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**

**«Подсчёт математических функций:sin(x),cos(x)ln(1+x),exp(x)»**

**Выполнил**:

студент группы 3821Б1ПМ2

Сулаймонов Н.Д.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2021

**Содержание**

**1)Постановка задачи …...................................................................................4**

**2)Метод решения........................................................................................5**

**3)Руководство пользователя..................................................................6**

**4)Описание программной реализации...............................................8**

**5)Подтверждение корректности..........................................................10**

**6)Результаты экспериментов...............................................................11**

**7)Заключение..........................................................................................14**

**8)Приложение.........................................................................................15**

￼

￼

￼

￼

￼

￼

￼

￼

# Постановка задачи

В этой лабораторной работе главная задача является в реализации вычислений 4 математических функций: cos(x), sin(x), exp(x), ln(1+x). (При помощи рядов Маклорена).Необходимо реализовать это при помощи метода прямого, попарного и обратного суммирования. Так же описать метод решения и описание своей программы. Подтвердить корректность своей программы и составить эксперимент по правильности написанной программы. Сделать свое заключение по своей программе.

# Метод решения

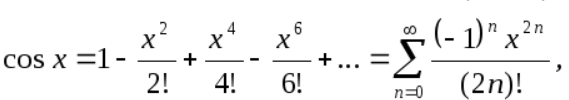
Метод решения программы основан на ряде Тейлора. **Ряд Тейлора** - это разложение функции в бесконечную сумму степенных функций. Попарное, обратное и прямое суммирование помогает в реализации программы.

**Попарное суммирование** - это метод суммирования последовательности чисел конечной точности с плавающей запятой, который существенно снижает накопленную ошибку округления по сравнению с наивным накоплением суммы по порядку.

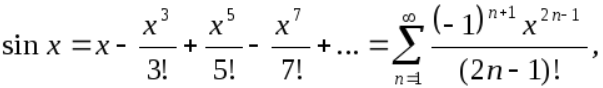
**Обратное суммирование -** метод вычислительной математики, позволяющий вычесть одно число из другого, используя только операцию сложения.

**Прямое суммирование -** каждое из слагаемых складывается в порядке уменьшения их номера порядка.

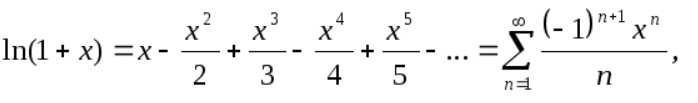
Разложение по Тейлору для математической функции cos(x):



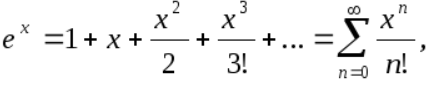
Разложение по Тейлору для математической функции sin(x):



Разложение по Тейлору для математической функции ln(x+1):



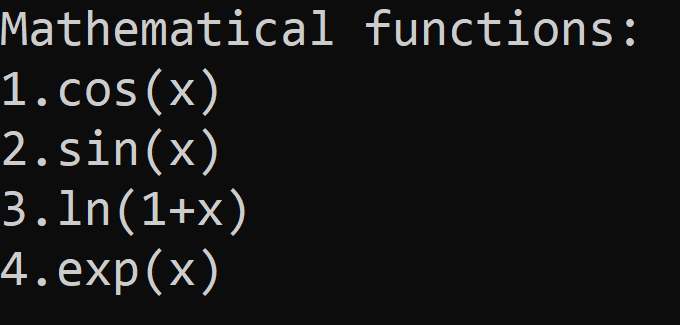
Разложение по Тейлору для математической функции exp(x):



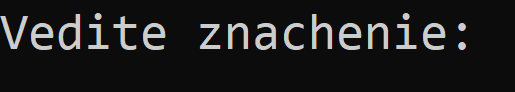
# Руководство пользователя

Руководство для пользователя для использования подсчета математических функций очень просто в использовании.

