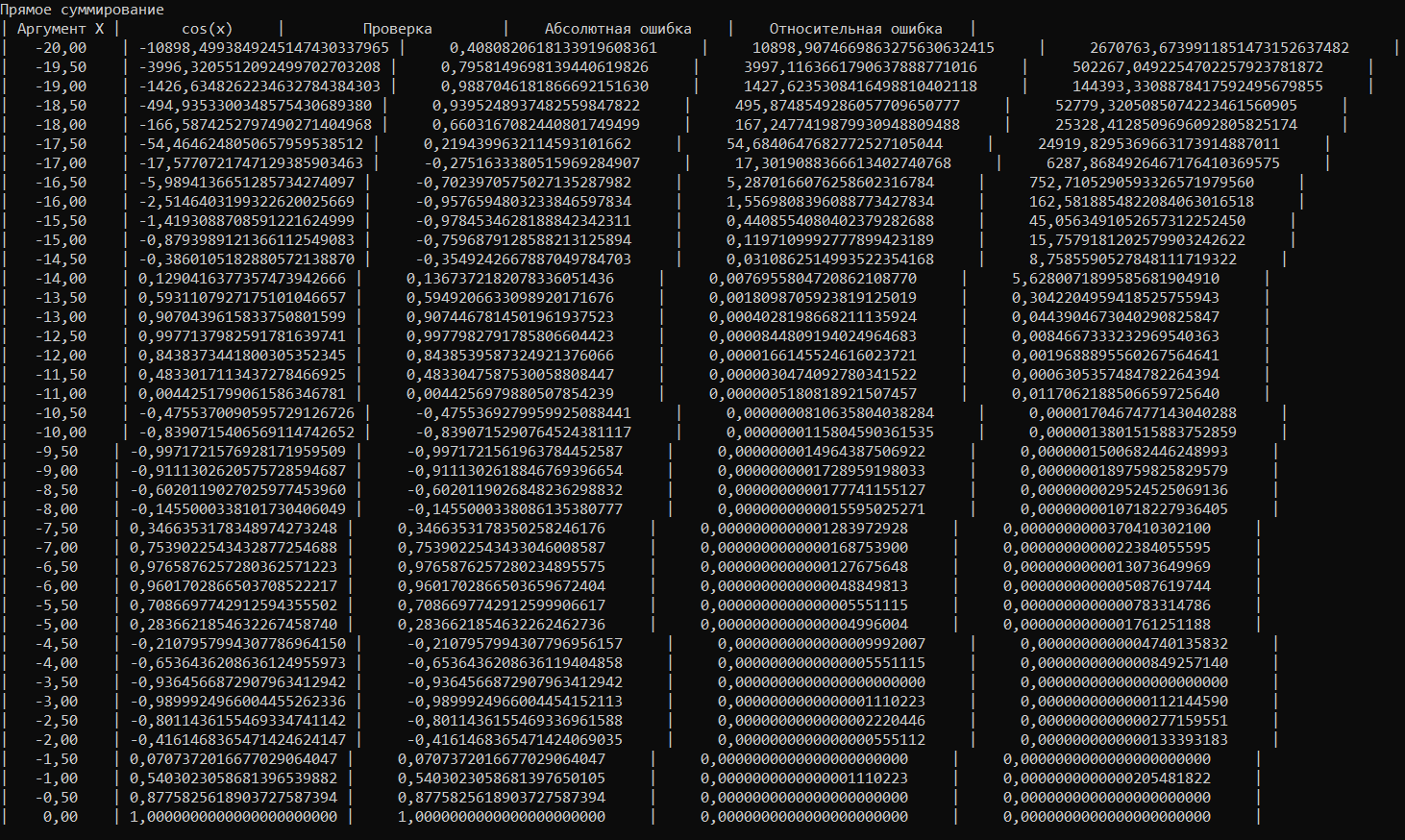


Попарное суммирование для sin(x).

