Федеральное государственное бюджетное

образовательное учреждение

высшего образования

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ»**

(национальный исследовательский университет)

Факультет № 8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**КУРСОВОЙ ПРОЕКТ**

**«Вещественный тип. Приближённые вычисления.   
Табулирование функций»**

по дисциплине «Вычислительные системы»

1 семестр

|  |  |
| --- | --- |
| Студент | Мезенин О.А. |
| Группа | М8О-106Б-21 |
| Преподаватель | Дубинин А.В. |
| Оценка |  |

Оглавление

[Оглавление 2](#_Toc91540293)

[Введение 3](#_Toc91540294)

[IEEE 754 4](#_Toc91540295)

[Основные понятия в представлении чисел с плавающей запятой 4](#_Toc91540296)

[Описание преобразования чисел 5](#_Toc91540297)

[Общее представление нормализованных чисел 6](#_Toc91540298)

[Представление денормализованного числа и других чисел 7](#_Toc91540299)

[Погрешность и округление чисел 10](#_Toc91540300)

[Машинный эпсилон 12](#_Toc91540301)

[Ряд Тейлора 13](#_Toc91540302)

[Постановка задачи 15](#_Toc91540303)

[Описание программы 16](#_Toc91540304)

[Тесты 18](#_Toc91540305)

[Заключение 20](#_Toc91540306)

[Список источников 21](#_Toc91540307)

# Введение

Компьютер способен обрабатывать информацию гораздо быстрее, чем мозг человека, поэтому он применяется во многих сферах нашей жизни, особенно там, где нужны быстрые и точные вычисления а также работа с большими наборами данных. Но память в компьютере ограничена, а значит, ограничен диапазон представляемых чисел. Особенно это касается точности вещественных чисел, которые часто используются в таких точных науках как, например, математике или физике.

Цель курсовой работы – ознакомиться со стандартом IEEE 754, описывающий представление чисел с плавающей запятой в ЭВМ, а также провести сравнение значений заданной функции двумя способами: с помощью встроенных функций языка программирования Си и по формуле Тейлора.

# IEEE 754

Данный стандарт разработан ассоциацией IEEE (Institute of Electrical and Electronics Engineers) и используется для представления действительных чисел (чисел с плавающей точкой) в двоичном коде. Наиболее используемый стандарт для вычислений с плавающей точкой, используется многими микропроцессорами и логическими устройствами, а также программными средствами.

Полное название стандарта в ассоциации IEEE:

IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985)

В 2008 года ассоциация IEEE выпустила стандарт IEEE 754-2008, который включил в себя стандарт IEEE 754-1985.

**Стандарт IEEE 754-1985 определяет:**

* как представлять нормализованные положительные и отрицательные числа с плавающей точкой
* как представлять денормализованные положительные и отрицательные числа с плавающей точкой
* как представлять нулевые числа
* как представлять специальную величину "бесконечность" (Infinity)
* как представлять специальную величину "Не число" (NaN или NaNs)
* четыре режима округления

**IEEE 754-1985 определяет четыре формата представления чисел с плавающей запятой:**

* с одинарной точностью (single-precision) 32 бита
* с двойной точностью (double-precision) 64 бита
* с одинарной расширенной точностью (single-extended precision) >= 43 бит (редко используемый)
* с двойной расширенной точностью (double-extended precision) >= 79 бит (обычно используют 80 бит)

Основные понятия в представлении чисел   
с плавающей запятой

Представление чисел в памяти компьютера основывается на *нормализованном* *экспоненциальном виде*. Например, десятичное число 155,625 имеет следующую запись в нормализованном экспоненциальном виде: 1,55625∙10+2 = 1,55625∙exp10+2. Это же число в двоичной системе счисления 10011011,1012 будет иметь следующий вид: 1,55625∙exp10+2 = 1,0011011101∙exp2+111.

