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

Коляда В.С.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2025

# Постановка задачи

В данной лабораторной работе необходимо реализовать вычисление функций , и с использованием разложения в ряд Тейлора. Кроме того, требуется провести сравнение результатов, рассчитанных разными методами суммирования, с значениями, полученными с помощью стандартных функций из библиотеки math.h.

# Метод решения

В лабораторной работе рассматриваются три метода суммирования членов ряда: прямое, обратное и попарное. Для вычисления членов ряда применяются рекуррентные формулы, что позволяет избежать переполнения из-за больших факториалов.

Формулы для вычисления функций:

: ,

: ,

: ,

: ,

**Прямое суммирование**

Этот метод предполагает последовательное сложение элементов ряда в порядке их вычисления, начиная с первого. Такой способ удобен для понимания и реализации, однако при большом количестве итераций возможны ошибки округления, особенно если члены ряда существенно отличаются по модулю. Это может привести к снижению точности вычислений.

Функция принимает два аргумента:

* **x** — значение аргумента функции,
* **n** — количество членов ряда, которые необходимо суммировать.

Перед вычислениями проводится проверка входных данных: для тригонометрических функций x предварительно приводится к диапазону [0,2π] с использованием остатка от деления на 2π. Это предотвращает возможные ошибки при работе с большими значениями аргумента.

Члены ряда рассчитываются рекуррентно, что позволяет избежать необходимости явного вычисления факториала, упрощая вычисления. В цикле каждый новый член прибавляется к общей сумме, а затем передается на следующую итерацию.

**Обратное суммирование**

В этом методе сумма членов ряда вычисляется, начиная с последнего элемента и двигаясь к первому. Такой подход позволяет снизить ошибки округления, возникающие при сложении чисел, значительно отличающихся по величине.

В отличие от прямого суммирования, здесь сначала вычисляются все члены ряда, но их суммирование выполняется в обратном порядке. Это особенно полезно, когда последние элементы ряда имеют малый порядок величины, так как уменьшает потерю значащих цифр при сложении.

**Попарное суммирование**

Данный метод заключается в сложении элементов ряда не последовательно, а группами по два. Это помогает уменьшить накопление ошибок округления, так как разница в порядке величины слагаемых внутри одной пары обычно меньше, чем при последовательном сложении всех членов ряда.

В процессе вычисления элементы ряда группируются, и сначала суммируются попарно. Если количество членов ряда нечётное, то последний элемент прибавляется к сумме отдельно. Такой метод позволяет частично компенсировать ошибки округления, возникающие при работе с числами разного порядка величины.

# Руководство пользователя

Чтобы начать работу с программой, сначала нужно определить, какую функцию вы хотите использовать (см. рис. 1). Затем следует задать число элементов ряда (см. рис. 2). После этого необходимо указать значение аргумента для выбранной функции (см. рис. 3). Когда все данные введены, программа выведет на экран результат вычисления функции, полученный с использованием моей реализации и реализации через библиотеку math.h, а также максимальную абсолютную разницу между этими значениями (см. рис. 4).

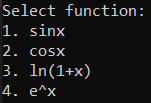


Рисунок 1. Выбор функции



Рисунок 2. Ввод числа членов ряда



Рисунок 3. Ввод значения аргумента

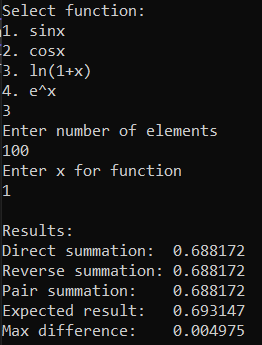


Рисунок 4. Вывод результатов

# Описание программной реализации

Данная программа на языке C реализует вычисление значений математических функций (синус, косинус, натуральный логарифм, экспонента) с использованием рядов Тейлора и сравнение результатов прямого, обратного и попарного суммирования с ожидаемыми значениями из библиотеки math.h.

