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**

**«Подсчет математический функций посредством разложения по формуле»**

**Выполнил**:

студентка группы 3822Б1ПМ1

Шарамыгина И.В.

**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

**Цель работы** – изучить, реализовать и сравнить алгоритмы вычисления значений синуса, косинуса, натурального логарифма и экспоненты.

**Задачи:**

* Изучить и реализовать алгоритмы прямого (от первого элемента до последнего) и обратного (от последнего до первого) суммирования.
* Проверить корректность работы алгоритмов.
* Сравнить точность алгоритмов.

# Метод решения

Для каждой функции выполним разложение в ряд Маклорена, получим следующие формулы:

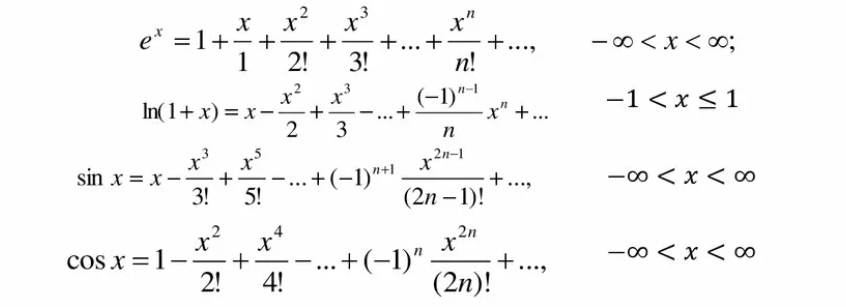


Рисунок 1. Разложение функций в ряд Маклорена

Так, получим, что каждое слагаемое можно выразить через предыдущее следующим образом:

Для синуса:

Для косинуса:

Для ln(1+x):

Для экспоненты:

Так как прямой порядок суммирование предполагает суммирование с больших величин, могут быть потеряны значащие разряды, снижена точность, поэтому реализуем обратное суммирование с меньших по значению слагаемых, где подсчет следующего слагаемого, будет реализован по формуле:

Для синуса:

Для косинуса:

Для ln(1+x):

Для экспоненты:

Реализуем проверку, чтобы слагаемое не стало меньше 10-38 (минимальное нормальное число больше 0 для float 2-126 ≈ 1.18 × 10-38) и весь ряд не обнулился.

# Руководство пользователя

При запуске программы необходимо ввести число x для подсчета функций.

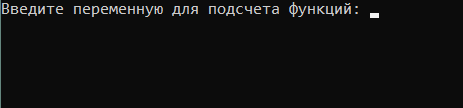


Рисунок 2 - Ввод переменной x

Программа выведет итог подсчета функций для заданной переменной, погрешности, а также канонические значения в следующем формате:

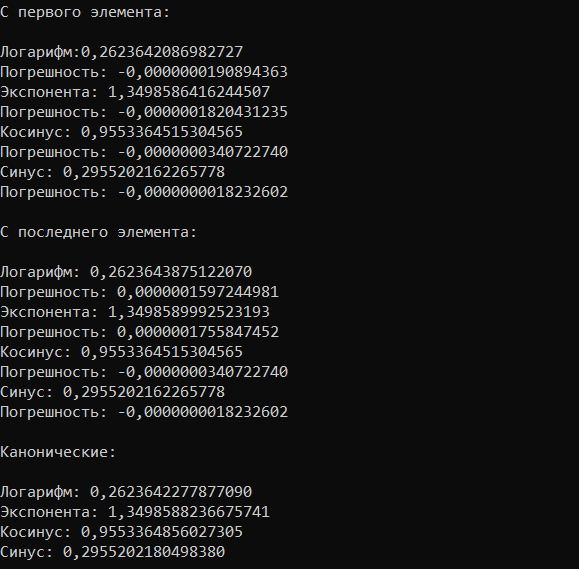


Рисунок 3 - Вывод в программе

При введении значения по модулю близкого к 1, логарифм определяется с меньшей точностью, больше либо равное 1 – очень неточно. Для введения нового значения программу нужно будет запустить заново.

# Описание программной реализации

Программа состоит из единственного файла (labwork2.cpp), в которой описаны все необходимые функции (10 шт).

float sum (float summa, float(\*anyfunc)(float, int, float), float x, float elem) – на вход принимает переменную, в которую будет считаться сумма, функция для подсчета нового слагаемого, переменная для подсчета функций и первый элемент. Выполняет суммирование ста элементов в прямом порядке.

float sumo(float summa, float(\*anyfunc)(float, int, float), float x, float elem, int n) – на вход принимает переменную, в которую будет считаться сумма, функция для подсчета нового слагаемого, переменная для подсчета функций, последний элемент и количество слагаемых. Выполняет суммирование в обратном порядке.