Число 1,55625∙exp10+2 состоит из двух частей: мантиссы M=1.55625 и экспоненты exp10=+2. Экспонента представлена основанием системы исчисления (в данном случае 10) и порядком (в данном случае +2). Порядок экспоненты может иметь отрицательное значение, например число 0,0155625=1,55625∙exp10-2.

Если мантисса ≥ 1 и меньше основания системы счисления, то число считается **нормализованным**.

Описание преобразования чисел

Основное применение в технике и программирование получили форматы 32 и 64 бита. Например, в Си используют типы данных float (32 бита) и double (64 бита).

Описание преобразования в 32-х битный формат IEEE 754:

1. Число может быть положительное или отрицательное. Поэтому отводится 1 бит для обозначения знака числа:  
   0 - положительное  
   1 - отрицательное  
   Это самый старший бит в 32-х битной последовательности.
2. Далее пойдут биты экспоненты, для этого выделяют 1 байт (8 бит). Экспонента может быть, как и число, со знаком + или -. Для определения знака экспоненты, чтобы не вводить ещё один бит знака, добавляют смещение к экспоненте в половину байта +127(0111 1111). То есть, если наша экспоната = +7 (+111 в двоичной), то смещенная экспонента = 7+127=134. А если бы наша экспонента была -7, то смещенная экспонента была равна 127-7 =120. Смещенную экспоненту записывают в отведенные 8 бит.
3. Оставшиеся 23 бита отводят для мантиссы. Но у нормализованной двоичной мантиссы первый бит всегда равен 1, так как число лежит в диапазоне 1<=M<2. Нет смысла записывать эту единицу, поэтому в отведенные 23 бита записывают остаток от мантиссы.

В следующей таблице представлено число 155,62510 = 1,0011011101∙exp2+111 в 32-х битном формате IEEE754:

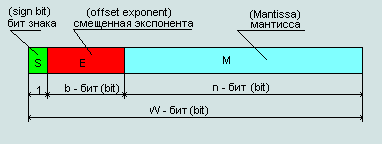
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 1. Представление числа 155,62510 в 32-х битном формате   |  |  |  |  | | --- | --- | --- | --- | | 1 бит | 8 бит | 23 бит | IEEE 754 | | 0 | 1000 0110 | 001 1011 1010 0000 0000 0000 | 431BA000 (hex) | | 0 (dec) | 134 (dec) | 1810432 (dec) |  | | знак числа | смещённая экспонента | остаток от мантиссы | число 155,625 в формате IEEE754 | |
|  |

Для остальных форматов точности преобразование аналогичное.

Общее представление нормализованных чисел

Формальное представление нормализованных чисел в стандарте IEEE 754 для любого формата точности представлено на следующем рисунке:

Рис. 1 Представление числа в формате IEEE 754



где:

* S - бит знака, если S=0 - положительное число; S=1 - отрицательное число
* E - смещенная экспонента двоичного числа, exp2 = E - (2(b-1) - 1) - экспонента двоичного нормализованного числа с плавающей точкой, (2(b-1) -1) - заданное смещение экспоненты
* M - остаток мантиссы двоичного нормализованного числа с плавающей точкой

Формула вычисления десятичных чисел с плавающей точкой из чисел, представленных в стандарте IEEE754:

Формула нормализованых чисел IEEE754 (Формула №1)

Используя формулу №1 вычислим формулы для нахождения десятичных чисел из форматов одинарной (32 бита) и двойной (64 бита) точности IEEE 754:

Рис.2 Формат числа одинарной точности (single-precision) 32 бита

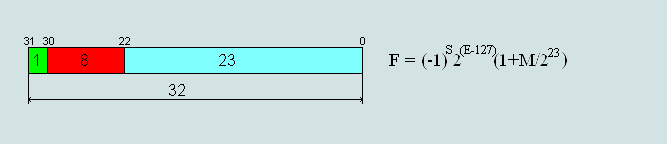
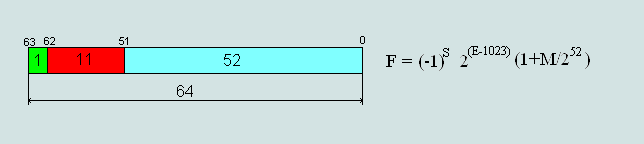