**Основные функции**

1. **Вычисление членов ряда**
   * sinx, cosx, ln, expx: заполняют массив terms значениями ряда Тейлора для соответствующих функций с нормализацией аргумента x по 2π для периодических функций.
2. **Методы суммирования**
   * direct\_sum: Прямое суммирование от первого к последнему члену.
   * reverse\_sum: Обратное суммирование от последнего к первому.
   * pair\_sum: Попарное суммирование с учетом нечетного числа членов.
3. **Функция обработки**
   * filling: выбирает функцию, заполняет массив terms, вычисляет результаты суммирования и ожидаемое значение.
4. **Главная функция**
   * main: запрашивает у пользователя номер функции, количество членов ряда (n) и аргумент (x), выделяет память, вызывает filling и выводит результаты.

**Используемые библиотеки**

* <stdio.h>: Ввод-вывод (printf, scanf\_s).
* <stdlib.h>: Управление памятью (malloc, free).
* <math.h>: Математические функции и константа M\_PI.

# Подтверждение корректности

Для подтверждения корректности в программе использовались математические функции из библиотеки math.h. В результатах выводится значение, полученное math.h и разница по модулю между этим значением и полученным моими функциями.

# Результаты экспериментов

Чтобы получить результаты экспериментов, я записал погрешность для запусков с каждым n от 1 до 20, x = 5.0 для , и x = 1.0 для . С помощью библиотека matplotlib в питоне построил графики для каждого набора данных.

**Прямое суммирование**

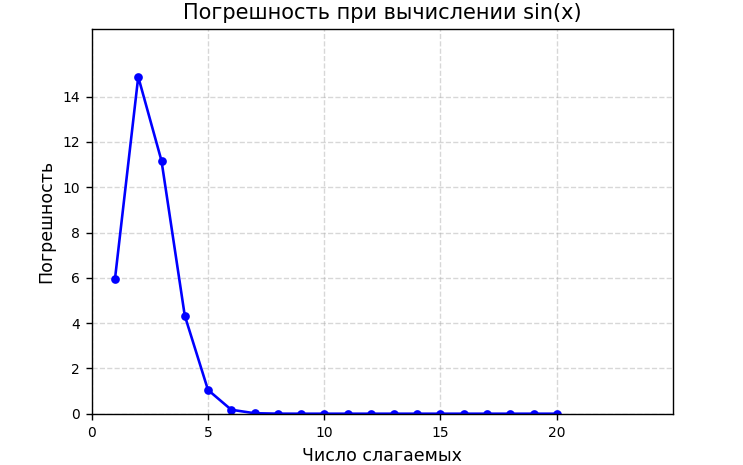
**sin(x)** n от 1 до 20, x = 5.0**** (рисунок 6)

Рисунок 6. sin(x)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 5.958924 |
| 2 | 14.874410 |
| 3 | 11.167258 |
| 4 | 4.333735 |
| 5 | 1.048554 |
| 6 | 0.174694 |
| 7 | 0.021340 |
| 8 | 0.001998 |
| 9 | 0.000147 |
| 10 | 0.000010 |
| 11 | 0.000000 |
| 12 | 0.000000 |
| 13 | 0.000000 |
| 14 | 0.000000 |
| 15 | 0.000000 |
| 16 | 0.000000 |
| 17 | 0.000000 |
| 18 | 0.000000 |
| 19 | 0.000000 |
| 20 | 0.000000 |

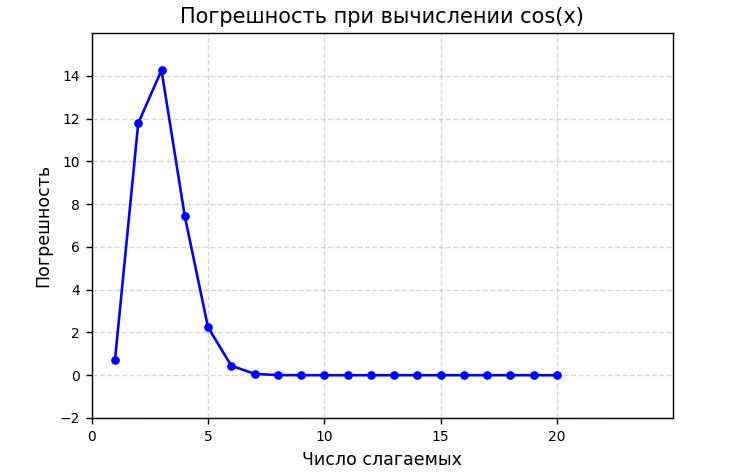
**cos(x)** n от 1 до 20, x = 5.0 (рисунок 7)

