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-22 |
| Студент: | Абдисаламов Э. |
| Преподаватель: | Дубинин А. В. |
| Оценка: |  |
| Дата: |  |

Москва, 2022

Содержание

[Введение 3](#_Toc89950956)

[1.Теоретическая часть 4](#_Toc89950957)

[1.1 Представление вещественных чисел 4](#_Toc89950958)

[1.2 Стандарт IEEE-754 5](#_Toc89950959)

[1.3 Типы float и double 5](#_Toc89950960)

[1.4 Машинное эпсилон 7](#_Toc89950961)

[1.5 Ряды Тейлора 7](#_Toc89950962)

[2. Практическая часть 9](#_Toc89950963)

[2.1 Задание 9](#_Toc89950964)

[2.1.1 Вариант 9](#_Toc89950965)

[2.2 Алгоритм решения 10](#_Toc89950966)

[2.3 Описание программы 10](#_Toc89950967)

[2.4 Использованные переменные 11](#_Toc89950968)

[2.5 Протокол 12](#_Toc89950969)

[2.6 Тесты 13](#_Toc89950970)

[Заключение 18](#_Toc89950971)

[Список использованных источников 19](#_Toc89950972)

# Введение

С каждым годом сфера программирования все больше и больше развивается, появляются новые отрасли, новые профессии, новые направления в вузах. Однако фундаментом для новых знаний служат «старые», базовые. Так, уважающий себя современный программист должен владеть знаниями о представлении чисел в памяти компьютера. Любой человек из сферы IT должен понимать, как устроены и как «работают» числа с плавающей запятой, ведь не секрет, что программисты очень часто сталкиваются с ними в процессе работы, порой в самый неожиданный момент. Поэтому мы, как будущие IT-деятели, обязаны уже сейчас познакомиться с этой темой для того, чтобы она прочно осела в нашей памяти, что мы и сделали в этом курсовом проекте.

Нельзя не упомянуть и необходимость применения знаний из смежных областей для написания данного курсового проекта. Так, нам пригодились навыки вычисления значения функции по формуле Тейлора, которые были получены на математическом анализе. Приятно видеть, как знания из такого сложного и непонятного предмета могут пригодиться там, где совсем не ожидаешь их применить.

Цель работы: необходимо написать программу на Си, которая будет печатать таблицу значений элементарной функции, вычисленной двумя способами: по формуле Тейлора и с помощью встроенных функций языка программирования

1.Теоретическая часть

## 1.1 Представление вещественных чисел

Вещественными числами в компьютерной технике называются числа, имеющие дробную часть. При их изображении во многих языках программирования вместо запятой принято ставить точку. Так, например, число 5 - целое, а числа 5.1 и 5.0 - вещественные. Для удобства отображения чисел, принимающих значения из достаточно широкого диапазона (то есть, как очень маленьких, так и очень больших), используется форма записи чисел с порядком основания системы счисления. Например, десятичное число 1.75 можно в этой форме представить так:

Любое число в системе счисления с основанием можно записать в виде , где называется мантиссой числа, а - порядком. Такой способ записи чисел называется представлением с плавающей точкой. Если «плавающая» точка расположена в мантиссе перед первой значащей цифрой, то при фиксированном количестве разрядов, отведённых под мантиссу, обеспечивается запись максимального количества значащих цифр числа, то есть максимальная точность представления числа в машине. Из этого следует, что мантисса должна быть правильной дробью, первая цифра которой отлична от нуля: M ∈ [0.1,1). Такое, наиболее выгодное для компьютера, представление вещественных чисел называется нормализованным. Мантиссу и порядок q-ичного числа принято записывать в системе с основанием q, а само основание - в десятичной системе.

## 1.2 Стандарт IEEE-754

Стандарт IEEE 754 - наиболее распространённый формат чисел с плавающей запятой: число представляется в виде набора битов, часть из которых кодирует собой мантиссу числа, другая - показатель степени, и ещё один бит используется для указания знака числа (если число положительное, то 0, если отрицательное - 1). При этом порядок записывается как целое число в коде со сдвигом, а мантисса - в нормализованном виде, своей дробной частью в двоичной системе счисления. На рисунке 1 представлен пример числа с плавающей запятой из 16 двоичных разрядов.