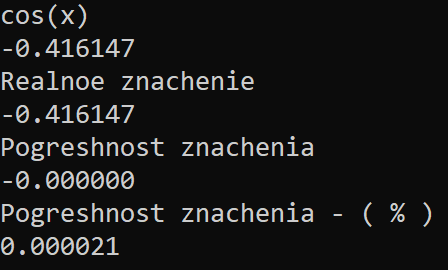
При компиляции программы выводится окно с 4 математическими функциями. Для подсчета нужной математической функции надо вести ее номер.



После этого выводится текст с надписью ”Вести значение”. Выбираете нужное вам значение.



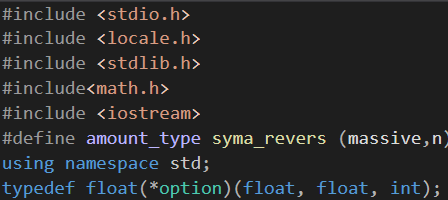
В результате этого выводится значение выбранной вами функции, ее точное значение из библиотеки math.h, погрешность значения и относительная погрешность в процентах.



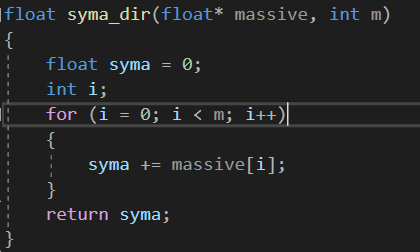
Это и есть руководство пользователя для моей программы.

# Описание программной реализации

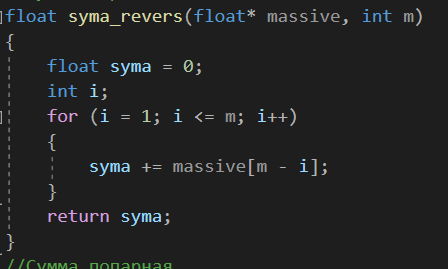
Ведение важных библиотек для реализации программы.



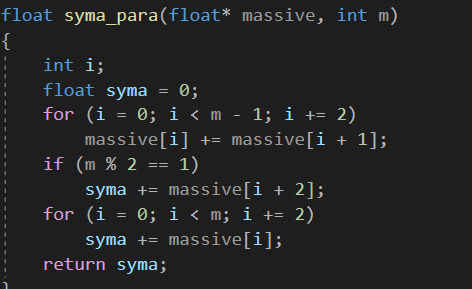
Функция прямого суммирования, где и производится суммирование элементов.



Функция обратного суммирования, где и производится суммирование элементов.



Функция попарного суммирования, где и производится суммирование элементов.

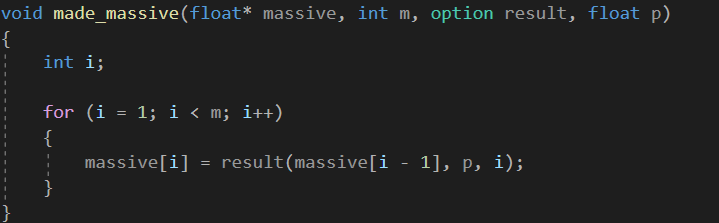


Функции возвращающие численные значения ряда для каждой математической функции.

1. **float cos\_x(float str, float n, int i)**
2. **float sin\_x(float str, float n, int i)**
3. **float exp\_x(float str, float n, int i)**
4. **float ln\_x(float str, float n, int i)**

**void main()** - Функция отвечающая за меню программы(Вывод и ввод значений)

Функция создающая массив элементов ряда Тейлора.



# Подтверждение корректности

Для подтверждения корректности моей программы я использовал погрешность значений (абсолютная погрешность) математической функции и ее относительная погрешность в процентном соотношении.

Относительная погрешность - погрешность значения(абсолютная погрешность) делится на значение математической функции и все это умноженное на 100.

Погрешность значения(Абсолютная погрешность) - разница значений между библиотечным и вычисленным в программе.