Рисунок 7. cos(x)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 0.716338 |
| 2 | 11.783662 |
| 3 | 14.258004 |
| 4 | 7.443384 |
| 5 | 2.244735 |
| 6 | 0.446409 |
| 7 | 0.063278 |
| 8 | 0.006734 |
| 9 | 0.000559 |
| 10 | 0.000037 |
| 11 | 0.000002 |
| 12 | 0.000000 |
| 13 | 0.000000 |
| 14 | 0.000000 |
| 15 | 0.000000 |
| 16 | 0.000000 |
| 17 | 0.000000 |
| 18 | 0.000000 |
| 19 | 0.000000 |
| 20 | 0.000000 |

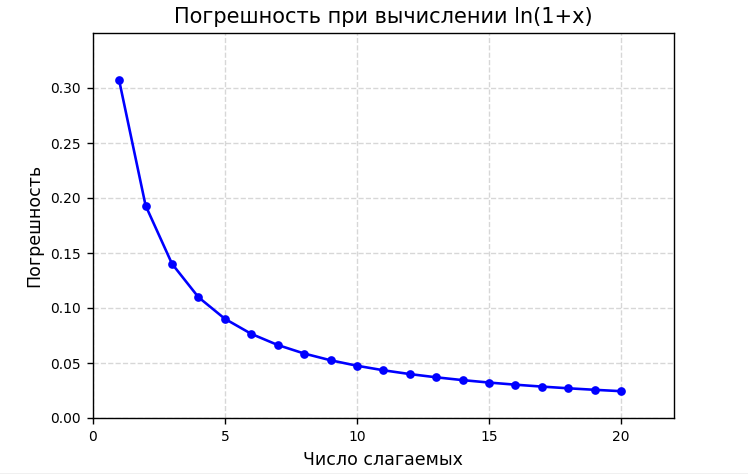
**ln(1 + x)** n от 1 до 20, x = 1.0 (рисунок 8)

Рисунок 8. ln(1 + x)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 0.306853 |
| 2 | 0.193147 |
| 3 | 0.140186 |
| 4 | 0.109814 |
| 5 | 0.090186 |
| 6 | 0.076481 |
| 7 | 0.066377 |
| 8 | 0.058623 |
| 9 | 0.052488 |
| 10 | 0.047512 |
| 11 | 0.043397 |
| 12 | 0.039936 |
| 13 | 0.036987 |
| 14 | 0.034442 |
| 15 | 0.032225 |
| 16 | 0.030275 |
| 17 | 0.028548 |
| 18 | 0.027007 |
| 19 | 0.025624 |
| 20 | 0.024376 |

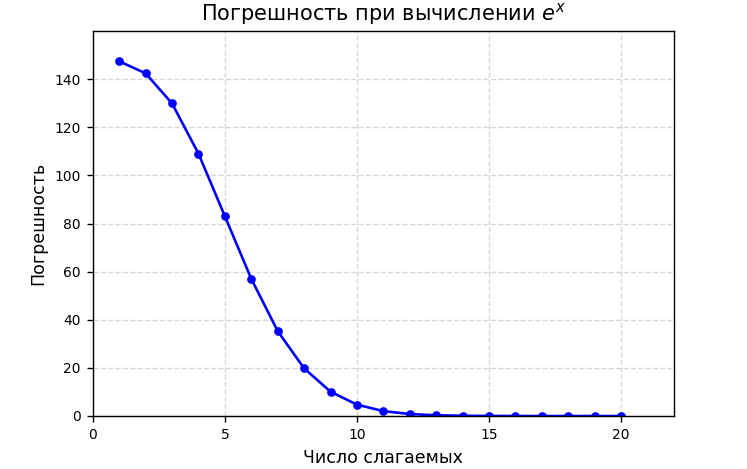
n от 1 до 20, x = 5.0 (рисунок 9)

Рисунок 9.

