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**

**«Математические функции»**

**Выполнил**:

студент группы 3821Б1ПМ2

Фомичев Дмитрий Евгеньевич

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc103724798)

[МЕТОД РЕШЕНИЯ 4](#_Toc103724799)

[РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 5](#_Toc103724800)

[ОПИСАНИЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ 6](#_Toc103724801)

[ПОДТВЕРЖДЕНИЕ КОРРЕКТНОСТИ 10](#_Toc103724802)

[РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ 11](#_Toc103724803)

[ЗАКЛЮЧЕНИЕ 19](#_Toc103724804)

# ПОСТАНОВКА ЗАДАЧИ

― Реализовать математические функции: sin(x), cos(x), e^x, ln(x) – с разными видами суммирования ряда.

― Высчитать погрешность вычисления для каждого вида суммирования относительно библиотечных функций.

― Провести исследования для выяснения более точного вида суммирования.

― Сделать вывод о проделанной работе.

# МЕТОД РЕШЕНИЯ

Для вычисления приближенных значений функций применяется ряд Маклорена, частный случай ряда Тейлора.

Для синуса:



Для косинуса:



Для экспоненты:



Для натурального логарифма применялся не «ряд Меркатора» (который сходится только при -1 < x <= 1), а более удобный способ:

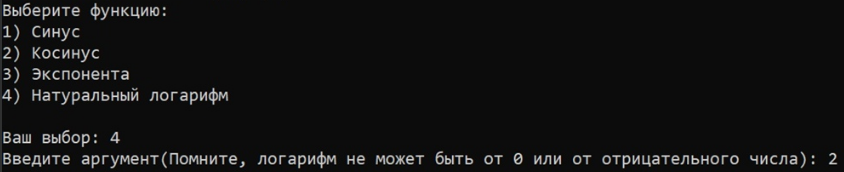


Для вычислений мы представляем x = (a - 1) / (a + 1), где а – аргумент функции.

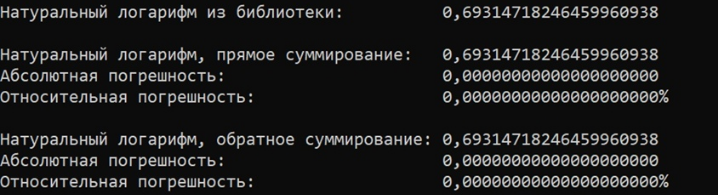
# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Лабораторная работа 2.sln

Запустите программу. На экране появится меню выбора функций (рис. 1). Выберите необходимую функцию. После этого введите аргумент функции. При выборе натурального логарифма появится напоминание о допустимых значениях аргумента.

Рис. 1

После ввода аргумента на экране появятся значения функции и информация о погрешности (рис. 2).

Рис. 2

Чтобы вернуться в меню выбора функций – введите любой символ.

# ОПИСАНИЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

Лабораторная работа 2.c

*main()*

В «мейне» содержатся функция установки русского языка *setlocale(LC\_ALL, "Russian")* и цикл while(1). В цикле реализовано меню выбора: математической функции и аргумента. Все реализовано с помощью оператора выбора switch/case. Внутри содержится функция system(“cls”), очищающая консоль.

void warning(int status)

Функция, основная задача которого вывод информации об области определения натурального логарифма. Появляется только при выборе логарифма.

void nul\_0(float\* array, int size)

Функция, заполняющая массив нулями. На вход принимает массив и его размер.

long long int fct(int a)

Факториал целого числа а.

float dtr(float a)

Функция приводит переменную а типа данных float к промежутку от 0 до 2\*Pi(где const float Pi = 3.141592653589). Допустим а < 0. Тогда к а будет прибавляться 2\*Pi, пока а не будет удовлетворять заданному промежутку. Если а > 2\*Pi, то будет вычитаться 2\*Pi. Дальше а передается в функцию tr().

float tr(float a)

Функция, определяющая квадрант плоскости, в которой находится переменная а. Если переменная находится в I или во II квадрантах(0 <= a <= Pi), то переменная возвращается без изменений. Если в III(Pi < a <= 3/2\*Pi), то из переменной вычитается Pi, а глобальная переменная st приравнивается к 3. Если переменная в IV квадранте(3/2\*Pi < a <= 2Pi), то из переменной вычитается Pi, а глобальная переменная st приравнивается к 4.

float tr\_sign(int status)

Функция, определяющая знак тригонометрических функций(синуса и косинуса). На вход подается переменная status, которая принимает значения 1 или 2 в зависимости от выбранной функции. Если значение status равняется 2 (выбрана функция косинуса), тогда при значении переменной st = 3 или st = 4 вернется -1. При других значениях st вернется 1. При status = 1(выбран синус) и st = 0 вернется 1, при других значениях st вернется -1.

float sign(int i, int status)

Функция определяет знак элемента ряда. На вход поступает порядковый номер элемента массива и информация о выбранной функции(1 – синус, 2 - косинус). При выбранном синусе: если остаток от деления порядкового номера элемента на четыре это 1 – функция возвращает 1, иначе -1. При выбранном косинусе: если остаток 0 – функция возвращает 1, иначе -1.

void error(float a, float b)

Функция принимает значение библиотечной и написанной функции.