# Результаты экспериментов

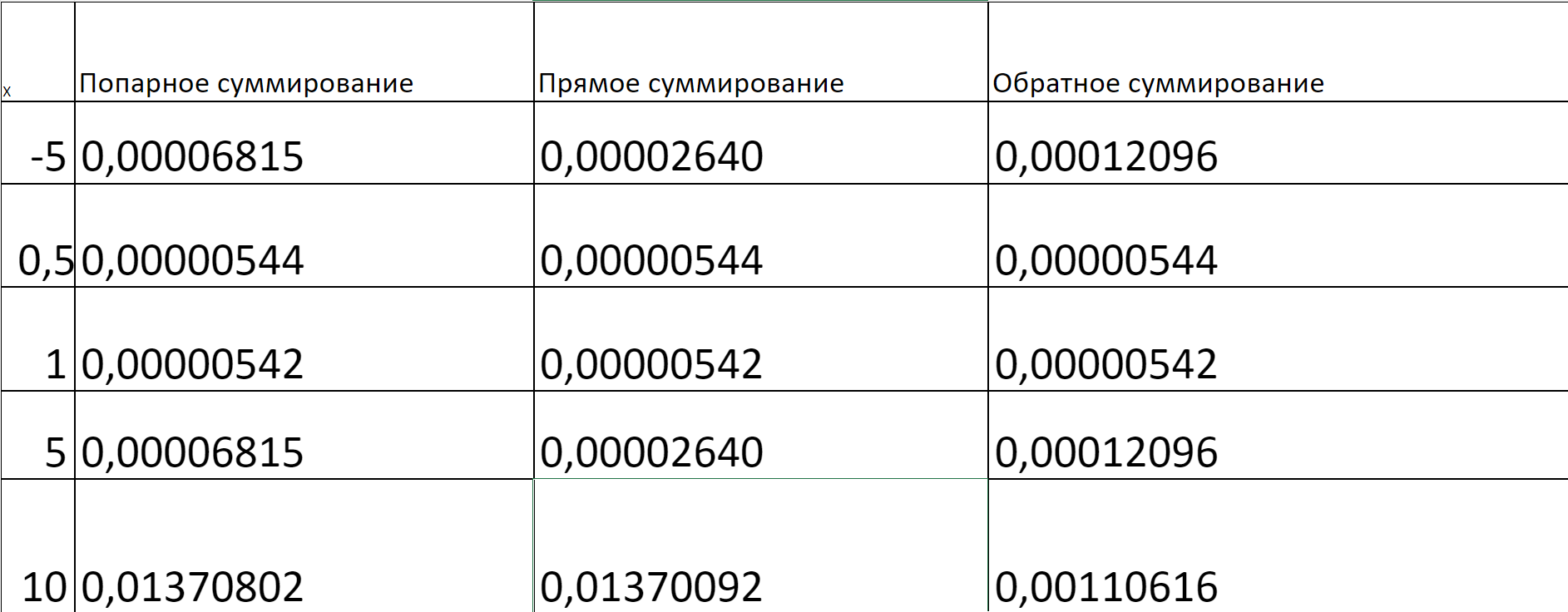
**1 столбец** - Веденное значение математической функции