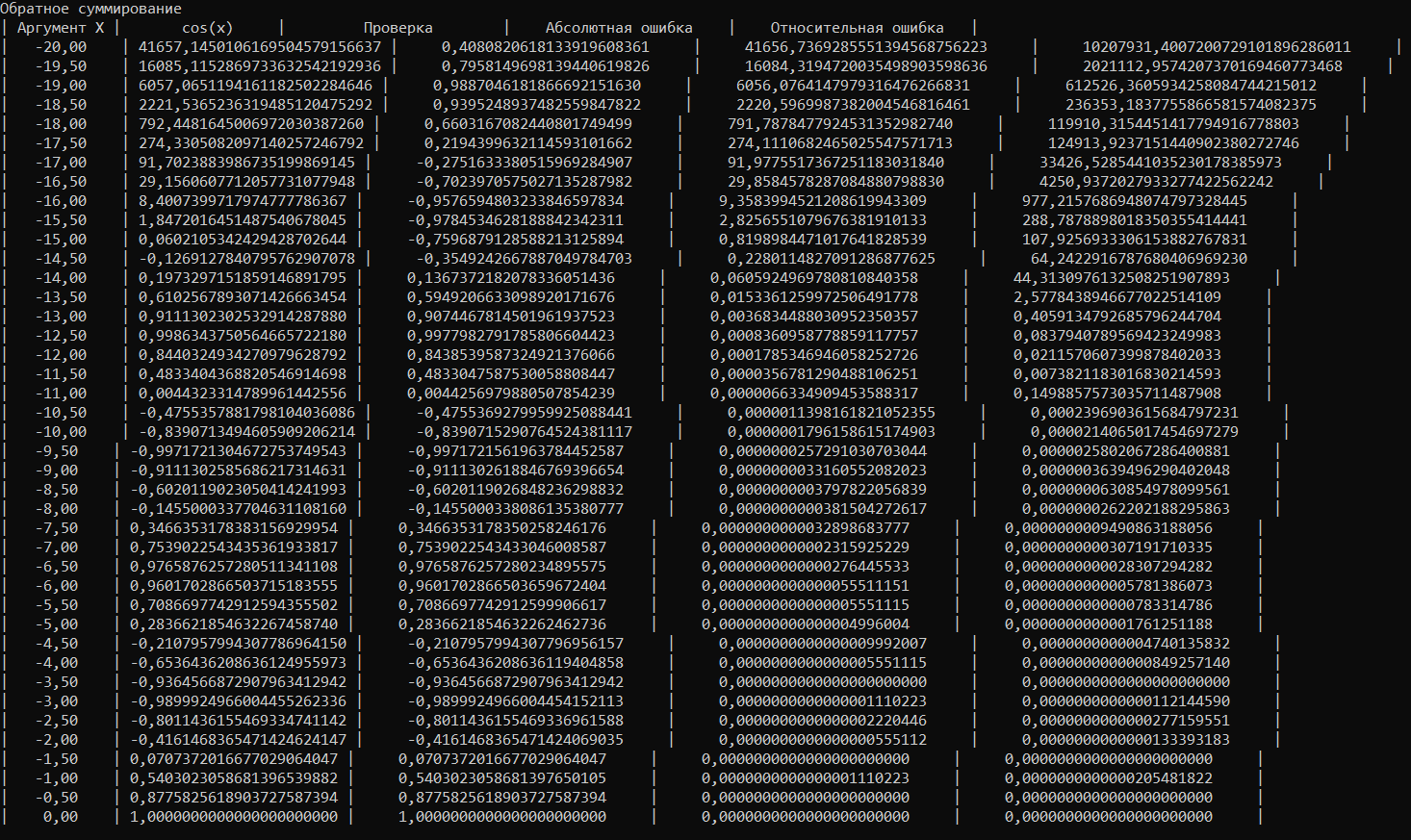


Вывод: для sin(x) лучше всех себя показали прямое и попарное суммирования, их результаты были точнее, чем обратное суммирование. Более точные результаты показало обратное суммирование, но при больших x было большое расхождение с действительными результатами.