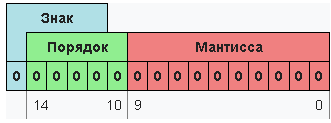


Рисунок 1. Пример числа с плавающей запятой из 16 двоичных разрядов

Так как старший двоичный разряд (целая часть) мантиссы вещественного числа в нормализованном виде всегда равен «1», то его можно не записывать, сэкономив таким образом один бит, что и используется в стандарте IEEE 754.

## 1.3 Типы float и double

Число одинарной точности (float) - компьютерный формат представления чисел, занимающий в памяти одно машинное слово. Используется для работы с вещественными числами везде, где не нужна очень высокая точность. Порядок записан со сдвигом –127 (рис.2).



Рисунок 2. Число одинарной точности

Число двойной точности (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. Порядок записан со сдвигом – 1023 (рис.3).

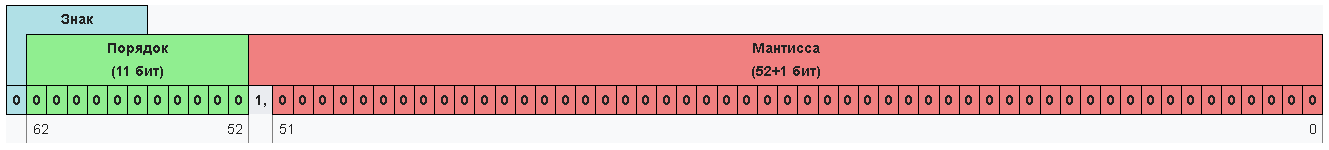


Рисунок 3. Число двойной точности

И тип float, и тип double хранят числа с плавающей запятой, разница заключается в количестве используемых байтов и, следовательно, точности представления числа. В таблице 1 представлено сравнение типов float и double.

Таблица 1.Сравнение типов float и double

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип | Точность | Размер | Количество знаков мантиссы | Максимальное положительное значение | Максимальное значение |
| float | Одинарная | 4 байта | 7 | 1.4e-45 | 3.4e38 |
| double | Двойная | 8 байт | 15 | 5.0e-324 | 1.7e308 |

## 1.4 Машинное эпсилон

Машинное эпсилон - числовое значение, меньше которого невозможно задавать относительную точность для любого алгоритма, возвращающего вещественные числа. Абсолютное значение для машинного эпсилон зависит от разрядности сетки применяемой ЭВМ и от разрядности используемых при расчёте чисел. Формально машинный эпсилон обычно определяют как минимальное из чисел , для которого при машинных расчетах с числами данного типа. Фактически, два отличных от нуля числа являются равными с точки зрения машинной арифметики, если их модуль разности меньше машинного эпсилона на отрезке от 1 до 2. В других отрезках будет погрешность в виде eps\*k = , где n – порядок числа, eps – машинный эпсилон на отрезке от 1 до 2.

В языке Си машинное эпсилон определено для следующих типов:  - машинный эпсилон для чисел типа float (32 бита),  - для типа double (64 бита) и  - для типа long double (80 бит).

## 1.5 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 :

# 2. Практическая часть

## 2.1 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 - 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.

### 2.1.1 Варианты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Ряд | a | b | Функция |
| 8 |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Ряд | a | b | Функция |
| 9 |  |  |  |  |

## 2.2 Алгоритм решения

Рассчитываем машинный эпсилон при помощи цикла, каждый раз деля переменную на 2, пока (1 + переменная) не станет равно 1. Тогда берем предыдущее значение переменной, т. к. текущая уже меньше, чем машинный эпсилон, поэтому равен нулю.

Считаем ряд Тейлора вычисляя его элементы, пока следующий элемент этого ряда нее станет меньше нашего delta (машинное эпсилон \* k). Также считаем значение функции по формуле простой постановкой. Дальше сравниваем значения по формуле и по ряду Тейлора и выводим все на терминал.