**2 столбец-** Относительная погрешность попарного суммирования в процентах

**3 столбец-** Относительная погрешность прямого суммирования в процентах

**4 столбец-** Относительная погрешность обратного суммирования в процентах

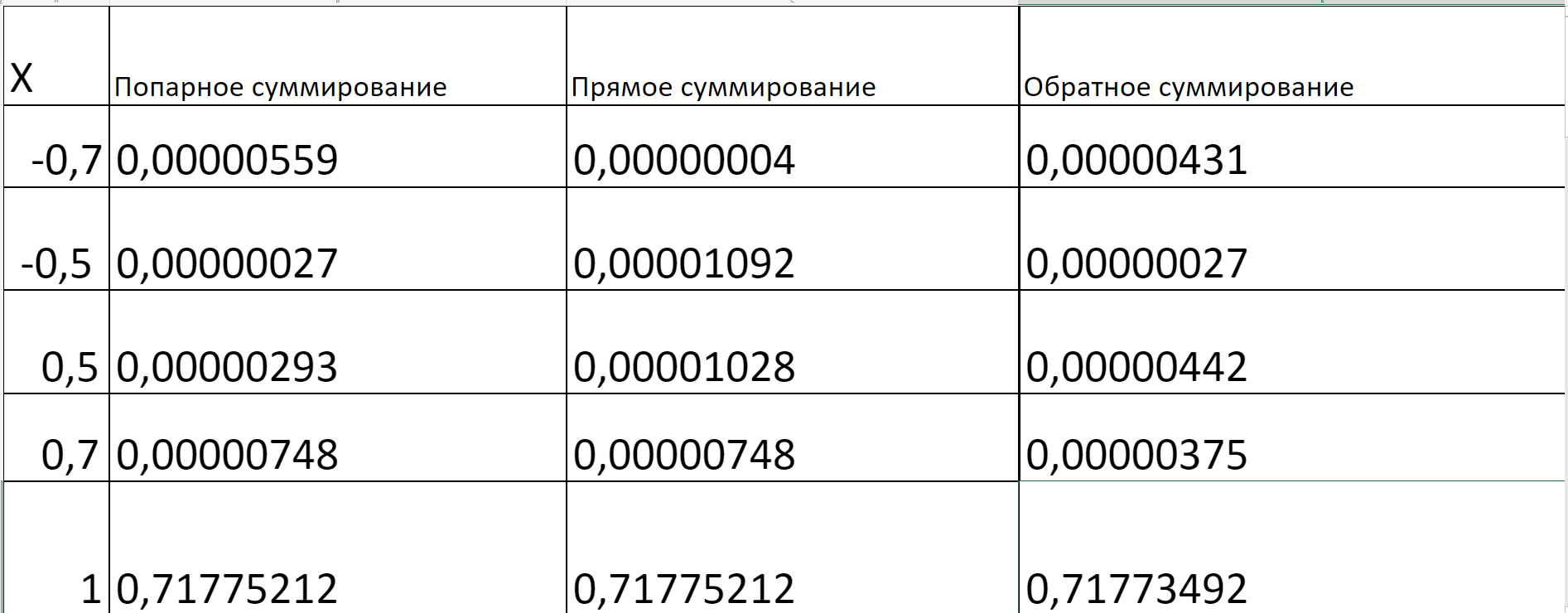
**1)COS(X)**



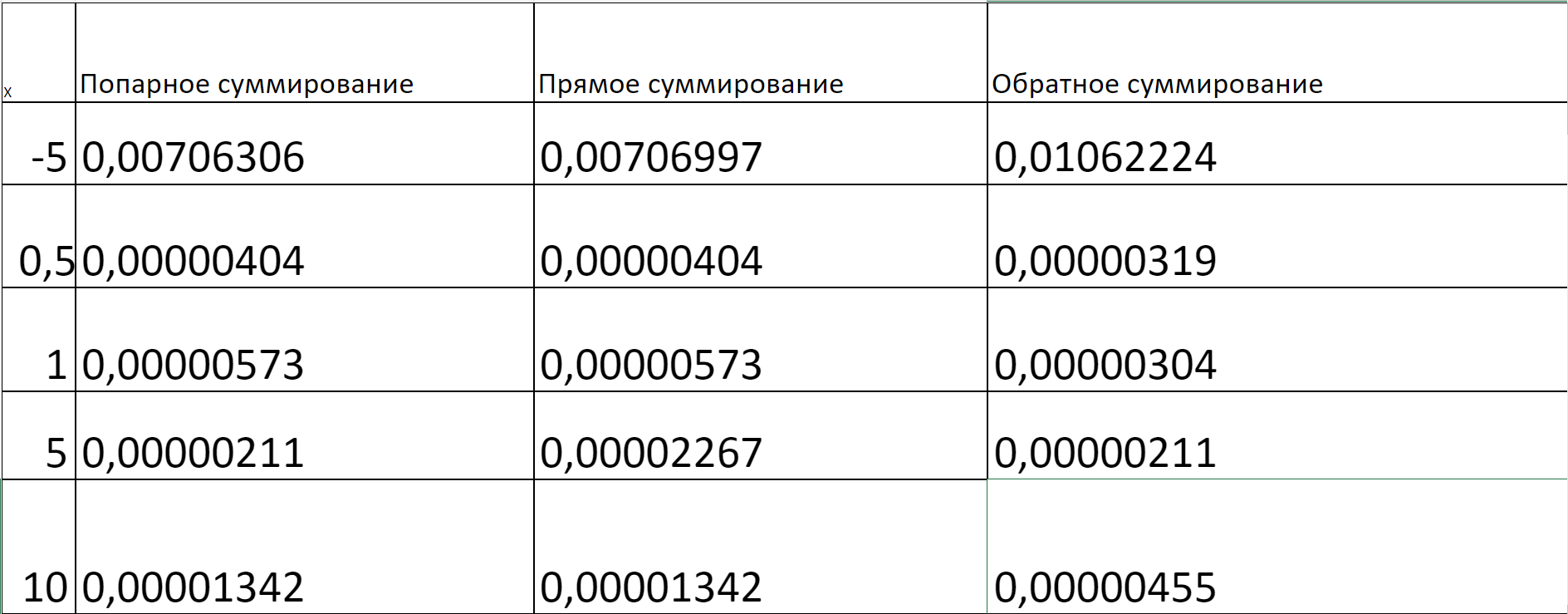
**2)SIN(X)**



**3)LN(1+X)**



**4)EXP(X)**



В результате экспериментов можно скачать что: для математических функций cos(x) , sin(x) лучше всего работает попарное суммирование(при небольших веденных значениях более лучше работает обратное суммирование). Для математической функции ln(1+x) лучше работает попарное суммирование. Для математической функции exp(x) лучше работает обратное суммирование.

# Заключение

Я выполнил данную лабораторную работу. Реализовал все 4 вычисления математических функций : sin(x) ,cos(x), ln(1+x),exp(x) . Также я показал метод решения и описал программную реализацию каждого вычисления моей математической функции. Я подтвердил корректность моего вычисления и провел ряд экспериментов, показывающих наиболее правильный и точный способ вычисления.

# Приложение

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include<math.h>

#include <iostream>

#define amount\_type syma\_revers (massive,n);