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 147.413162 |
| 2 | 142.413162 |
| 3 | 129.913162 |
| 4 | 109.079826 |
| 5 | 83.038162 |
| 6 | 56.996490 |
| 7 | 35.295097 |
| 8 | 19.794098 |
| 9 | 10.105972 |
| 10 | 4.723679 |
| 11 | 2.032532 |
| 12 | 0.809280 |
| 13 | 0.299591 |
| 14 | 0.103561 |
| 15 | 0.033554 |
| 16 | 0.010223 |
| 17 | 0.002930 |
| 18 | 0.000778 |
| 19 | 0.000183 |
| 20 | 0.000031 |

**Обратное суммирование**

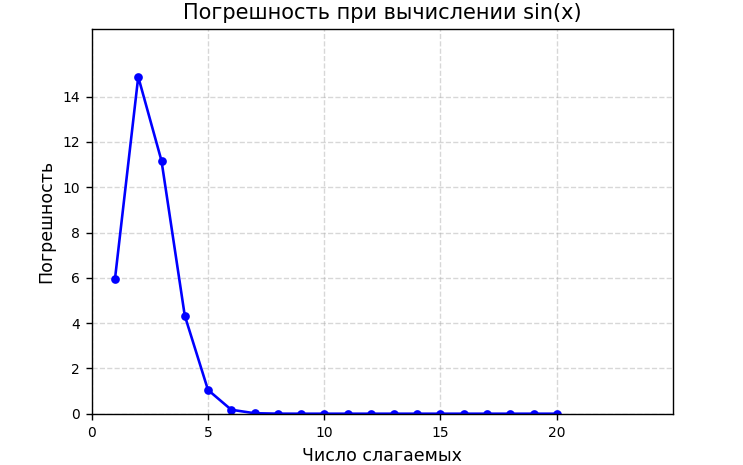
**sin(x)** n от 1 до 20, x = 5.0**** (рисунок 10)

Рисунок 10. sin(x)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 5.958924 |
| 2 | 14.874410 |
| 3 | 11.167258 |
| 4 | 4.333735 |
| 5 | 1.048554 |
| 6 | 0.174693 |
| 7 | 0.021339 |
| 8 | 0.001998 |
| 9 | 0.000148 |
| 10 | 0.000010 |
| 11 | 0.000000 |
| 12 | 0.000000 |
| 13 | 0.000000 |
| 14 | 0.000000 |
| 15 | 0.000000 |
| 16 | 0.000000 |
| 17 | 0.000000 |
| 18 | 0.000000 |
| 19 | 0.000000 |
| 20 | 0.000000 |

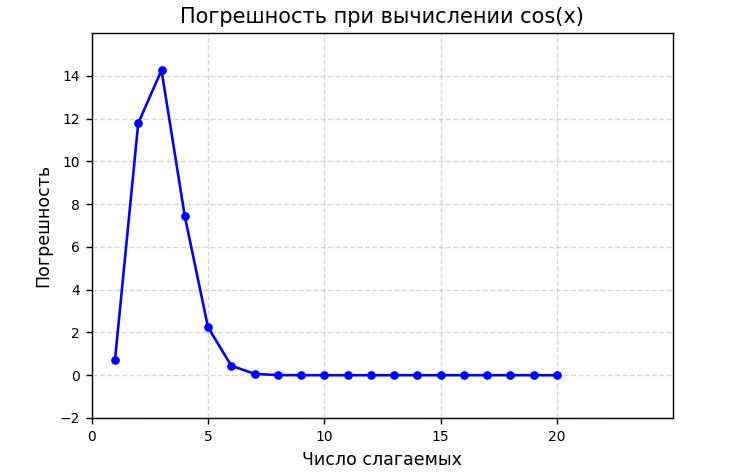
**cos(x)** n от 1 до 20, x = 5.0 (рисунок 11)

Рисунок 11. cos(x)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 0.716338 |
| 2 | 11.783662 |
| 3 | 14.258004 |
| 4 | 7.443384 |
| 5 | 2.244735 |
| 6 | 0.446409 |
| 7 | 0.063278 |
| 8 | 0.006734 |
| 9 | 0.000558 |
| 10 | 0.000038 |
| 11 | 0.000003 |
| 12 | 0.000000 |
| 13 | 0.000000 |
| 14 | 0.000000 |
| 15 | 0.000000 |
| 16 | 0.000000 |
| 17 | 0.000000 |
| 18 | 0.000000 |
| 19 | 0.000000 |
| 20 | 0.000000 |