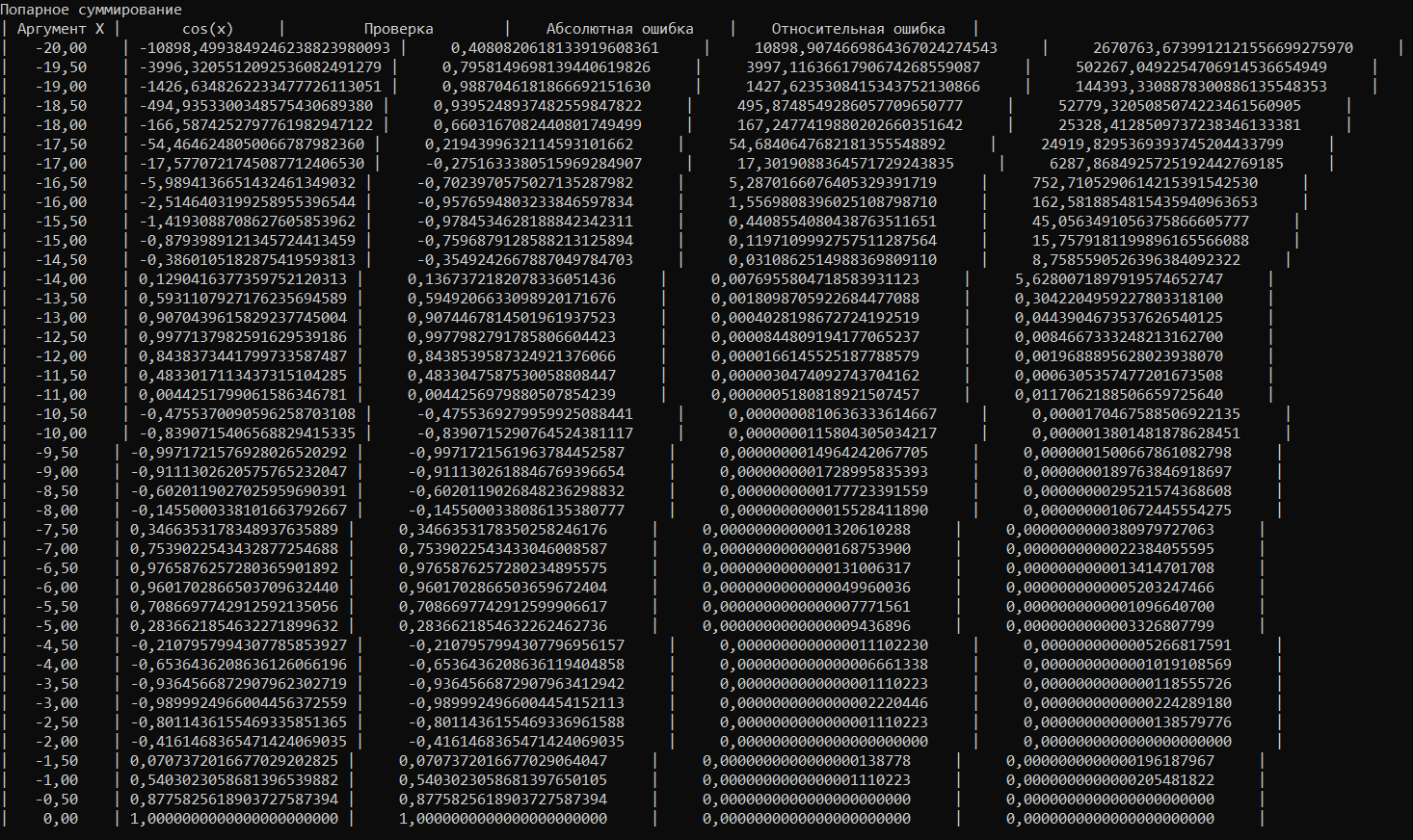
Прямое суммирование для cos(x).



Обратное суммирование для cos(x).