using namespace std;

typedef float(\*option)(float, float, int);

//Сумма рядов

float syma\_dir(float\* massive, int m)

{

float syma = 0;

int i;

for (i = 0; i < m; i++)

{

syma += massive[i];

}

return syma;

}

float syma\_revers(float\* massive, int m)

{

float syma = 0;

int i;

for (i = 1; i <= m; i++)

{

syma += massive[m - i];

}

return syma;

}

//Сумма пар

float syma\_para(float\* massive, int m)

{

int i;

float syma = 0;

for (i = 0; i < m - 1; i += 2)

massive[i] += massive[i + 1];

if (m % 2 == 1)

syma += massive[i + 2];

for (i = 0; i < m; i += 2)

syma += massive[i];

return syma;

}

//sin(x)

float sin\_x(float str, float n, int i)

{

i = 2 \* i + 1;

return (str \* (-1.f) \* n \* n) / (i \* (i - 1.f));

}

//cos(x)

float cos\_x(float str, float n, int i)

{

i = 2 \* i;

return (str \* (-1.f) \* n \* n) / (i \* (i - 1.f));

}

//exp(x)

float exp\_x(float pred, float n, int i)

{

return (pred \* n) / ((float)i);

}

//ln(x+1)

float ln\_x(float str, float n, int i)

{

return (str \* (-1.f) \* n \* i) / (i + 1.f);

}

void made\_massive(float\* massive, int m, option result, float p)

{

int i;

for (i = 1; i < m; i++)

{

massive[i] = result(massive[i - 1], p, i);

}

}