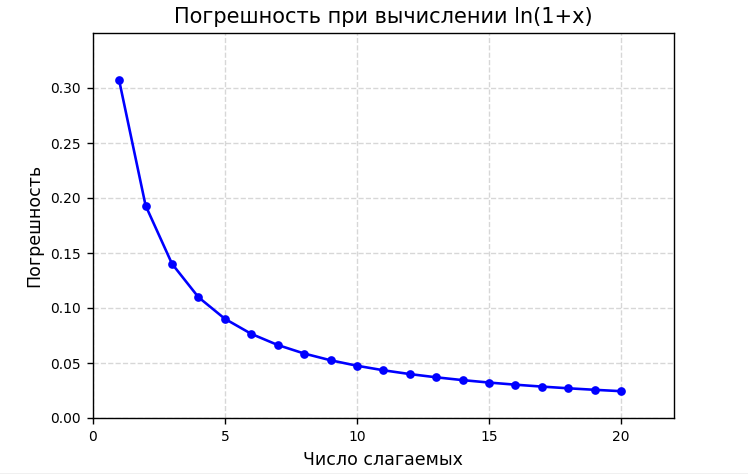
**ln(1 + x)** n от 1 до 20, x = 1.0 (рисунок 12)

Рисунок 12. ln(1 + x)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 0.306853 |
| 2 | 0.193147 |
| 3 | 0.140186 |
| 4 | 0.109814 |
| 5 | 0.090186 |
| 6 | 0.076481 |
| 7 | 0.066377 |
| 8 | 0.058623 |
| 9 | 0.052488 |
| 10 | 0.047512 |
| 11 | 0.043397 |
| 12 | 0.039936 |
| 13 | 0.036987 |
| 14 | 0.034442 |
| 15 | 0.032225 |
| 16 | 0.030275 |
| 17 | 0.028548 |
| 18 | 0.027007 |
| 19 | 0.025624 |
| 20 | 0.024376 |

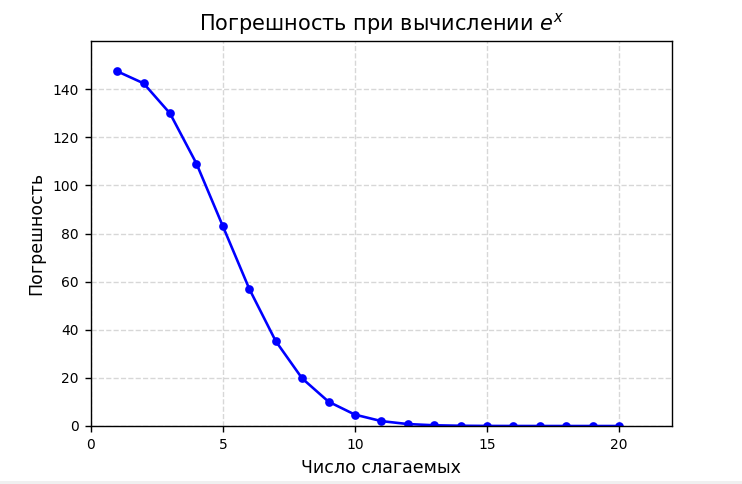
****

Рисунок 13.

n от 1 до 20, x = 5.0 (рисунок 13)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 147.413162 |
| 2 | 142.413162 |
| 3 | 129.913162 |
| 4 | 109.079826 |
| 5 | 83.038162 |
| 6 | 56.996490 |
| 7 | 35.295097 |
| 8 | 19.794113 |
| 9 | 10.105972 |
| 10 | 4.723679 |
| 11 | 2.032547 |
| 12 | 0.809311 |
| 13 | 0.299622 |
| 14 | 0.103592 |
| 15 | 0.033569 |
| 16 | 0.010239 |
| 17 | 0.002945 |
| 18 | 0.000793 |
| 19 | 0.000198 |
| 20 | 0.000046 |

