МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский Авиационный Институт»

(Национальный Исследовательский Университет)

Институт №8: «Информационные технологии и прикладная математика»

Кафедра 806: «Вычислительная математика и программирование»

**КУРСОВАЯ РАБОТА №3**

По курсу «Вычислительные системы»

I семестр

Тема:

«Вещественный тип. Приближенные вычисления. Табулирование функций»

**Группа:** М80-106Б-22

**Студент:** Ларченко А.О

**Преподаватели:** Дубинин А.В.

**Оценка: \_\_\_\_\_\_\_\_\_ Подпись: \_\_\_\_\_\_\_\_**

**Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Москва, 2023

# Содержание

[**Содержание**](#_thjnqjfokida) **2**

[**Введение**](#_g3z0t0q3evu7) **3**

[**Глава 1. Теоретическая часть**](#_ldhuzcqn95pj) **4**

[Представление вещественных чисел в компьютере](#_3xp91wrdlcp9) 4

[Вещественные типы данных](#_hqhtjxn6k7st) 4

[Машинный эпсилон](#_fxxfadnqqtzi) 5

[Ряды Тейлора([2])](#_574r5ou3nrsg) 5

[**Глава 2. Практическая реализация**](#_qo05s8d8a5b4) **6**

[Задание](#_6zxpc4e92hy0) 6

[Алгоритм](#_d0nvwgrrqkvb) 6

[Сценарий](#_wd7whw5n321) 6

[Описание переменных](#_qgnb1duzjhxa) 7

[Протокол](#_1wfiv6jncz33) 7

[**Заключение**](#_tcc0rn4hdzyc) **11**

[**Список используемой литературы**](#_iegueqqpucgk) **12**

# Введение

Ещё в школе я познакомился с вещественными числами, узнал откуда они появляются и как с ними работать. И, естественно, когда я проводил разные вычисления на компьютере, не раз с ними сталкивался, но, честно говоря, особо не задумывался, как они представляются компьютером. Но уже в первом семестре мне открыли глаза на эту “тайну”, и так я познакомился с мантиссой, машинным эпсилоном и вообще представлением числа с плавающей точкой в компьютере.

Цель 3 курсового проекта не только написать программу, работающую с числами с плавающей точкой, но и закрепить полученные в данной области знания.

# Глава 1. Теоретическая часть

## *Представление вещественных чисел в компьютере*

Вещественное число это - это числа, имеющие дробную часть.

В компьютере вещественные числа обычно представляются в виде чисел с плавающей запятой. Число с плавающей запятой состоит из набора отдельных двоичных разрядов, условно разделенных на так называемые **знак**, **порядок(** *экспонентой или просто показателем степени***)** и **мантиссу**.

Первым делом, при работе с числом компьютер его нормализует(оставляет одну значащую единичку слева от запятой, а все остальные числа переносятся за запятую).

Потом определяет **знак**, если число положительное, то 0, отрицательное - 1.

Затем записывает **порядок** следующее 8 “клеточек” ( для float) / 11 (для double). если порядок положительный 10^10(в двоичной), то к (n - количество “клеточек” под порядок ) прибавляем показатель степени, в противном случае вычитаем.

Например, тип float, число 7.2 5 в двоичной: 1.1101 \* 10^10, тогда к 2^7(01111111) + 2(10) = 10000001.

После этого компьютер записывает числа, стоящие за запятой, в мантиссу

**

*(рис 1, источник* [*[1]*](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%87%D0%B8%D1%81%D0%B5%D0%BB_%D1%81_%D0%BF%D0%BB%D0%B0%D0%B2%D0%B0%D1%8E%D1%89%D0%B5%D0%B9_%D1%82%D0%BE%D1%87%D0%BA%D0%BE%D0%B9)*)для float*

Внимание! “1,” в компьютере не запоминается, тут она представлена только для наглядности

## *Вещественные типы данных*

float - 32 бита( 8 - порядок, 23 - мантисса)

double - 64 бита( 11 - порядок, 52 - мантисса)

double означает, что точность этого числа вдвое превышает точность числа типа float. В большинстве случаев тип double является наиболее удобным. Ограниченной точности чисел float во многих случаях попросту недостаточно. Причина, по которой тип float все еще используется, - экономия памяти при хранении (это важно для больших массивов вещественных чисел).

