### 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 синуса, косинуса, экспоненты, натурального логарифма»**

## Выполнил:

### студент группы 3821Б1ПМ2 Сятов Н.А.

## Проверил:

### преподаватель каф. МОСТ, Волокитин В.Д.

### Нижний Новгород 2021

**Содержание**

[Постановка задачи 3](file:///C:\Users\syato\1_КУРС\ЯиМП\лабараторная%20работа%201\mp1-3821B1PM2\SyatovNikita\Lab_Sort_1\Отчёт%20по%20лабароторной%20работе1.docx#_bookmark0)

[Метод решения 4](file:///C:\Users\syato\1_КУРС\ЯиМП\лабараторная%20работа%201\mp1-3821B1PM2\SyatovNikita\Lab_Sort_1\Отчёт%20по%20лабароторной%20работе1.docx#_bookmark1)

[Руководство пользователя 6](file:///C:\Users\syato\1_КУРС\ЯиМП\лабараторная%20работа%201\mp1-3821B1PM2\SyatovNikita\Lab_Sort_1\Отчёт%20по%20лабароторной%20работе1.docx#_bookmark2)

[Описание программной реализации 7](file:///C:\Users\syato\1_КУРС\ЯиМП\лабараторная%20работа%201\mp1-3821B1PM2\SyatovNikita\Lab_Sort_1\Отчёт%20по%20лабароторной%20работе1.docx#_bookmark3)

[Подтверждение корректности 8](file:///C:\Users\syato\1_КУРС\ЯиМП\лабараторная%20работа%201\mp1-3821B1PM2\SyatovNikita\Lab_Sort_1\Отчёт%20по%20лабароторной%20работе1.docx#_bookmark4)

[Результаты экспериментов 9](file:///C:\Users\syato\1_КУРС\ЯиМП\лабараторная%20работа%201\mp1-3821B1PM2\SyatovNikita\Lab_Sort_1\Отчёт%20по%20лабароторной%20работе1.docx#_bookmark5)

[Заключение 16](file:///C:\Users\syato\1_КУРС\ЯиМП\лабараторная%20работа%201\mp1-3821B1PM2\SyatovNikita\Lab_Sort_1\Отчёт%20по%20лабароторной%20работе1.docx#_bookmark6)

[Приложение 17](file:///C:\Users\syato\1_КУРС\ЯиМП\лабараторная%20работа%201\mp1-3821B1PM2\SyatovNikita\Lab_Sort_1\Отчёт%20по%20лабароторной%20работе1.docx#_bookmark7)

# Постановка задачи

Цель работы: разобрать идею и реализовать на языке C следующие математические функции: синус, косинус, экспонента, натуральный логарифм. Код данных функций я должен был реализовать с помощью рядов Маклорена для типа данных double. Нужно было описать алгоритмы, использованные в коде, подтвердить корректность результата функций, вычислить относительную и абсолютную погрешность, сделать вывод. Описать способы подтверждения корректности, сделать вывод, какой метод суммирования (прямой, обратный, попарный) более точен для определённых функций.

# Метод решения

Для подсчета данных функций я использовал ряды Маклорена:

* Экспонента



* Синус



* Косинус



* Логарифм натуральный (от -1 до 1)



Как видно по формулам, ряды Маклорена не имеют определённого последнего элемента, но мы реализуем функции в типе данных double, поэтому мы ограничены 15-16 знаками. Факториал растёт очень быстро, быстрее степенной, и в double до переполнения можно вывести 14!. Значит в синусе, косинусе и экспоненте мы ограничены 14 элементами суммирования, а в натуральном логарифме почти не ограничены. Есть три способа суммирования: прямой ход (с начала), обратный (с конца) и попарный(суммирование каждых двух после общее суммирование). Какой из них удобнее, узнаем на этапе экспериментов. Функция, проверяющая переполнение не имеет смысла в данной программе, потому что степенная функция переполнится при очень больших числах, при которых наша функция уже бесполезна.