**Попарное суммирование**

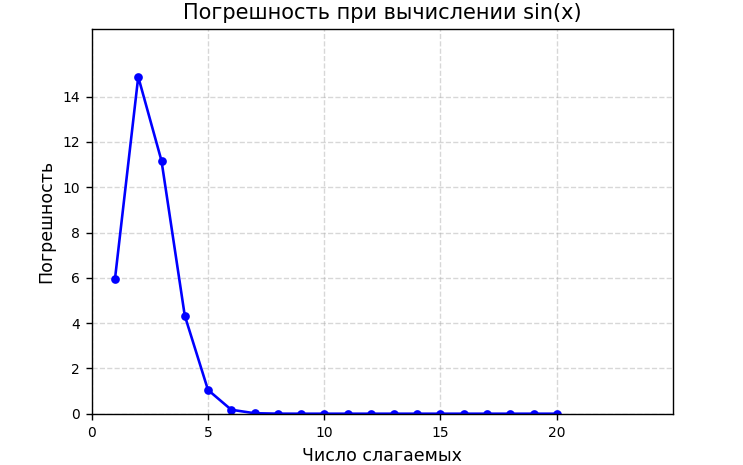
**sin(x)** n от 1 до 20, x = 5.0**** (рисунок 14)

Рисунок 14. sin(x)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 5.958924 |
| 2 | 14.874410 |
| 3 | 11.167258 |
| 4 | 4.333735 |
| 5 | 1.048554 |
| 6 | 0.174694 |
| 7 | 0.021340 |
| 8 | 0.001998 |
| 9 | 0.000147 |
| 10 | 0.000010 |
| 11 | 0.000000 |
| 12 | 0.000000 |
| 13 | 0.000000 |
| 14 | 0.000000 |
| 15 | 0.000000 |
| 16 | 0.000000 |
| 17 | 0.000000 |
| 18 | 0.000000 |
| 19 | 0.000000 |
| 20 | 0.000000 |

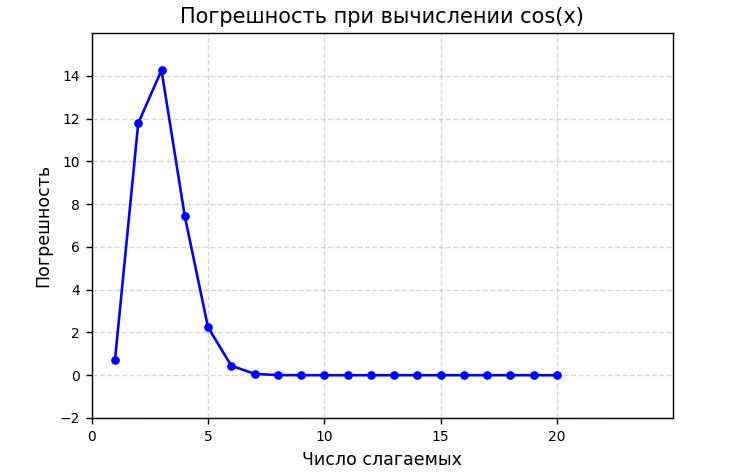
**cos(x)** n от 1 до 20, x = 5.0 (рисунок 15)

Рисунок 15. cos(x)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 0.716338 |
| 2 | 11.783662 |
| 3 | 14.258004 |
| 4 | 7.443384 |
| 5 | 2.244735 |
| 6 | 0.446409 |
| 7 | 0.063278 |
| 8 | 0.006734 |
| 9 | 0.000559 |
| 10 | 0.000037 |
| 11 | 0.000002 |
| 12 | 0.000000 |
| 13 | 0.000000 |
| 14 | 0.000000 |
| 15 | 0.000000 |
| 16 | 0.000000 |
| 17 | 0.000000 |
| 18 | 0.000000 |
| 19 | 0.000000 |
| 20 | 0.000000 |

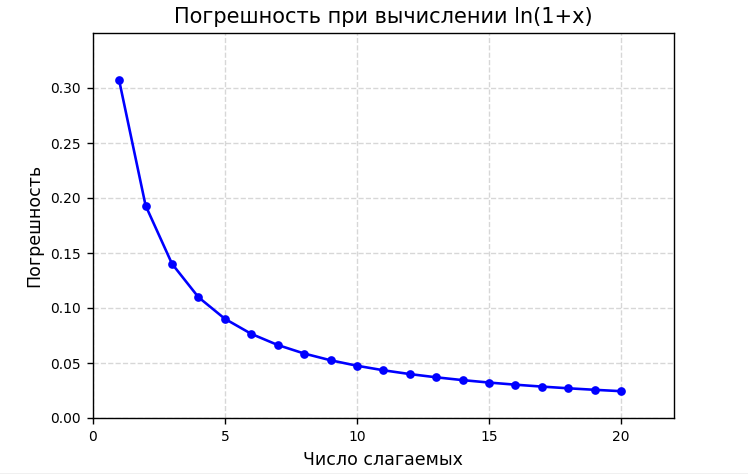
**ln(1 + x)** n от 1 до 20, x = 1.0 (рисунок 16)