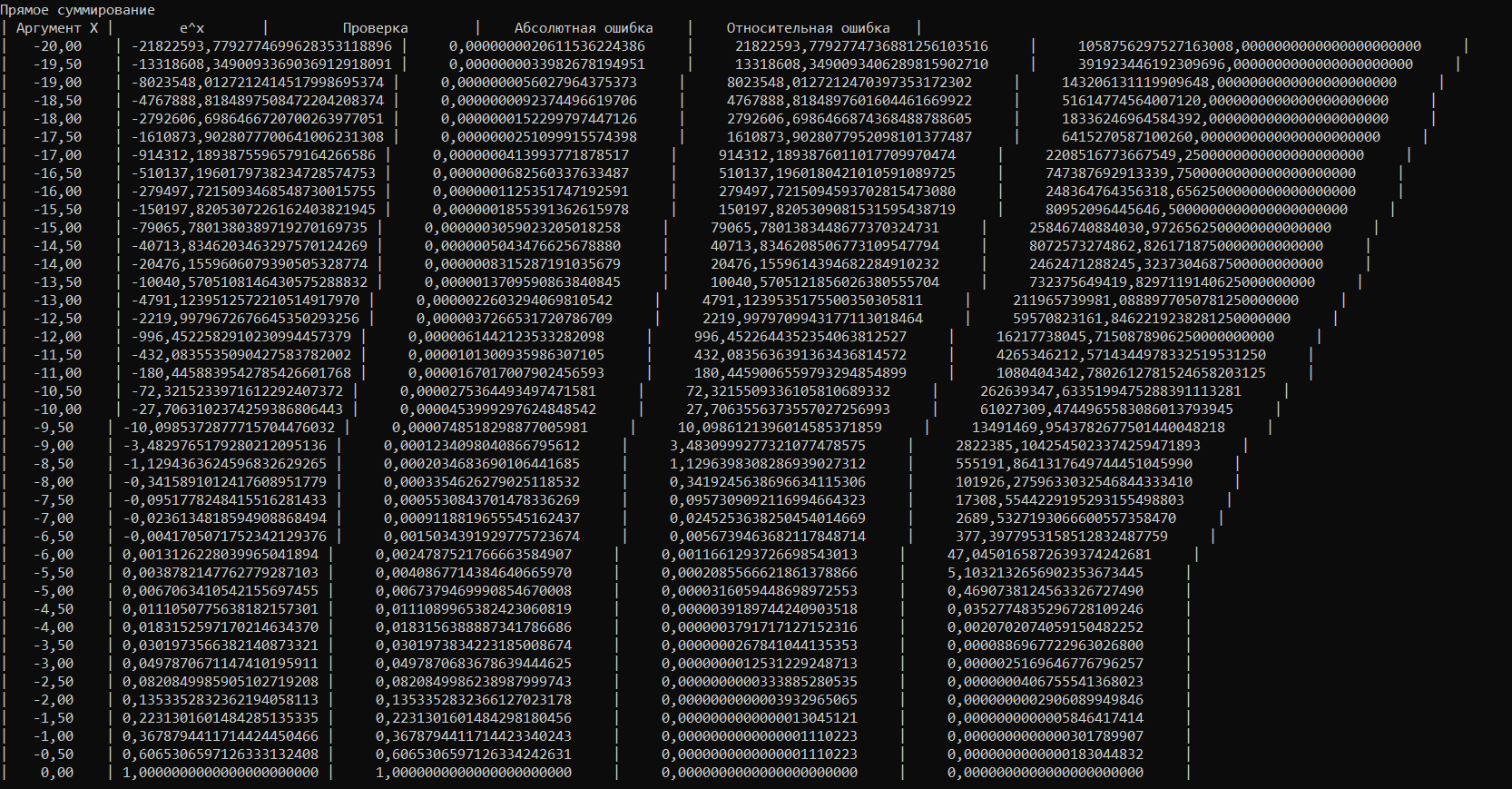


Попарное суммирование для cos(x).