Рис.3 Формат числа двойной точности (double-precision) 64 бит

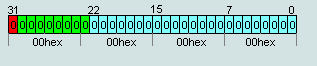
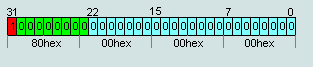
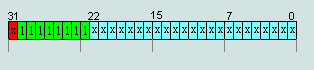


Представление денормализованного числа  
и других чисел

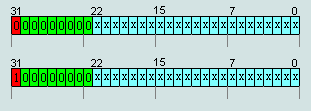
Если применить формулу №1 для вычисления минимального и максимального числа одинарной точности представленного в IEEE754, то получим следующие результаты:

* 00 00 00 00 (hex) = 5,87747175411144e-39 (минимальное положительное число)
* 80 00 00 00 (hex) =-5,87747175411144e-39 (минимальное отрицательное число)
* 7f ff ff ff (hex) = 6,80564693277058e+38 (максимальное положительное число)
* ff ff ff ff (hex) =-6,80564693277058e+38 (максимальное отрицательное число)

Поэтому формула №1 не применяется в следующих случаях:

1. Число 00 00 00 00 (hex) считается числом +0  
     
   Число 80 00 00 00 (hex) считается числом -0  
   
2. Число 7F 80 00 00 (hex) считается числом +∞  
   число +∞ в формате 32 бит IEEE754   
   число FF 80 00 00 (hex) считается числом -∞  
   число -∞ в формате 32 бит IEEE754 
3. Числа FF (1xxx)X XX XX (hex) и 7F (1xxx)X XX XX (hex) не считается числами (not a number, NaN), кроме случая п.2  
   Число представленное в битах с 0...22 могут быть любым числом кроме 0 (т.е.+∞ и -∞ ).  
   

NaN может получиться в результате операций с неопределённым результатом (например, 0/0). По определению NaN ≠ NaN.

1. Числа (x000) (0000) (0xxx)X XX XX (hex) считаются денормализованными числами, за исключением чисел п.1 (то есть -0 и +0)  
   

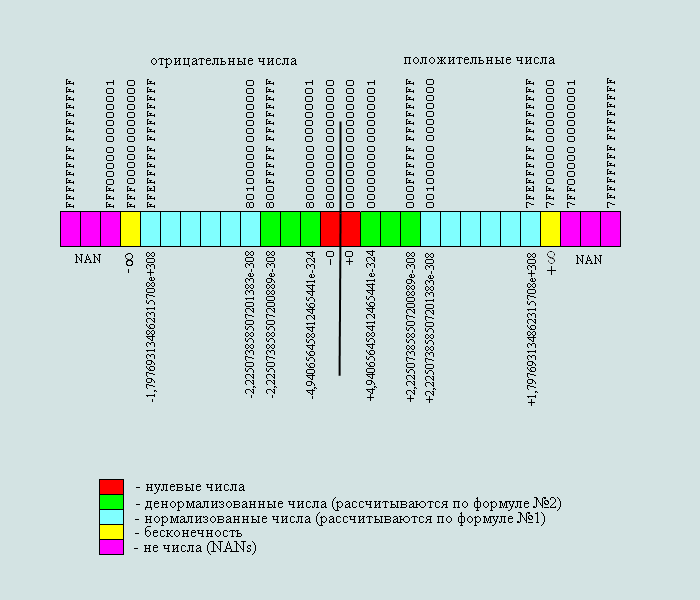
*Денормализованные числа* – это способ увеличить количество представимых числом значений около нуля для повышения точности вычисления.