## *Машинный эпсилон*

Машинный эпсилон - это наименьшее число , такое что +1>1.

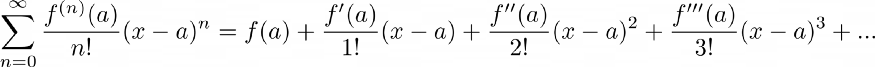
Иначе говоря, при округлении дробного числа до ближайшего к нему числа с плавающей точкой относительная погрешность округления меньше машинного эпсилона. Абсолютное значение «машинного эпсилон» зависит от разрядности компьютера.

## *Ряды Тейлора*([[2]](#_iegueqqpucgk))

Ряд Тейлора для функции представляет собой разложение функции в бесконечную сумму степенных членов, которая использует информацию о производных этой функции для создания многочлена, упрощающего эту функцию. Более точное упрощение можно вывести, взяв производные более высокого порядка и используя многочлены более высокой степени.

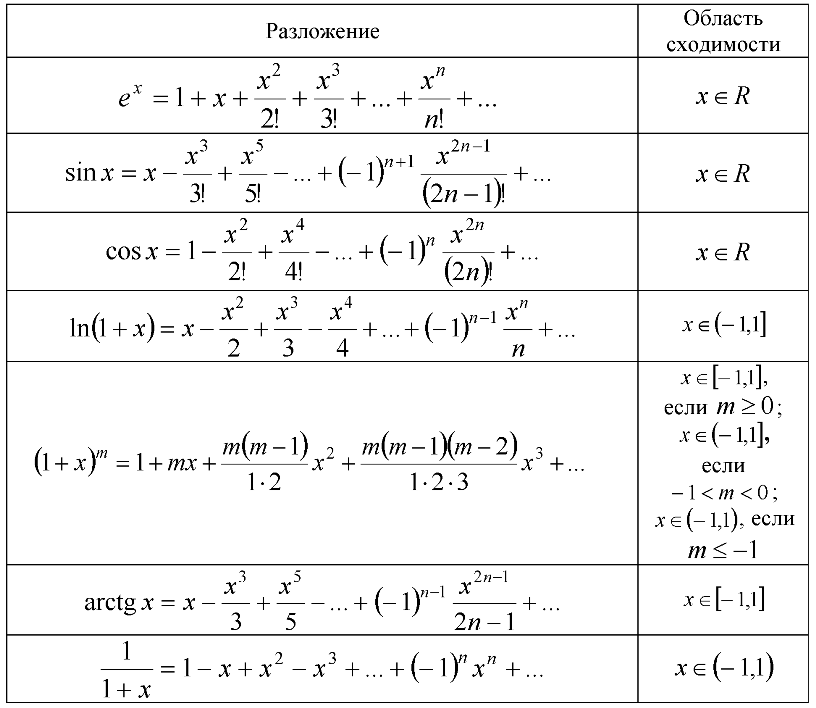
Другими словами ряд Тейлора показывает поведение функции в окрестности некоторой точки.

Членам ряда Тейлора функции f(x) требуются производные последовательно более высоких порядков f^(n)(x) — для определения коэффициентов полинома. Следовательно, ряд Тейлора может быть определен для f(x), только если она бесконечно дифференцируема. Члены ряда определяются выражением



Где а - окрестностью точки а. Если a=0, то ряд называют рядом Маклорена.

Разложение основных функций по Маклорену:



# Глава 2. Практическая реализация

## *Задание*

Необходимо составить программу на Си, которая печатает таблицу значений элементарной функции, вычисленной двумя способами: по формуле Тейлора и с помощью встроенных функций языка программирования. В качестве аргументов таблицы взять точки разбиения отрезка [a, b] на n равных частей, находящихся в рекомендованной области хорошей точности формулы Тейлора. Вычисления по формуле Тейлора проводить до экономной в сложностном смысле схеме с точностью , где - машинное эпсилон аппаратно реализованного вещественного типа для данной ЭВМ, – экспериментально подбираемый коэффициент, обеспечивающий приемлемую сходимость. Число итераций должно ограничиваться сверху числом порядка 100. Программа должна сама определять машинное и обеспечивать корректные размеры генерируемой таблицы.