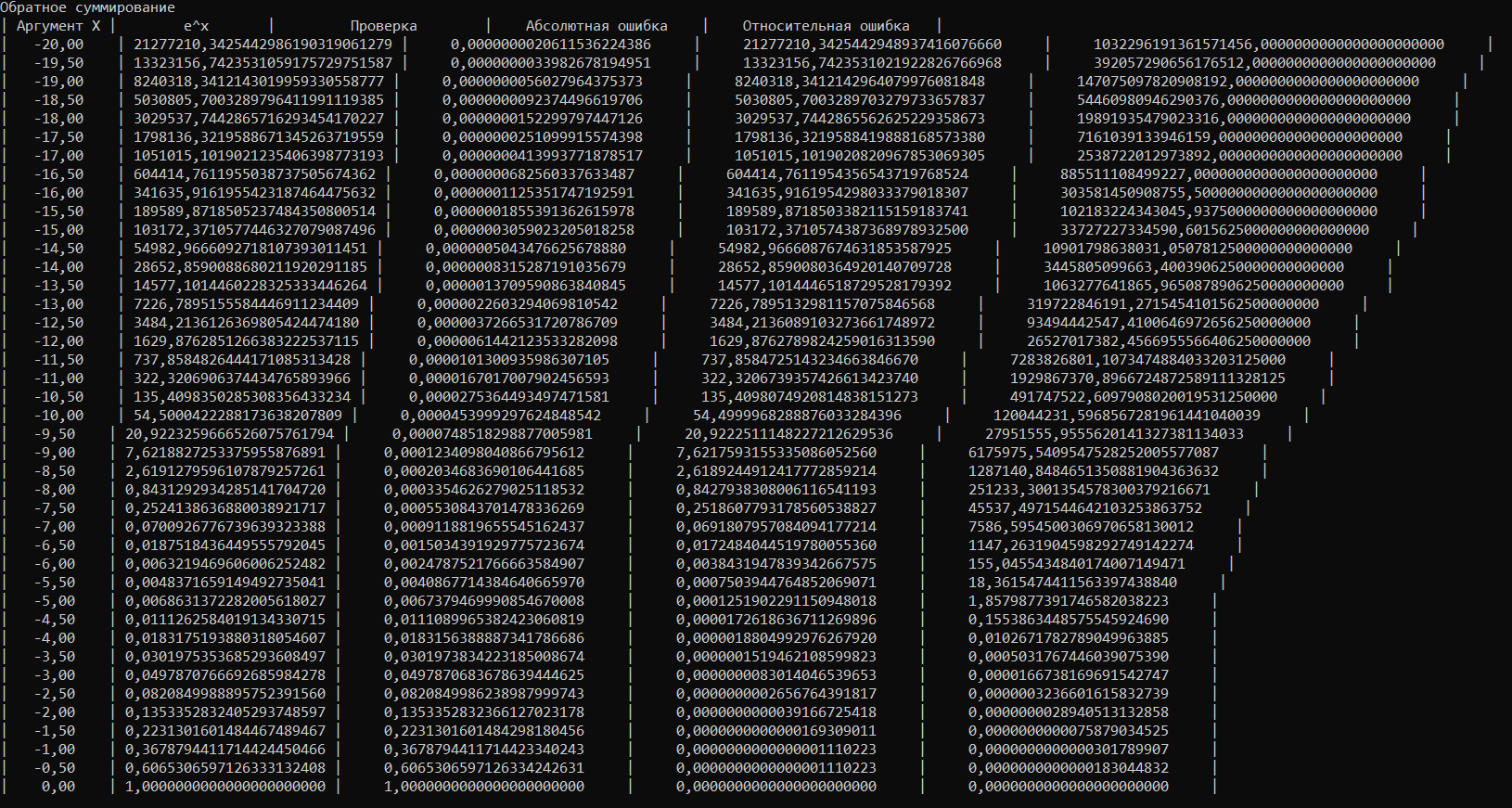


Вывод: для cos(x) лучше всего были прямое попарное суммирования. Обратное суммирование показывает небольшую ошибку при маленьких значениях x, но при больших разница увеличивается.

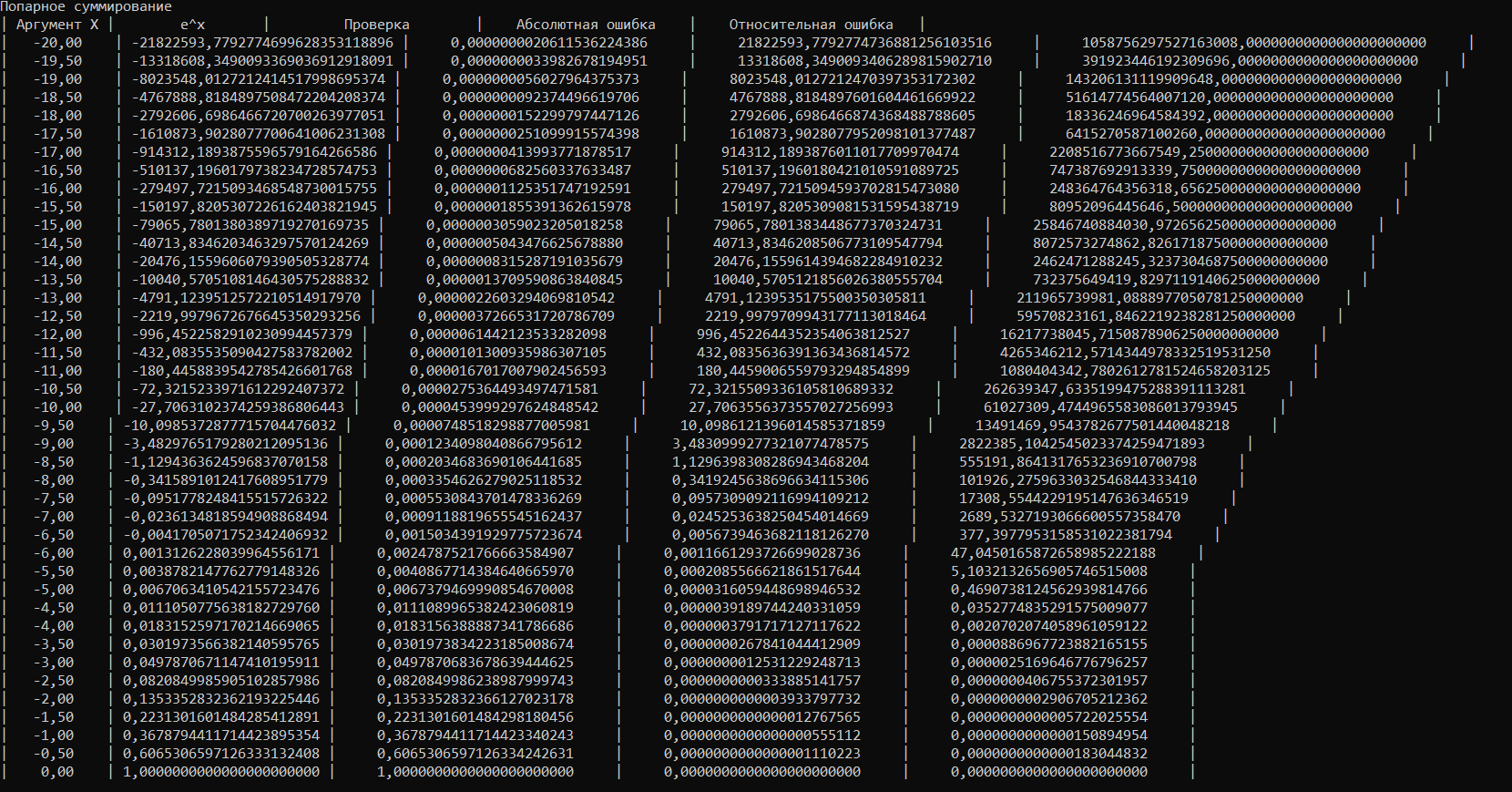
Прямое суммирование для e^x.