Формула расчета денормализованных чисел:

Формула для 32битых денормализованных IEEE754 (Формула №2)

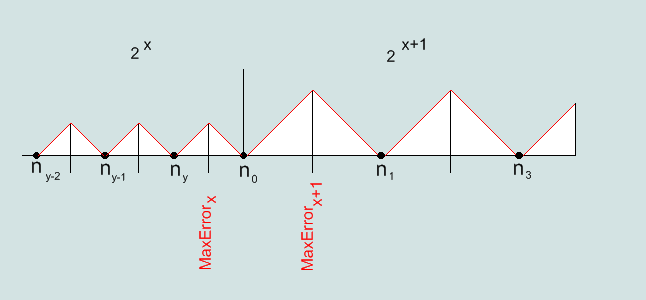
Полный диапазон чисел двойной точности представлен на следующем рисунке:

Рис.4 Диапазон чисел формата двойной точности (64 бита) представленных по стандарту IEEE 754



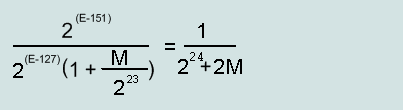
Погрешность и округление чисел

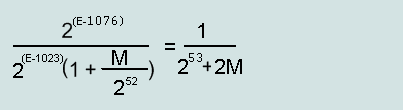
Числа представленные в формате IEEE754 представляют конечное множество, на которое отображается бесконечное множество вещественных чисел. Поэтому исходное число может быть представлено в формате IEEE754 с ошибкой (погрешностью).

Рис.5 Функция ошибки точности представления числа в IEEE754  


Абсолютная максимальная ошибка для числа в формате IEEE754 равна в пределе половине шага чисел. Шаг чисел удваивается с увеличением экспоненты двоичного числа на единицу. То есть, чем дальше от нуля, тем шире шаг чисел в формате IEEE754 по числовой оси.

Шаг нормализованных чисел равен 2(E-150) (Single) и 2(E-1075) (Double). Соответственно предел максимальной абсолютной ошибки будет равен 1/2 шага числа: 2(E-151) (Single) и 2(E-1076) (Double). Относительная ошибка в % будет равна: (2(E-151)/F)\*100% (Single) и (2(E-1076)/F)\*100% (Double).

Максимальная относительная ошибка нормализованного числа(single):  


Максимальная относительная ошибка нормализованного числа(double):  


|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 2. Максимальная возможная ошибка для чисел Double | | | |
| IEEE754, hex | число, dec | абсолютная ошибка, dec | относительная, % |
| 00000000 00000001 | 2-1074 ≈4,940656e-324 | 2-1075≈2,470328e-324 | =50 |
| 00000000 00000002 | 2-1073 ≈9,881313e-324 | 2-1075≈2,470328e-324 | =25 |
| 00000000 00000032 | ≈2,470328e-322 | 2-1075≈2,470328e-324 | =1 |
| 000FFFFF FFFFFFFF | ≈2,225073e-308 | 2-1075≈2,470328e-324 | ≈1,110223e-14 |
| 00100000 00000001 | ≈2,225074e-308 | 2-1074 ≈4,940656e-324 | ≈2,220446e-14 |
| 2B2BFF2E E48E0530 | ≈1,0e-100 | 2-385 ≈1,268971e-116 | ≈1,268971e-14 |
| 3FF00000 00000000 | =1,0 | 2-52 ≈2,220446e-16 | ≈2,220446e-14 |
| 54B249AD 2594C37D | ≈1,0e+100 | 2280 ≈1,942669e+84 | ≈1,942669e-14 |
| 6974E718 D7D7625A | ≈1,0e+200 | 2612 ≈1,699641e+184 | ≈1,699641e-14 |
| 7FEFFFFF FFFFFFFF | ≈1,79769e+308 | 2971 ≈1,99584e+292 | ≈1,110223e-14 |

Стандарт IEEE754 предусматривает четыре способа округления чисел.