# Руководство пользователя

# Пользователю предлагается ввести число, от которого нужно будет взять одну из функций, дальше предлагается выбрать одну из 4 функций (1 – экспонента, 2 – синус, 3 – косинус, 4 - логарифм). Через дефайн define можно изменить тип суммирования. Программа высчитывает результат, выводит его на экран, на следующую строку выводит результат, полученный с помощью стандартной функции из библиотеки, также подсчитывает абсолютную и относительную погрешность.

# Описание программной реализации

Я написал функцию, которая в зависимости от выбора пользователя считает нужную математическую функцию. Я использовал такие библиотеки: stdio.h, stdlib.h, math.h, locale.h.

Основные фунуции:

1. double factorial(double i) – высчитывает i! (применяется для подсчета последнего элемента в обратном ходе)
2. double deg(double x, double i) – высчитывает х в степени i (применяется для подсчета последнего элемента в обратном ходе)
3. double PrevExp (double element, double\* x, double i,),

double PrevSinCos(double element, double\* x, double i)

double PrevLog(double element, double\* x, double i) – высчитывает предыдущий элемент через указанный.

1. double NextSinCos(double element, double\* x, double i)

double NextExp(double element, double\* x, double i)

double NextLog(double element, double\* x, double i) - высчитывает следующий элемент через указанный.

1. void revers(double\* x, double\* arr, int choose) – заполняет массив элементами ряда маклорена в обратной порядке.
2. void pairs(double\* x, double\* arr, int choose) – заполняет массив попарно суммирование элементами.
3. void simple(double\* x, double\* arr, int choose) – заполняет массив элементами по порядку.
4. double summa(double\* arr) – суммирует элемнты массива.
5. double exp\_sin\_cos\_ln(double x, double\* arr, int choose) – основная функция с выбором функции и изменением способа суммирования.

# Подтверждение корректности

Для подтверждения корректности в программе использовал стандартные функции (sin(), cos(), exp(), log()) из библиотеки math.h. Высчитывается функция с помощью моей функции и с помощью стандартной функции. После я высчитываю относительную и абсолютную погрешность.

# Результаты экспериментов

Программа высчитывает относительную погрешность (%) каждой функции функцию двумя способами по очереди в интервале от 0,5 до 20 с шагом 0,5. Потом я строю графики на основании этих данных.

1. Экспонента

Вывод: экспонента очень быстро растёт и для х > 10 такой способ подсчёта не имеет смысл. НО на графиках заметно, что погрешность при обратном ходе намного меньше, значит, этот способ более точен. Прямое и попарное суммирование совпадают, потому что у экспоненты не меняется знак.

ЗАМЕЧАНИЕ: синус и косинус – периодические функции, поэтому для увеличения точности можно приводить х к одному периоду (от 0 до 2 пи). В данной работе нам нужно понять, какой способ суммирования точнее, а увеличение точности таким способом усложняет исследование. Для примера, вот график погрешности для синуса, если уменьшать х на период:

При сравнении, можно увидеть, что погрешность уменьшается примерно в 100 раз.

1. Синус

Вывод: по графику видно, что для синуса обратный и попарный ход намного точнее, чем прямой. Но как было сказано в замечании, для получения точного результата на больших числах, стоит уменьшать число на период.

1. Косинус

Вывод: с косинусом мы получаем аналогичную ситуацию, как и синусом. Обратный и попарный ход точнее.

1. Натуральный логарифм

Натуральный логарифм не имеет значения, если х не в диапозоне от -1 до 1, причем ряд считается от 1+х аргумента. Поэтому здесь я беру х от -1 до 1 с шагом 0,05. Также в ряде Маклорена от логарифма нет факториала, поэтому я беру 30 элементов суммирования.

Вывод: натуральный логарифм в окрестностях -1 и 1 начинает расходиться, из-за чего появляются большие погрешности, прямой ход считает точнее.

# Заключение

Я реализовал на языке С четыре функции: синуса, косинуса, экспоненты, натурального логарифма. Описал их алгоритмы работы, проверил корректность и вычислил погрешность, показал с помощью графиков, какой из трёх способов суммирования точнее для каждой функции.

# Приложение