Рисунок 16. ln(1 + x)

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 0.306853 |
| 2 | 0.193147 |
| 3 | 0.140186 |
| 4 | 0.109814 |
| 5 | 0.090186 |
| 6 | 0.076481 |
| 7 | 0.066377 |
| 8 | 0.058623 |
| 9 | 0.052488 |
| 10 | 0.047512 |
| 11 | 0.043397 |
| 12 | 0.039936 |
| 13 | 0.036987 |
| 14 | 0.034442 |
| 15 | 0.032225 |
| 16 | 0.030275 |
| 17 | 0.028548 |
| 18 | 0.027007 |
| 19 | 0.025624 |
| 20 | 0.024376 |

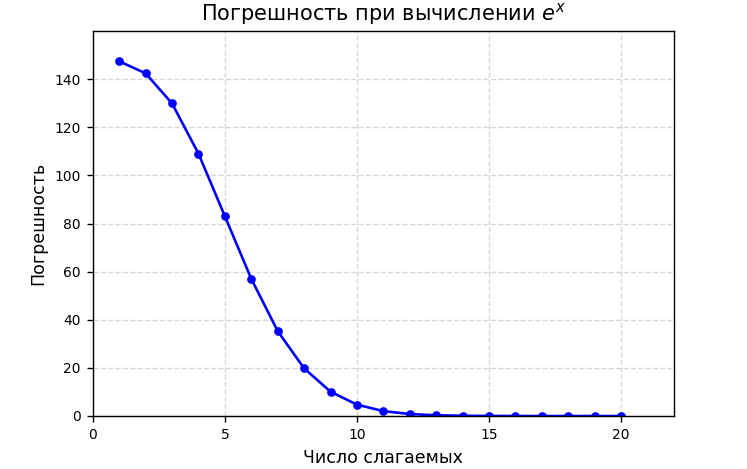
n от 1 до 20, x = 5.0 (рисунок 17)

Рисунок 17.

|  |  |
| --- | --- |
| n | Погрешность |
| 1 | 147.413162 |
| 2 | 142.413162 |
| 3 | 129.913162 |
| 4 | 109.079826 |
| 5 | 83.038162 |
| 6 | 56.996490 |
| 7 | 35.295097 |
| 8 | 19.794098 |
| 9 | 10.105972 |
| 10 | 4.723679 |
| 11 | 2.032532 |
| 12 | 0.809280 |
| 13 | 0.299591 |
| 14 | 0.103561 |
| 15 | 0.033554 |
| 16 | 0.010223 |
| 17 | 0.002930 |
| 18 | 0.000778 |
| 19 | 0.000183 |
| 20 | 0.000031 |

**Вывод:**

Проанализировав методы суммирования, можно сделать вывод что различия между ними незначительные, но в обратном суммировании есть несколько отличий. Например, для косинуса при n = 9, 10, 11 погрешность отличается на 0.000001 (меньше в 9 и больше в 10, 11). Для синуса при n = 6, 7, 9 погрешность в обратном суммировании также отличается на 0.000001 (меньше в 6, 7 и больше в 9). У экспоненты же значительные отличия также в обратном суммировании. При n = 8, 11, 15, 17, 18, 19, 20 погрешность больше на 0.000015. При n = 12, 13, 14 погрешность больше уже на 0.000031 и для n = 16 больше на 0.000016. Для подсчета значений функций необходимо большее число слагаемых в сумме, метод суммирования не так важен.