float sinus1(float x, int index, float elem) – на вход принимает переменную для подсчета функций, индекс данного слагаемого и само слагаемое. Производит подсчет следующего слагаемого функции синуса.

float cosinus1(float x, int index, float elem) – на вход принимает переменную для подсчета функций, индекс данного слагаемого и само слагаемое. Производит подсчет следующего слагаемого функции косинуса.

float log1(float x, int index, float elem) – на вход принимает переменную для подсчета функций, индекс данного слагаемого и само слагаемое. Производит подсчет следующего слагаемого функции логарифма.

float eshka1(float x, int index, float elem) – на вход принимает переменную для подсчета функций, индекс данного слагаемого и само слагаемое. Производит подсчет следующего слагаемого функции экспоненты.

float sinus2(float x, int index, float elem) – на вход принимает переменную для подсчета функций, индекс нового слагаемого и текущее слагаемое. Производит подсчет предыдущего слагаемого функции синуса.

float cosinus2(float x, int index, float elem) – на вход принимает переменную для подсчета функций, индекс нового слагаемого и текущее слагаемое. Производит подсчет предыдущего слагаемого функции косинуса.

float log2(float x, int index, float elem) – на вход принимает переменную для подсчета функций, индекс нового слагаемого и текущее слагаемое. Производит подсчет предыдущего слагаемого функции логарифма.

float eshka2(float x, int index, float elem) – на вход принимает переменную для подсчета функций, индекс нового слагаемого и текущее слагаемое. Производит подсчет предыдущего слагаемого функции экспоненты.

int main() – основная функция, отвечает за корректную работу, использование подпрограмм, взаимодействие с пользователем, возвращает 0 в случае корректной работы программы.

# Подтверждение корректности

Для подтверждения корректности в программе производится вывод канонических значений каждой из функций, а также погрешностей к значениям, полученным при прямом и обратном суммировании. В качестве эталонных используются функции sinf(x), cosf(x), expf(x) и logf(1+x) соответственно стандартной библиотеки math.h.

# Результаты экспериментов

Для определения алгоритма с лучшей точностью были проведены тесты для каждой из функций в диапазоне от 0 до 15 и от 0 до 1 для логарифма. Результаты приведены в графиках.



Рисунок 4. Таблица абсолютных погрешностей подсчета функций

Как мы видим, оба метода подсчета не являются достаточно точными, однако прямая сумма показывает меньшую погрешность для всех функций. Данный результат может быть объяснен следующим образом: при прямом суммировании погрешность рекурсивного подсчёта слагаемого влияет на результат единожды, а при обратном – дважды, на проходе от 1-го до n-го слагаемого и от n-го до 1-го, что говорит о большем влиянии погрешности подсчета слагаемых, нежели суммирования.

# Заключение

Итак, реализован подсчёт математических функций: sin(x), cos(x), ex, ln(1+x) (для -1<x<1) для аргумента, вводимого пользователем, типа данных float, через разложение в ряды Маклорена. В работе были реализованы два алгоритма поиска значений функции, рассмотрена их точность при рассмотрении абсолютной погрешности и проведено сравнение между самими алгоритмами.

# Приложение

float sum(float summa, float(\*anyfunc)(float, int, float), float x, float elem)

{

for (int i = 0; i < 100; i++)

{

elem = anyfunc(x, i, elem);

summa += elem;

}

return summa;

}

float sumo(float summa, float(\*anyfunc)(float, int, float), float x, float elem, int n)

{

summa += elem;

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

{

elem = anyfunc(x, i, elem);

summa += elem;

}

return summa;

}

float sinus1(float x, int index, float elem)

{

if (index == 0) return x;

elem = -elem \* x \* x / (2 \* index \* (2 \* index + 1));

return elem;

}

float sinus2(float x, int index, float elem)

{

if (index == 0) return x;

elem = -elem \* (2 \* index + 3) \* (2 \* index + 2) / (x \* x);

return elem;

}

float cosinus1(float x, int index, float elem)

{

if (index == 0) return 1;

elem = -elem \* x \* x / (2 \* index \* (2 \* index - 1));

return elem;

}

float cosinus2(float x, int index, float elem)

{

if (index == 0) return 1;

elem = -elem \* (2 \* index + 2) \* (2 \* index + 1) / (x \* x);

return elem;

}

float log1(float x, int index, float elem)

{

if (index == 0) return x;

elem = -elem \* x \* index / (index + 1);

return elem;

}

float log2(float x, int index, float elem)

{

if (index == 0) return 0;

elem = -elem \* (index + 1) / (index \* x);

return elem;

}

float eshka1(float x, int index, float elem)

{

if (index == 0) return 1;

elem = elem \* x / index;

return elem;

}

float eshka2(float x, int index, float elem)

{

if (index == 0) return 1;

elem = elem \* (index + 1) / x;

return elem;

}