## 2.3 Описание программы

Программа состоит из стандартных подключенных библиотек stdio.h и math.h, заголовочного файла func\_kp3.h (объявление используемых функций), файла func\_for\_kp3.c (определение используемых функций), также главный файл kp\_3.c с основной функцией main().

Используемые функции:

* get\_mch\_eps – вычисляет и возвращает машинный эпсилон для типа double в языке С.
* elmnt\_tylr8(int n, double x) – вычисляет и возвращает значение n-ного элемента ряда Тейлора для x для 8 варианта.
* elmnt\_tylr9(int n, double x) – вычисляет и возвращает значение n-ного элемента ряда Тейлора для x для 9 варианта.
* print\_begin(int n) – выводит начало таблицы для n-ного варианта.
* var8(double delta) – выводит требуемую в задании таблицу для 8 варианта.
* var9(double delta) – выводит требуемую в задании таблицу для 9 варианта

Программа сперва выводит машинный эпсилон для типа double в языке С. После требуется ввести число, которое означает количество проверок на заданном отрезке для 8 варианта. Далее выводится требуемая в задании таблица для 8 варианта. После требуется ввести число, которое означает количество проверок на заданном отрезке для 9 варианта. Далее выводится требуемая в задании таблица для 9 варианта и программа завершается.

## 2.4 Использованные переменные

|  |  |  |  |
| --- | --- | --- | --- |
| Имя переменной | Начальное значение | Тип переменной | Описание |
| eps | Отсутствует | double | Машинный эпсилон для типа double |
| eps1 | 1 | double | Необходимая переменная для получения eps |
| delta | 3.552714e-15 | double | Целевая погрешность |
| a | зависит от варианта | double | Начало отрезка |
| b | зависит от варианта | double | Конец отрезка |
| step | Отсутствует | double | Изменяемость x от итераций |
| sum\_tylr | 0 | double | Значение функции по ряду Тейлора |
| forml\_value | Отсутствует | double | Значение функции по формуле |
| x | Отсутствует | double | Текущее значение аргумента |
| elmnt\_tylr | Отсутствует | double | Элемент ряда Тейлора |
| nmbr\_of\_step | Отсутствует | int | Количество итераций |
| n | зависит от варианта | int | Количество итераций, требуемая для вычисления ряда Тейлора |

## 2.5 Протокол

kp\_3.c:

#include <stdio.h>

#include "func\_kp3.h"

int main(){

    double eps = get\_mch\_eps();

    double delta = eps \* 16;

    printf("mch eps for double = %e\n",delta);

    //Для 8 варианта:

    var8(delta);

    //Для 9 варианта:

    var9(delta);

}

func\_kp3.h:

#pragma once

double get\_mch\_eps();

double elmnt\_tylr8(int n, double x);

double elmnt\_tylr9(int n, double x);

void print\_begin8();

void print\_begin9();

void var8(double eps);

void var9(double eps);

func\_for\_kp3.c:

#include <math.h>

#include <stdio.h>

double get\_mch\_eps(){

    double eps1=1, eps;

    while(1.0+eps1!=1.0){

        eps=eps1;

        eps1/=2;

    }

    return eps;

}

double elmnt\_tylr8(int n, double x){

    return -((pow(2, n-1) \* pow(x, n-1))/pow(5, n));

}

double elmnt\_tylr9(int n, double x){

    return -(1+2/(pow(3, n+1)))\*pow(x, n);

}

void print\_begin(int n){

    printf("----------------------------------%d-variant----------------------------------\n", n);

    printf("|   x   |     taylor value     |     formula value     |  difference  |  n  |\n");

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

}

void var8(double delta){

    double a=0.0, b=2.0, step, sum\_tylr, forml\_value, x, elmnt\_tylr;

    int nmbr\_of\_step, n = 1;

    printf("Enter the number of partitions for variant 8: ");

    scanf("%d", &nmbr\_of\_step);

    step=(b-a)/(nmbr\_of\_step-1);

    x=a;

    print\_begin(8);

    for(int i = 0; i < nmbr\_of\_step; i++) {

        sum\_tylr = 0;

        forml\_value = 1/(2\*x - 5);

        elmnt\_tylr = elmnt\_tylr8(n, x);

        while(fabs(elmnt\_tylr)>delta){

            sum\_tylr += elmnt\_tylr;

            n++;

            elmnt\_tylr = elmnt\_tylr8(n, x);