# Заключение

В ходе данной лабораторной работы было реализовано вычисление функций C:\Users\vkoli\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{FD5A0BA3-2565-4158-A41D-598DF71507DA}\{1AD2450C-C887-4F4B-9B12-8BC400C18F3B}\ResourceMap\{F78FB6B8-42AE-4AE6-A99D-AE46C180375B} C:\Users\vkoli\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{FD5A0BA3-2565-4158-A41D-598DF71507DA}\{1AD2450C-C887-4F4B-9B12-8BC400C18F3B}\ResourceMap\{F405AC61-FDE3-455B-BFC4-1ADA1D6D890D}, C:\Users\vkoli\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{FD5A0BA3-2565-4158-A41D-598DF71507DA}\{1AD2450C-C887-4F4B-9B12-8BC400C18F3B}\ResourceMap\{4E7B6E56-3AFA-4D88-AD23-73F69E92B924} и C:\Users\vkoli\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{FD5A0BA3-2565-4158-A41D-598DF71507DA}\{1AD2450C-C887-4F4B-9B12-8BC400C18F3B}\ResourceMap\{E7BE7AF9-8823-48C3-9848-81147573FB52}  с использованием разложения в ряд Тейлора. Кроме того, было проведено сравнение результатов, рассчитанных разными методами суммирования, с значениями, полученными с помощью стандартных функций из библиотеки math.h.

# Приложение

#include <stdio.h>

#include <stdlib.h>

#define \_USE\_MATH\_DEFINES

#include <math.h>

#define PI M\_PI

void sinx(float\* terms, int n, float x)

{

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

terms[0] = x;

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

terms[i] = (terms[i - 1] \* -1 \* x \* x) / ((2 \* i + 1) \* (2 \* i));

}

}

void cosx(float\* terms, int n, float x)

{

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

terms[0] = 1.0f;

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

terms[i] = (terms[i - 1] \* -1 \* x \* x) / ((2 \* i - 1) \* (2 \* i));

}

}

void ln(float\* terms, int n, float x)

{

terms[0] = x;

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

terms[i] = (terms[i - 1] \* -1 \* x \* i) / (i + 1);

}

}

void expx(float\* terms, int n, float x)

{

terms[0] = 1.0f;

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

terms[i] = (terms[i - 1] \* x) / i;

}

}

// Прямое суммирование

float direct\_sum(float\* terms, int n)

{

float fx = 0.0f;

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

fx += terms[i];

}

return fx;

}

// Обратное суммирование

float reverse\_sum(float\* terms, int n)

{

float fx = 0.0f;

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

fx += terms[i];

}

return fx;

}

// Попарное суммирование

float pair\_sum(float\* terms, int n)

{

float fx = 0.0f;

for (int i = 0; i < n / 2; i++) {

fx += terms[2 \* i];

if (2 \* i + 1 < n) {

fx += terms[2 \* i + 1];

}

}

if (n % 2 != 0) {

fx += terms[n - 1];

}

return fx;

}

void filling(float\* terms, int n, int func, float x,

float\* direct\_fx, float\* reverse\_fx,

float\* pair\_fx, float\* expected\_res)

{

if (func == 1) {

sinx(terms, n, x);

\*expected\_res = sin(x);

}

else if (func == 2) {

cosx(terms, n, x);

\*expected\_res = cos(x);

}

else if (func == 3) {

ln(terms, n, x);

\*expected\_res = log(1 + x);

}

else {

expx(terms, n, x);

\*expected\_res = exp(x);

}

\*direct\_fx = direct\_sum(terms, n);

\*reverse\_fx = reverse\_sum(terms, n);

\*pair\_fx = pair\_sum(terms, n);

}

int main()

{

int n, func;

float x, direct\_fx, reverse\_fx, pair\_fx, expected\_res;

printf("Select function:\n");

printf("1. sinx\n");

printf("2. cosx\n");

printf("3. ln(1+x)\n");

printf("4. e^x\n");

scanf\_s("%d", &func);

printf("Enter number of elements\n");

scanf\_s("%d", &n);

printf("Enter x for function\n");

scanf\_s("%f", &x);

float\* terms = (float\*)malloc(sizeof(float) \* n);

filling(terms, n, func, x, &direct\_fx, &reverse\_fx, &pair\_fx, &expected\_res);

printf("\nResults:\n");

printf("Direct summation: %f\n", direct\_fx);

printf("Reverse summation: %f\n", reverse\_fx);

printf("Pair summation: %f\n", pair\_fx);

printf("Expected result: %f\n", expected\_res);

printf("Max difference: %f\n", fmaxf(fmaxf(fabs(expected\_res - direct\_fx), fabs(expected\_res - reverse\_fx)), fabs(expected\_res - pair\_fx)));

free(terms);

return 0;

}