**Способы округления чисел по стандарту IEEE 754:**

1. Округление стремящееся к ближайшему целому.
2. Округление стремящееся к нулю.
3. Округление стремящееся к +∞
4. Округление стремящееся к -∞

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 3. Примеры округления чисел до десятых | | | | |
| исходное число | к ближ. целому | к нулю | к +∞ | к -∞ |
| 1,33 | 1,3 | 1,3 | 1,4 | 1,3 |
| -1,33 | -1,3 | -1,3 | -1,3 | -1,4 |
| 1,37 | 1,4 | 1,3 | 1,4 | 1,3 |
| -1,37 | -1,4 | -1,3 | -1,3 | -1,4 |
| 1,35 | 1,4 | 1,3 | 1,4 | 1,3 |
| -1,35 | -1,4 | -1,3 | -1,3 | -1,4 |

Как происходит округление показано на примерах в таблице 3. При преобразовании чисел необходимо выбрать один из способов округления. По умолчанию это первый способ - округление к ближайшему целому. Часто в различных устройствах используют второй способ - округление к нулю. При округлении к нулю нужно просто отбросить незначащие разряды числа, поэтому этот способ самый легкий в аппаратной реализации.

Машинный эпсилон

**Машинный эпсилон** – это *минимальное* положительное число, такое, что при прибавлении к нему единицы результат будет отличен от единицы: 1+ ɛ>1.

Машинное эпсилон характеризует длину мантиссы, т.е. точность, с которой могут производиться вычисления с плавающей запятой.

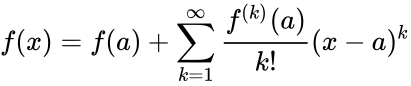
Величина машинного эпсилон — это величина относительной погрешности, т.е. погрешность представления чисел с порядком 0 (от 1 до 2) будет равна eps, а с порядком "x" погрешность будет равна 2x ∙ eps.

Практическая важность машинного эпсилон связана с тем, что два (отличных от нуля) числа являются одинаковыми с точки зрения машинной арифметики, если их относительная разность по модулю меньше машинного эпсилон.

# Ряд Тейлора

Ряд Тейлора – разложение функции в бесконечную сумму степенных функций. Ряд назван в честь английского математика Брука Тейлора.

Пусть функция f(x) бесконечно дифференцируема в некоторой окрестности точки a, тогда ряд



называется рядом Тейлора функции f в точке a.

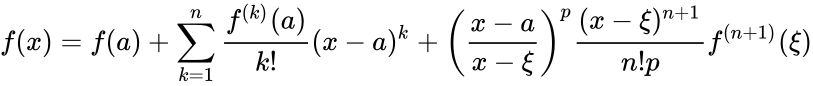
В случае, если a = 0, этот ряд иногда называется **рядом Маклорена**.

Формула Тейлора используется при доказательстве большого числа теорем в дифференциальном исчислении. Говоря нестрого, формула Тейлора показывает поведение функции в окрестности некоторой точки.

Теорема:

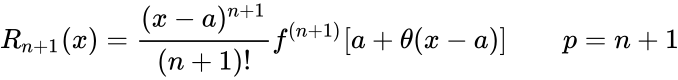
* Пусть функция f(x) имеет n+1 производную в некоторой окрестности точки a, U(a, ɛ);
* Пусть x **∈** U(a, ɛ);
* Пусть p — произвольное положительное число.

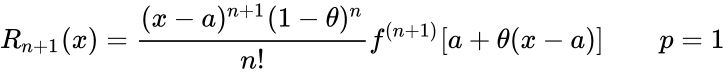
Тогда: ∃ точка ξ ∈ (x, a) при x < a или ξ ∈ (a, x) при x > a:

**

Это формула Тейлора с остаточным членом в общей форме (форме **Шлемильха — Роша**).

**Различные формы остаточного члена.**

В форме Лагранжа:

В форме Коши:

Остаточный член в асимптотической форме (форме Пеано):

*{\displaystyle R_{n+1}(x)=o[(x-a)^{n}]}*

# Постановка задачи

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 (n+1 включая концы отрезка), находящихся в рекомендованной области хорошей точности формулы Тейлора. Вычисления по формуле Тейлора проводить по экономной в сложностном смысле схеме с точностью *ɛ\*k*, где ɛ - машинное эпсилон аппаратно реализованного вещественного типа для данной ЭВМ, а k – экспериментально подбираемый коэффициент, обеспечивающий приемлемую сходимость. Число итераций должно ограничиваться сверху числом порядка 100. Программа должна сама определять машинное ɛ и обеспечивать корректные размеры генерируемой таблицы.

