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»

**КУРСОВОЙ ПРОЕКТ**

По курсу «Вычислительные системы»

I семестр

Тема:

«Вещественный тип. Приближенные значения. Табулирование функций»

|  |  |
| --- | --- |
| Группа: | М8О-106Б-21 |
| Студент: | Лохматов Н. И. |
| Преподаватель: | Дубинин А.В. |
| Оценка: |  |
| Дата: |  |

Москва, 2021

Оглавление

[Введение 3](#_Toc91788178)

[Теория 4](#_Toc91788179)

[Представление вещественных чисел. IEEE-754 4](#_Toc91788180)

[Float и double 5](#_Toc91788181)

[Машинный эпсилон 6](#_Toc91788182)

[Ряды Тейлора 6](#_Toc91788183)

[Практика 7](#_Toc91788184)

[Задание 7](#_Toc91788185)

[Описание программы 7](#_Toc91788186)

[Использованные переменные 9](#_Toc91788187)

[Тесты 9](#_Toc91788188)

[Заключение 11](#_Toc91788189)

[Источники 12](#_Toc91788190)

# Введение

Числа с плавающей запятой являются очень важной составляющей программирования. Этими фундаментальными знаниями должен владеть каждый программист, ведь они сталкиваются с ними очень часто.

Так же основой в программировании лежат различные области математики, без знаний которых невозможно стать хорошим программистом.

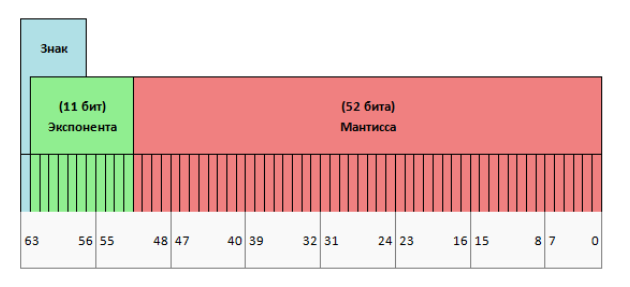
Целью работы является написание программы на Си, которая будет выводить таблицу значений элементарной функции, вычисленной двумя способами: по формуле Тейлора и с помощью встроенных функций языка программирования.

# Теория

## Представление вещественных чисел. IEEE-754

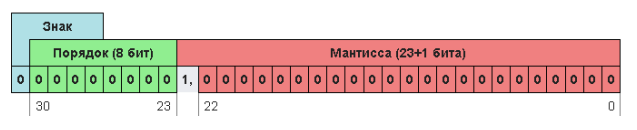
Вещественные числа обычно представляются в виде чисел с плавающей запятой. Числа с плавающей запятой — один из возможных способов представления действительных чисел, который является компромиссом между точностью и диапазоном принимаемых значений, его можно считать аналогом экспоненциальной записи чисел, но только в памяти компьютера.

Число с плавающей запятой состоит из набора отдельных двоичных разрядов, условно разделенных на так называемые знак, порядок и мантиссу. В наиболее распространённом формате (стандарт IEEE 754) число с плавающей запятой представляется в виде набора битов, часть из которых кодирует собой мантиссу числа, другая часть — показатель степени, и ещё один бит используется для указания знака числа (00 — если число положительное, 11 — если число отрицательное). При этом порядок записывается как целое число в [коде со сдвигом](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%86%D0%B5%D0%BB%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB:_%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B9_%D0%BA%D0%BE%D0%B4,_%D0%BA%D0%BE%D0%B4_%D1%81%D0%BE_%D1%81%D0%B4%D0%B2%D0%B8%D0%B3%D0%BE%D0%BC,_%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), а мантисса — в [нормализованном виде](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_%D0%B2%D0%B5%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB#.D0.9D.D0.BE.D1.80.D0.BC.D0.B0.D0.BB.D1.8C.D0.BD.D0.B0.D1.8F_.D0.B8_.D0.BD.D0.BE.D1.80.D0.BC.D0.B0.D0.BB.D0.B8.D0.B7.D0.BE.D0.B2.D0.B0.D0.BD.D0.BD.D0.B0.D1.8F_.D1.84.D0.BE.D1.80.D0.BC.D0.B0), своей дробной частью в двоичной системе счисления. Вот пример такого числа из 1616 двоичных разрядов:

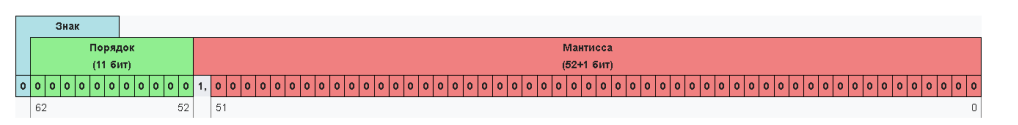


## Float и double

Число одинарной точности или float - компьютерный формат представления чисел, занимающий в памяти одно машинное слово. Используется для работы с вещественными числами везде, где не нужна очень высокая точность.



Число двойной точности или double - компьютерный формат представления чисел, занимающий в памяти два машинных слова. Часто используется благодаря своей неплохой точности, даже несмотря на двойной расход памяти и сетевого трафика относительно чисел одинарной точности.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип | Размер в байтах | Мин. положительное значение | Макс. значение | Количество знаков мантиссы |
| float | 48 | 1.175494351е-38 | 3.402823466е+38 | 7 |
| double |  | 2.2250738585072014е-308 | 1.7976931348623158е+308 | 15 |

## Машинный эпсилон

**Машинный эпсилон** — числовое значение, меньше которого невозможно задавать относительную точность для любого алгоритма, возвращающего [вещественные числа](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE). Абсолютное значение «машинного эпсилон» зависит от разрядности сетки применяемой [ЭВМ](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE-%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0), типа (разрядности) используемых при расчетах чисел, и от принятой в конкретном трансляторе структуры представления вещественных чисел (количества бит, отводимых на мантиссу и на порядок). Формально машинный эпсилон обычно определяют, как минимальное из чисел ε, для которого 1 + ε > 1 при машинных расчетах с числами данного типа. Альтернативное определение — максимальное ε, для которого справедливо равенство 1 + ε = 1.

## Ряды Тейлора

Пусть функция  имеет на отрезке [x,a] или [a,x] непрерывные производные вплоть до n-го порядка и (n+1)-ую производную на (x,a) или (a,x). Если функция непрерывна на отрезках и имеет ненулевые производные на интервале, то можно разложить в степенной ряд по формуле Тейлора:

Благодаря этому можно разложить функцию в степенные ряды, что упрощает вычисление ее значения в конкретной точке, при этом погрешность зависит от количества взятых членов и значения остаточного члена.

Остаточный член можно представить в различных формах (зависит от выбранной функции ), например, форма Лагранжа при :

# Практика

## 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 𝜀 ∗ 𝑘, где 𝜀 - 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 𝜀 и обеспечивать корректные размеры генерируемой таблицы.



## Описание программы

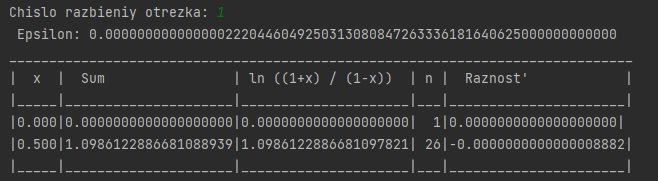
Программа состоит из главной функции main, функции, вычисляющей значение функции f(x) с помощью встроенных функций языка программирования т функции, вычисляющей машинное эпсилон.