Выводит информацию об абсолютной и относительной погрешностей.

float sum(float\* array, int size)

Функция прямого суммирования. На вход массив и размер массива. Элементы массива складываются в порядке возрастания их порядкового номера. На выход число типа float.

float reverse\_sum(float\* array, int size)

Функция обратного суммирования. На вход массив и размер массива. Элементы массива складываются в порядке убывания их порядкового номера.

На выход число типа float.

float duo\_sum(float\* array, int size)

Функция попарного суммирования. На вход массив и размер массива. Элемент массива складывается со своим соседом. После все пары складываются в порядке возрастания. На выход число типа float.

void sinus(float a)

Функция синуса. Принимает переменную типа float. Внутри функции создается массив на 20 элементов. Дальше массив передается в функцию nul\_0, которая заполняет массив нулями. В созданную переменную *lib* передается значение sinf(a) – функция синуса из библиотеки math.h типа float – от аргумента *а*. Переменная *а* передается в функцию dtr(). После этого массив заполняется по ряду Маклорена. В переменную sum1 записывается результат прямого суммирования умноженный на функцию tr\_sign(1). В переменную sum2 записывается результат обратного суммирования умноженный на функцию tr\_sign(1). В переменную sum3 записывается результат попарного суммирования умноженный на функцию tr\_sign(1). Значение глобальной переменной st меняется на ноль и выводится информация о вычислениях.

void cosinus(float a)

Функция косинуса. Принимает переменную типа float. Внутри функции создается массив на 20 элементов. Дальше массив передается в функцию nul\_0, которая заполняет массив нулями. В созданную переменную *lib* передается значение cosf(a) – функция косинуса из библиотеки math.h типа float – от аргумента *а*. Переменная *а* передается в функцию dtr(). После этого массив заполняется по ряду Маклорена. В переменную sum1 записывается результат прямого суммирования умноженный на функцию tr\_sign(2). В переменную sum2 записывается результат обратного суммирования умноженный на функцию tr\_sign(2). В переменную sum3 записывается результат попарного суммирования умноженный на функцию tr\_sign(2). Значение глобальной переменной st меняется на ноль и выводится информация о вычислениях.

void exponent(float a)

Экспонента. Принимает переменную типа float. Внутри функции создается массив на 20 элементов. Дальше массив передается в функцию nul\_0, которая заполняет массив нулями. После этого массив заполняется по ряду Маклорена. В переменную sum1 записывается результат прямого суммирования. В переменную sum2 записывается результат обратного суммирования. Выводится информация о вычислениях.

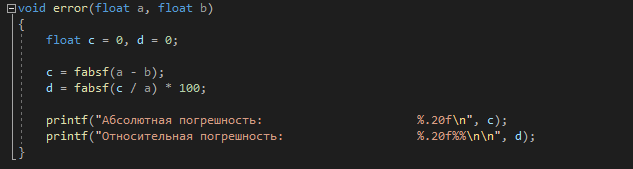
void ln(float a)

Натуральный логарифм. Принимает переменную типа float. Внутри функции создается массив на 20 элементов и переменная t. Переменная t нужна для разложения по ряду Маклорена. Дальше массив передается в функцию nul\_0, которая заполняет массив нулями. После этого массив заполняется по ряду Маклорена. В переменную sum1 записывается результат прямого суммирования. В переменную sum2 записывается результат обратного суммирования. Выводится информация о вычислениях.

# ПОДТВЕРЖДЕНИЕ КОРРЕКТНОСТИ

Для подтверждения корректности в программе считаются абсолютная и относительная погрешности вычислений. За эталон точности используются математические функции из библиотеки math.h языка Си – sinf(), cosf(), logf(), expf(). Абсолютная точность вычисляется по формуле: *модуль разности написанной и библиотечной функций.* Относительная точность вычисляется по формуле: *модуль частного абсолютной погрешности и библиотечной функции умноженный на 100%.*

За подсчет погрешностей отвечает функция void error(float *a*, float *b*), где *a* – библиотечная функция, а *b –* написанная. (рис. 3)

Рис. 3

# РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

**Sin**

Синус исследовался на отрезке от -5 до 5 с шагом 0,5. Также в особых точках: Pi, 3\*Pi/2 и 2\*Pi, где Pi = 3.141592653589. По таблице видно (Рис.5), что абсолютная погрешность обратного суммирования принимала значение 0 с точностью до 20 знаков после запятой 9 раз. Аналогично для прямого суммирования (Рис. 4): 7. Для попарного (Рис. 6) число «нулевых» строк абсолютной погрешности равно семи. Сравнивания остальные значения погрешности, можно сделать вывод, что более точной способ это обратное суммирование. Попарный чуть точнее прямого (Совсем на немного: 2 значения абсолютной погрешности попарного суммирования меньше чем у попарного, однако одно значение прямого меньше попарного).