Вариант 15. 

## *Алгоритм*

По условию задачи первым делом мы должны посчитать машинный эпсилон. Затем реализовать сравнение значений встроенной функций компилятора СИ(cos(x)) и ряда Тейлора, полученного путём сложения каждого элемента(высчитываемого по отдельности).

В конце необходимо вывести полученные результаты

## *Сценарий*

Разобьем нашу программу на подзадачи и объявим глобальные переменные(k, eps).

1. функция f(x) возвращает значение функции cos(x)
2. функция fact(x) возвращает значение факториала введённого числа, имеет тип int64\_t так как факториал быстрорастущая функция и значение быстро выходит за пределы int.
3. функция degree(x) получает на вход 2 числа - начальное число, степень, в которую необходимо возвести это число.
4. функция Teylor(x) высчитывает каждый элемент ряда Тейлора по отдельности пока элемент не будет меньше машинного эпсилона, затем складывает их в одну переменную - T
5. главная функция main, отвечающая за ввод и вывод результатов, также внутри неё просчитывается значение eps.

## *Описание переменных*

| Значение переменной | Тип | Назначение |
| --- | --- | --- |
| eps | double | показывает машинный эпсилон |
| a, b | double | границы отрезка |
| x | double | значение в промежутке [a, b] |
| h | double | шаг на отрезке |
| n | int | количество точек, в которых мы проверяем значение x |
| k | int64\_t | число итераций |
| T | double | значение ряда Тейлора |
| f(x) | функция double | возвращает значение функции |
| Teylor | функция double | возвращает значение ряда Тейлора |
| fact | массив int64\_t | возвращает значение факториала |
| degree | функция double | возведение в степень |

## *Протокол*

#include <stdio.h>

#include <math.h>

#include <inttypes.h>

int64\_t k;

double eps=1;

int64\_t fact[20];

double f(double x){

return cos(x);

}

double degree(double x, int n){

double a=1;

if (n==0){

return 1;

}

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

a=a\*x;

}

return a;

}

double Teylor(double x){

double T=0;

if (x==0){

return 1;

}

k=1;

double slag=1;

double x2=x\*x;

while ((slag>eps || -slag>eps) && k<=100){

T+=slag;

slag=degree(-1, k)\*(x2/fact[2\*k]);

x2=x2\*x\*x;

k++;

}

return T;

}

int main(){

int n;

double x, a=0, b=1, T,h;

x=a;

fact[0]=1;

fact[1]=1;

for (int i=2; i<21;++i){

fact[i]=fact[i-1]\*i;

}

printf("Введите число n ");

scanf("%d", &n);

while (1+eps/2>1){

eps=eps/2;

}

printf("\n Машинный епсилон = %e \n\n", eps); // .19 - count signs after point

printf("-----------------------------------------------------------------\n");

printf("| x | Teylor | f | k |\n");

printf("-----------------------------------------------------------------\n");

h=(b-a)/(n-1);

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

T=Teylor(x);

printf("|%.2f | %.19f | %.19f | %ld | \n", x, T, f(x), k);

x+=h;

}

printf("-----------------------------------------------------------------\n");

}

arsenii@LarchCompu:~/Documents/prog/projects/ones/Cprog$ gcc -std=c99 -pedantic kp3.c -lm