Вариант №4:

|  |  |  |  |
| --- | --- | --- | --- |
| ряд | a | b | функция |
|  | -1.0 | 1.0 | ln(2+x) |

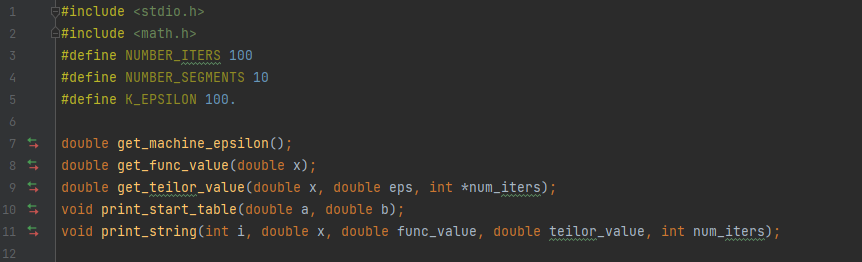
# Описание программы

Определим через define параметры:

* NUMBERS\_ITERS (максимальное количество итераций при вычислений по Тейлору);
* NUMBER\_SEGMENTS (количество точек отрезка [a, b]);
* K\_EPSILON (параметр k - экспериментально подбираемый коэффициент, обеспечивающий приемлемую сходимость).

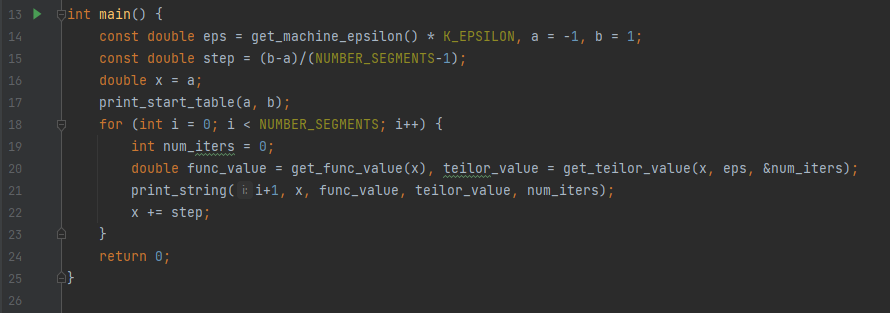
Объявим следующие функции:

* double get\_machine\_epsilon() – возвращает машинное эпсилон;
* double get\_func\_value(double x) – возвращает значение заданной функции;
* double get\_teilor\_value(double x, double eps, int \*num\_iters) – возвращает значение суммы разложения заданной функции по Тейлору. В num\_iters записывает количество пройденных итераций;
* void print\_start\_table(double a, double b, double eps) – печатает значение отрезка и машинного эпсилона, а также инициализирует таблицу;
* void print\_string(int i, double x, double func\_value, double tailor\_value, int num\_iters) – печатает строку в таблице.