Обратное суммирование для e^x.

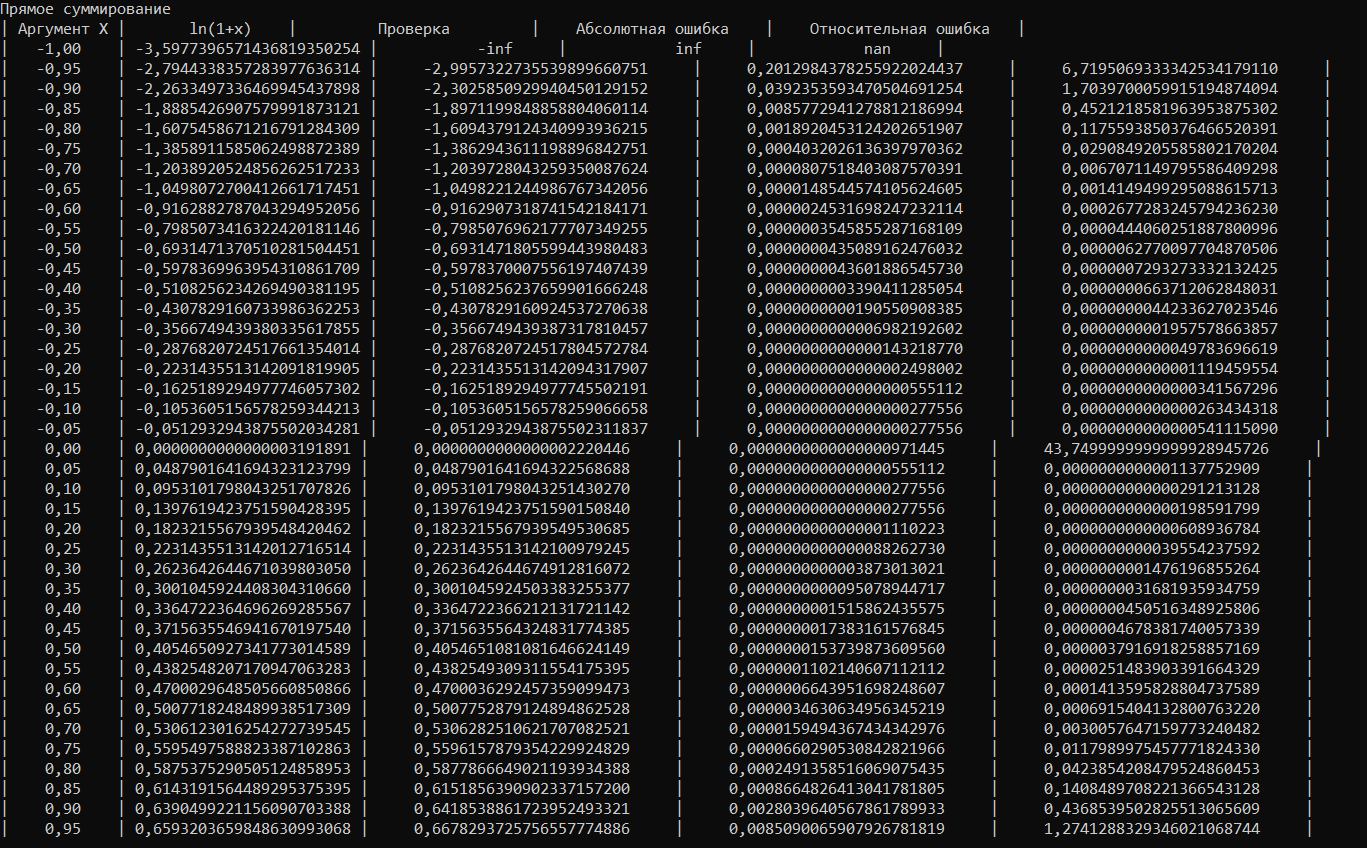


Попарное суммирование для e^x.