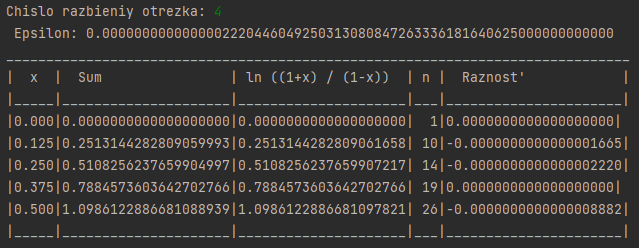
#include <stdio.h>  
#include <math.h>  
  
long double m\_eps() {  
 long double e = 1;  
 long double eps;  
 while(1 < (1 + e)){  
 eps = e;  
 e /= 2;  
 }  
 return eps;  
}  
  
long double func(long double x){  
 return log((1 + x) / (1 - x));  
}  
  
  
int main(){  
 long double n, a = 0, b = 0.5;  
 long double x = a;  
 printf("Chislo razbieniy otrezka: ");  
 scanf("%Lf", &n);  
 long double step = (b-a)/n;  
 long double eps = m\_eps();  
 printf("Epsilon: %.64Lf\n", eps);  
 printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
 printf("| x | Sum | ln ((1+x) / (1-x)) | n | Raznost' |\n");  
 printf("|\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");  
  
 for(int i = 0; i <= n; i++, x += step){  
 int n = 1;  
 long double zn\_x = x;  
 long double sum = 0;  
  
 long double k = 1.0;  
  
 while(((zn\_x > eps || zn\_x < -eps) && n < 100)) {  
  
 if (n == 1) {  
 zn\_x = 2\*zn\_x / k;  
 } else {  
 zn\_x = (k-2)\*(zn\_x\*x\*x) / k;  
 }  
  
 sum += zn\_x;  
 n++;  
 k += 2;  
 }  
  
 printf("|%.3Lf|%.19Lf|%.19Lf|%3d|%.19Lf|\n", x, sum, func(x), n, sum-func(x));  
 }  
  
 printf("|\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");  
}

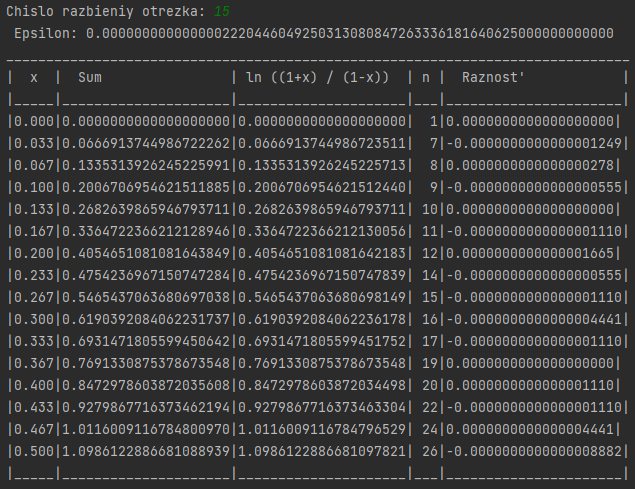
## Использованные переменные

|  |  |  |  |
| --- | --- | --- | --- |
| Имя переменной | Начальное значение | Тип переменной | Описание |
| n | - | int | Количество разбиений отрезка |
| a | 0 | long double | Начало отрезка |
| b | 0.5 | long double | Конец отрезка |
| eps | 1 | long double | Машинное эпсилон |
| zn\_x | x | long double | Слагаемое в ряде Тейлора |
| sum | 0 | long double | Значение ряда Тейлора |

## Тесты







# Заключение

В ходе работы над данным курсовым проектом я научился использовать ряды Тейлора, которые позволяют разложить сложную математическую функцию на сумму элементарных, что значительно облегчает вычисления, но увеличивает погрешность. Также я узнал о машинном эпсилоне, изучил методы работы с циклами.

Курсовой проект показал необходимость применения знаний из математического анализа при написании программы, что стало мотивацией для более тщательного изучения высшей математики.

# Источники

* Фоксфорд – стандарт IEEE-754 <https://foxford.ru/wiki/informatika/standart-ieee-754-ispolzovanie-bit-pri-hranenii-chisel-s-plavayuschey-tochkoy>
* Вещественные типы double и float <https://docs.mql4.com/ru/basis/types/double>
* Машинное эпсилон <https://it.wikireading.ru/25936>
* Битюков Ю.И.: «Лекции по математическому анализу. Курс лекций 1 семестр». Лекция 12 «Формула Тейлора», страница 50