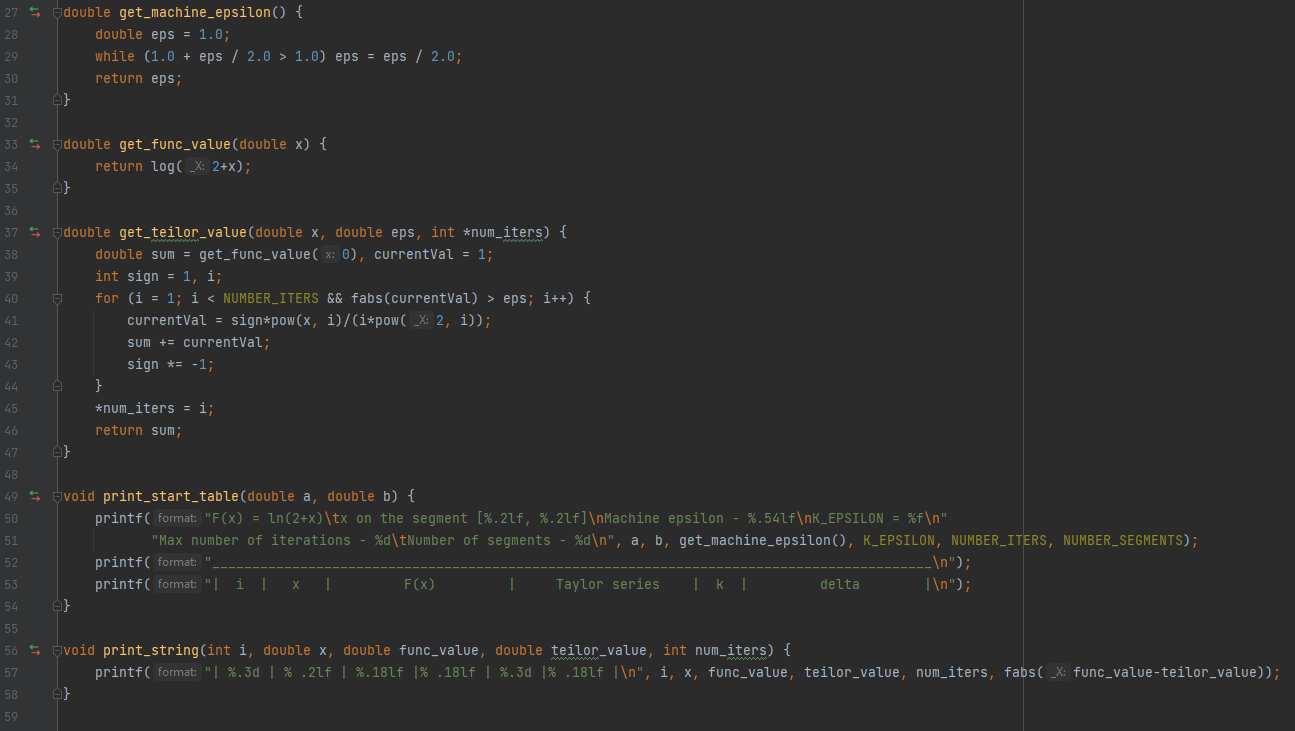


Алгоритм функции main:

1. Инициализация констант: eps = ɛ\*k, концы отрезка a и b, step – шаг, на который будет увеличиваться переменная x в пределах данного отрезка.
2. Инициализация переменной x, присваивание ей значения константы a.
3. Инициализация таблицы.
4. Цикл от 0 до NUMBER\_SEGMENTS. На каждой итерации вычисляются значения функции и суммы ряда Тейлора от переменной x, происходит вывод значений на экран, увеличивается значение переменной x на step.