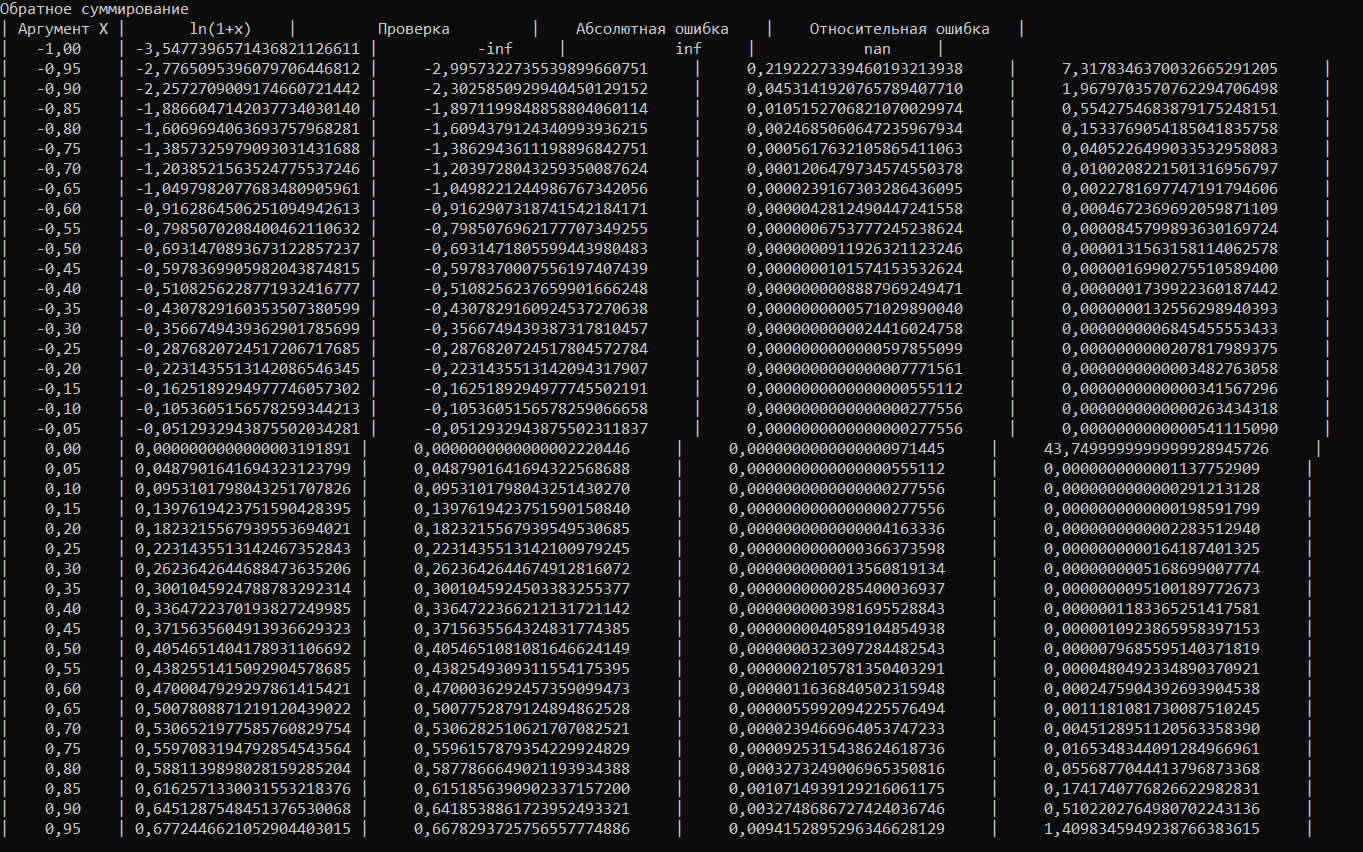


Вывод: для e^x лучше всех были прямое и попарное суммирования. Обратное суммирование показывает практически точные значения при маленьких значениях x, но при больших значениях разница увеличивается.