        }

        printf("| %.3f | %.17f | %.18f | %e | %.3d |\n", x, sum\_tylr, forml\_value, fabs(sum\_tylr - forml\_value), n);

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

        x += step;

    }

}

void var9(double delta){

    double a=0.0, b=0.5, step, sum\_tylr, forml\_value, x, elmnt\_tylr;

    int nmbr\_of\_step, n = 0;

    printf("Enter the number of partitions for variant 9: ");

    scanf("%d", &nmbr\_of\_step);

    step = (b - a)/(nmbr\_of\_step - 1);

    x = a;

    print\_begin(9);

    for(int i = 0; i < nmbr\_of\_step; i++) {

        sum\_tylr = 0;

        forml\_value = (3\*x - 5)/(x\*x - 4\*x + 3);

        elmnt\_tylr = elmnt\_tylr9(n, x);

        while(fabs(elmnt\_tylr)>delta){

            sum\_tylr += elmnt\_tylr;

            n++;

            elmnt\_tylr = elmnt\_tylr9(n, x);

        }

        printf("| %.3f | %.17f | %.18f | %e | %.3d |\n", x, sum\_tylr, forml\_value, fabs(sum\_tylr - forml\_value), n);

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

        x += step;

    }

}

## 2.6 Тесты

1. PS C: kp3> gcc -Wall -pedantic -std=c99 func\_for\_kp3.c kp\_3.c -o kp3.exe

PS C: kp3> ./kp3.exe

mch eps for double = 3.552714e-15

Enter the number of partitions for variant 8: 5

----------------------------------8-variant----------------------------------

| x | taylor value | formula value | difference | n |

|-------|----------------------|-----------------------|--------------|-----|

| 0.000 | -0.20000000000000001 | -0.200000000000000011 | 0.000000e+00 | 002 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.500 | -0.04999999999999737 | -0.250000000000000000 | 2.000000e-01 | 021 |

|-------|----------------------|-----------------------|--------------|-----|

| 1.000 | -0.00000000366503482 | -0.333333333333333315 | 3.333333e-01 | 036 |

|-------|----------------------|-----------------------|--------------|-----|

| 1.500 | -0.00000000859534520 | -0.500000000000000000 | 5.000000e-01 | 063 |

|-------|----------------------|-----------------------|--------------|-----|

| 2.000 | -0.00000098079712882 | -1.000000000000000000 | 9.999990e-01 | 143 |

|-------|----------------------|-----------------------|--------------|-----|

Enter the number of partitions for variant 9: 7

----------------------------------9-variant----------------------------------

| x | taylor value | formula value | difference | n |

|-------|----------------------|-----------------------|--------------|-----|

| 0.000 | -1.66666666666666652 | -1.666666666666666741 | 2.220446e-16 | 001 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.083 | -0.10995670995670911 | -1.776623376623376549 | 1.666667e+00 | 014 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.167 | -0.00000000001531115 | -1.905882352941176361 | 1.905882e+00 | 019 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.250 | -0.00000000000484945 | -2.060606060606060552 | 2.060606e+00 | 025 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.333 | -0.00000000000176792 | -2.250000000000000000 | 2.250000e+00 | 031 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.417 | -0.00000000000279988 | -2.488479262672810943 | 2.488479e+00 | 039 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.500 | -0.00000000000363087 | -2.799999999999999378 | 2.800000e+00 | 048 |

|-------|----------------------|-----------------------|--------------|-----|

1. PS C: kp3> ./kp3.exe

mch eps for double = 3.552714e-15

Enter the number of partitions for variant 8: 12

----------------------------------8-variant----------------------------------

| x | taylor value | formula value | difference | n |

|-------|----------------------|-----------------------|--------------|-----|

| 0.000 | -0.20000000000000001 | -0.200000000000000011 | 0.000000e+00 | 002 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.182 | -0.01568627450980358 | -0.215686274509803905 | 2.000000e-01 | 014 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.364 | -0.00000000000305179 | -0.234042553191489366 | 2.340426e-01 | 018 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.545 | -0.00000000000146843 | -0.255813953488372103 | 2.558140e-01 | 022 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.727 | -0.00000000000154286 | -0.282051282051282048 | 2.820513e-01 | 027 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.909 | -0.00000000000118487 | -0.314285714285714279 | 3.142857e-01 | 033 |