arsenii@LarchCompu:~/Documents/prog/projects/ones/Cprog$ ./a.out

Введите число n 5

Машинный эпсилон = 2.220446e-16

-----------------------------------------------------------------

| x | Teylor | f | k |

-----------------------------------------------------------------

|0.00 | 1.0000000000000000000 | 1.0000000000000000000 | 0 |

|0.25 | 0.9689124217106446224 | 0.9689124217106447334 | 7 |

|0.50 | 0.8775825618903727587 | 0.8775825618903727587 | 9 |

|0.75 | 0.7316888688738208968 | 0.7316888688738208968 | 10 |

|1.00 | 0.5403023058681397650 | 0.5403023058681397650 | 10 |

-----------------------------------------------------------------

arsenii@LarchCompu:~/Documents/prog/projects/ones/Cprog$ ./a.out

Введите число n 20

Машинный епсилон = 2.220446e-16

-----------------------------------------------------------------

| x | Teylor | f | k |

-----------------------------------------------------------------

|0.00 | 1.0000000000000000000 | 1.0000000000000000000 | 0 |

|0.05 | 0.9986152781425826630 | 0.9986152781425826630 | 6 |

|0.11 | 0.9944649474795757005 | 0.9944649474795755895 | 6 |

|0.16 | 0.9875605021181479426 | 0.9875605021181478316 | 7 |

|0.21 | 0.9779210635311099686 | 0.9779210635311100797 | 7 |

|0.26 | 0.9655733276010716182 | 0.9655733276010717292 | 8 |

|0.32 | 0.9505514906876966208 | 0.9505514906876966208 | 8 |

|0.37 | 0.9328971549228100013 | 0.9328971549228097793 | 8 |

|0.42 | 0.9126592129956349053 | 0.9126592129956349053 | 8 |

|0.47 | 0.8898937127472433817 | 0.8898937127472433817 | 9 |

|0.53 | 0.8646637019492131593 | 0.8646637019492131593 | 9 |

|0.58 | 0.8370390536963739914 | 0.8370390536963741024 | 9 |

|0.63 | 0.8070962728972043143 | 0.8070962728972044253 | 9 |

|0.68 | 0.7749182843977932000 | 0.7749182843977933111 | 9 |

|0.74 | 0.7405942033261461166 | 0.7405942033261461166 | 10 |

|0.79 | 0.7042190882928545115 | 0.7042190882928545115 | 10 |

|0.84 | 0.6658936781316238118 | 0.6658936781316237008 | 10 |

|0.89 | 0.6257241129087433018 | 0.6257241129087431908 | 10 |

|0.95 | 0.5838216399741469331 | 0.5838216399741470441 | 10 |

|1.00 | 0.5403023058681402091 | 0.5403023058681400981 | 10 |

-----------------------------------------------------------------

# 

# Заключение

В ходе выполнения задания я проделал большую работу: первым делом я ознакомился с теорией о числах с плавающей точкой и ряде Тейлора, потом я составил алгоритм выполнения программы, а затем перешёл к её реализации на СИ. В этом КП я научился работать с математической библиотекой в СИ, высчитывать машинный эпсилон, а также закрепил знания о представлении чисел с плавающей точкой на компьютере.

В ходе выполнения работы я столкнулся с некоторыми сложностями, например, в какой-то момент программа начинала выдавать значения nan, как я установил впоследствии, это было связано с факториалом, который быстро увеличивал своё значение и типа int этой функции не хватало, эту ошибку я исправил изменением типа на int64\_t (для этого пришлось подключить библиотеку *<inttypes.h>*).

Также я узнал как выводить фиксированное количество цифр после запятой(%.19lf) и как компилировать программы с математической библиотекой (ключ -lm).

В конечном итоге у меня получилось написать исправно-работающую программу, поэтому я считаю, что справился с поставленной задачей.

# Список используемой литературы

[1][Представление чисел с плавающей точкой — Викиконспекты](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%87%D0%B8%D1%81%D0%B5%D0%BB_%D1%81_%D0%BF%D0%BB%D0%B0%D0%B2%D0%B0%D1%8E%D1%89%D0%B5%D0%B9_%D1%82%D0%BE%D1%87%D0%BA%D0%BE%D0%B9)

<https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%87%D0%B8%D1%81%D0%B5%D0%BB_%D1%81_%D0%BF%D0%BB%D0%B0%D0%B2%D0%B0%D1%8E%D1%89%D0%B5%D0%B9_%D1%82%D0%BE%D1%87%D0%BA%D0%BE%D0%B9>

[2] Битюков Ю.И.: «Лекции по математическому анализу. Курс лекций 1 семестр».

<https://habr.com/ru/company/otus/blog/651659/>