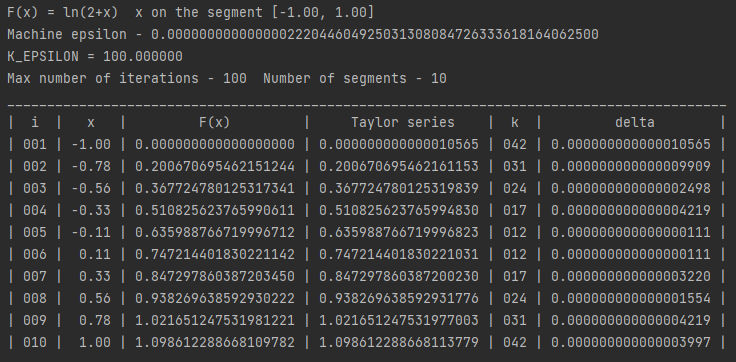


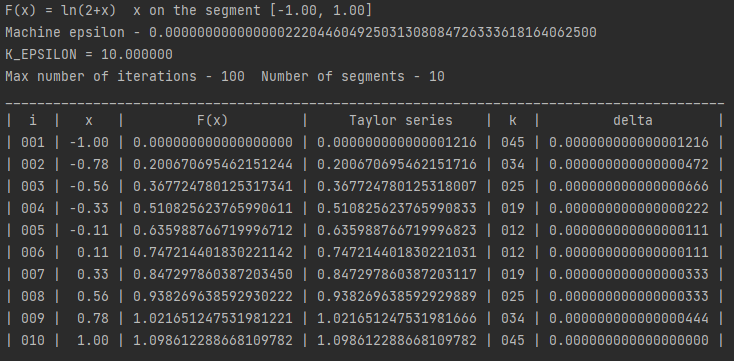
Реализация остальных функций представлена ниже:

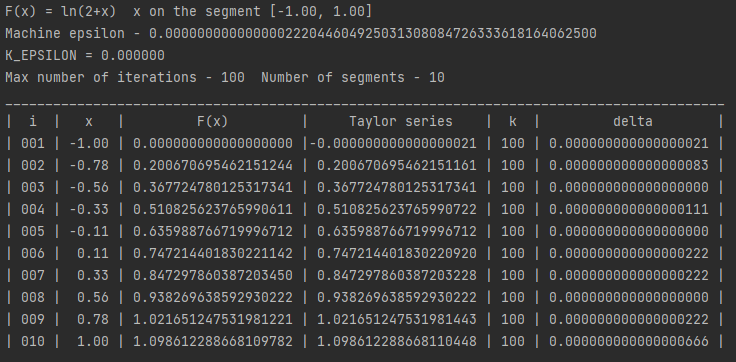


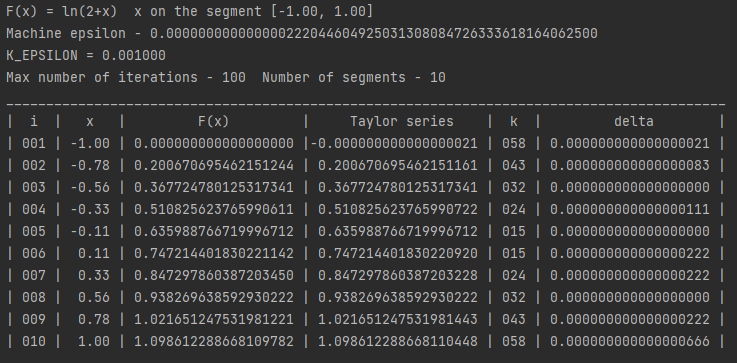
В функции get\_teilor\_value цикл идёт до параметра NUMBER\_ITERS или до того момента, пока текущее значение currentVal, вычисляемое на данной итерации, не будет находиться в окрестности eps = ɛ\*k (по условию задачи).

# Тесты









# Заключение

По приведённым выше тестам можно заметить, что коэффициент k регулирует точность вычисления по Тейлору. Но при k ≤ 1 точность начинает зависеть только от машинного эпсилона. Значение по формуле Тейлора будет отличаться от значения встроенной в Си функции в большинстве случаев. Это происходит ввиду погрешности, появляющийся из-за ограниченного диапазона представления вещественных чисел в памяти компьютера.

Формула Тейлора сводит вычисление трансцендентных функций к алгебраическим. Однако этот простой способ не применяется ввиду большой ресурсоёмкости и значительной погрешности.

При работе с вещественными числами стоит обязательно учесть особенности их представления в памяти компьютера.

# Список источников

1. IEEE 754 - стандарт двоичной арифметики  
с плавающей точкой – URL: <https://www.softelectro.ru/ieee754.html> (26.12.2021)

2. Что нужно знать про арифметику с плавающей запятой – URL: <https://habr.com/ru/post/112953/> (27.12.2021)

3. Ряд Тейлора – URL: [https://math.fandom.com/ru/wiki/Ряд\_Тейлора](https://math.fandom.com/ru/wiki/%D0%A0%D1%8F%D0%B4_%D0%A2%D0%B5%D0%B9%D0%BB%D0%BE%D1%80%D0%B0) (26.12.2021)