|-------|----------------------|-----------------------|--------------|-----|

| 1.091 | -0.00000000000105651 | -0.354838709677419373 | 3.548387e-01 | 040 |

|-------|----------------------|-----------------------|--------------|-----|

| 1.273 | -0.00000000000148965 | -0.407407407407407496 | 4.074074e-01 | 048 |

|-------|----------------------|-----------------------|--------------|-----|

| 1.455 | -0.00000000000420740 | -0.478260869565217517 | 4.782609e-01 | 060 |

|-------|----------------------|-----------------------|--------------|-----|

| 1.636 | -0.00000000000799088 | -0.578947368421052877 | 5.789474e-01 | 076 |

|-------|----------------------|-----------------------|--------------|-----|

| 1.818 | -0.00000000003107761 | -0.733333333333333726 | 7.333333e-01 | 101 |

|-------|----------------------|-----------------------|--------------|-----|

| 2.000 | -0.00000000020368627 | -1.000000000000000888 | 1.000000e+00 | 143 |

|-------|----------------------|-----------------------|--------------|-----|

Enter the number of partitions for variant 9: 2

----------------------------------9-variant----------------------------------

| x | taylor value | formula value | difference | n |

|-------|----------------------|-----------------------|--------------|-----|

| 0.000 | -1.66666666666666652 | -1.666666666666666741 | 2.220446e-16 | 001 |

|-------|----------------------|-----------------------|--------------|-----|

| 0.500 | -1.13333333333332598 | -2.799999999999999822 | 1.666667e+00 | 048 |

|-------|----------------------|-----------------------|--------------|-----|

1. PS C: kp3> ./kp3.exe

mch eps for double = 3.552714e-15

Enter the number of partitions for variant 8: 1

----------------------------------8-variant----------------------------------

| x | taylor value | formula value | difference | n |

|-------|----------------------|-----------------------|--------------|-----|

| 0.000 | -0.20000000000000001 | -0.200000000000000011 | 0.000000e+00 | 002 |

|-------|----------------------|-----------------------|--------------|-----|

Enter the number of partitions for variant 9: 1

----------------------------------9-variant----------------------------------

| x | taylor value | formula value | difference | n |

|-------|----------------------|-----------------------|--------------|-----|

| 0.000 | -1.66666666666666652 | -1.666666666666666741 | 2.220446e-16 | 001 |

|-------|----------------------|-----------------------|--------------|-----|

# Заключение

В ходе работы над данным курсовым проектом я научился использовать ряды Тейлора, которые (в моих вариантах) позволяют разложить простую математическую функцию на сумму элементарных, что значительно усложняет вычисления, к тому же увеличивает погрешность в значении функций. Поэтому ряды Тейлора лучше использовать для сложных функций, где сложно или невозможно посчитать по формуле, например, sh(48). Конечно, получится с некоторой погрешностью, но это лучше чем ничего. Также я закрепил знания о такой важной характеристике, как машинное эпсилон, научился выводить таблицу значений, используя только символы «|» и «\_».

# Список использованных источников

1. Кафедра алгоритмических языков ВМК МГУ// Представление чисел в ЭВМ (А.А. Вылиток)

<https://al.cs.msu.ru/static/classes/NumberRepresent.pdf>

1. Университет ИТМО// Представление вещественных чисел

<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.A2.D0.B8.D0.BF.D1.8B_.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_.28.D0.BF.D0.BE_IEEE_754.29>

1. ТГПУ им. Л. Н. Толстого// Машинный ноль

<http://poivs.tsput.ru/ru/Math/Elementary/Arithmetic/ComputerArithmetic/MachineZero>

1. Битюков Ю.И.: «Лекции по математическому анализу. Курс лекций 1 семестр». Лекция 12 «Формула Тейлора», страница 50.