Вывод: Для синуса более точный вид суммирования это обратный. Попарный немного точнее прямого суммирования.

Рис. 4

Рис. 5

Рис. 6

**COS**

Как и для синуса, диапазон исследования от -5 до 5 с шагом 0,5 и 3 точки: Pi, 3\*Pi/2, 2\*Pi (где Pi = 3.141592653589). Количество «нулевых» строк абсолютной погрешности обратного суммирования (Рис. 8) равно 9. Для прямого суммирования (Рис. 7): 7. Для попарного (Рис. 9): 5. Часть значений абсолютной погрешности обратного суммирования равнялась значениям прямого, однако 4 значения погрешности прямого суммирования были меньше обратного и 3 наоборот. Отсюда можно сделать вывод, что обратное суммирование чуть точнее прямого. Попарное суммирование оказалось не таким точным, как прямое. Из разных значений абсолютной погрешности 4 прямого суммирования оказались меньше попарного и только 1 больше.

Вывод: для косинуса более точный вид суммирования – обратный. Прямое суммирование точнее попарного.

Рис. 7

Рис. 8

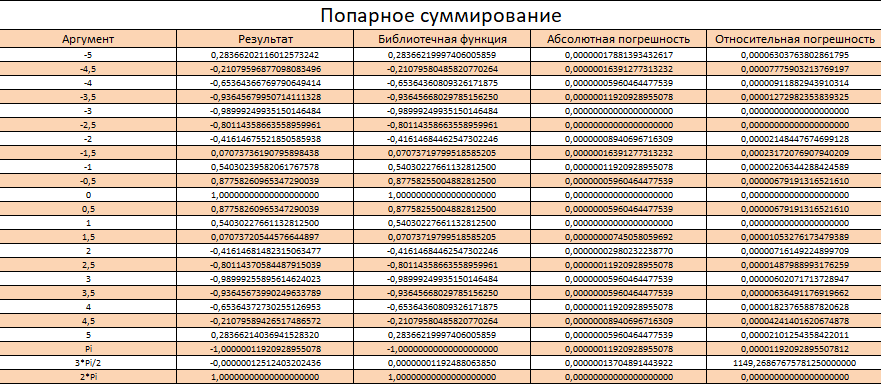


Рис. 9

**EXP**

Для экспоненты диапазон исследований был от -2,0 до -1,0 с шагом 0,2; 0; от 1 до 5 с шагом 1; от 10 до 21 с шагом 1. Количество «нулевых» строк абсолютной погрешности для обратного суммирования (Рис. 10) равно 6. Для прямого суммирования (Рис. 11): 3. На отрезке от -2.0 до -1.0 абсолютная погрешность обратного суммирования меньше прямого. Однако, начиная с 11, абсолютная погрешность прямого суммирования меньше обратного.

Вывод: Обратное суммирование более точный вид суммирования для экспоненты на аргументах меньше 11. С 11 прямое суммирование точнее.



Рис. 10



Рис. 11

**LN**

Натуральный логарифм исследовался на отрезке от 0,1 до 0,5 с шагом 0,1; 1,0; от 1,5 до 2,0 с шагом 0,1; от 2,0 до 5,0 с шагом 1,0; от 10,0 до 15,0 с шагом 1,0; от 20,0 до 22,0 с шагом 1,0. Количество «нулевых» строк абсолютной погрешности обратного суммирования (Рис. 12): 5. Прямого суммирования (Рис. 13): 4. В отличающихся строчках 4 раза обратное суммирование было точнее прямого. В 3 аналогичных строчках прямое было точнее.

Вывод: обратное суммирование точнее прямого.



Рис. 12



Рис.13

# ЗАКЛЮЧЕНИЕ

Я реализовал и исследовал 4 функции - sin, cos, exp, ln - разными видами суммирования: прямое, обратное и попарное (для синуса и косинуса). Высчитал погрешность вычислений для каждого вида суммирования относительно библиотечных функций. В результате, во всех функциях самый точный вид суммирования – обратное суммирование. У синуса попарное суммирование точнее прямого. У косинуса прямое суммирование точнее попарного.