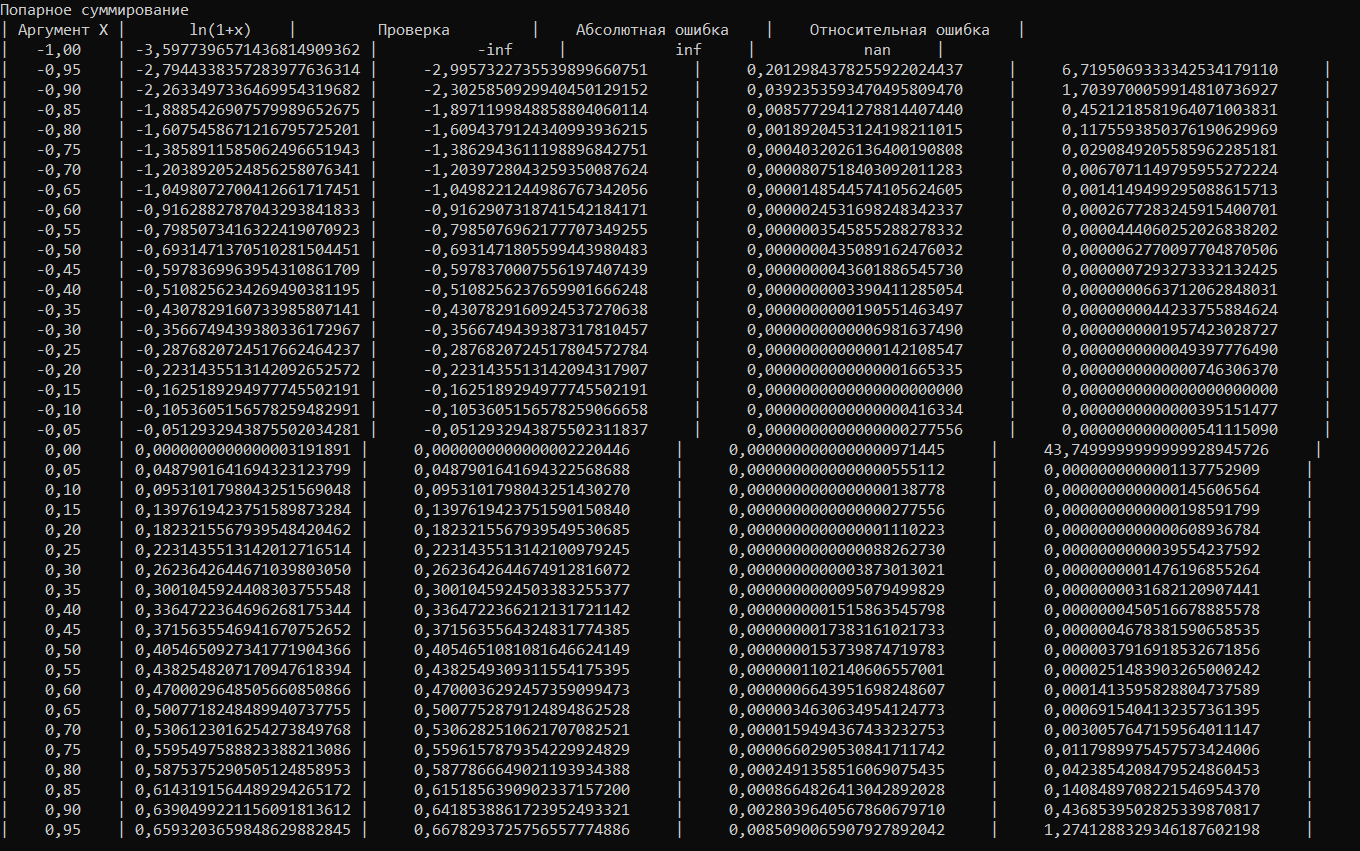
Прямое суммирование для ln(1+x).



Обратное суммирование для ln(1+x).



Попарное суммирование для ln(1+x).



Вывод: для ln(1+x) прямое и попарное суммирования были лучше. У обратного суммирования ближе к 1 и -1 увеличивается ошибка.

# Заключение

Я реализовал на языке С четыре функции: синуса, косинуса, экспоненты, натурального логарифма. Описал их алгоритмы работы, проверил корректность и вычислил погрешность. Результаты эксперимента показали, что лучше всех были прямое и попарное суммирования.

# Приложение

