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
по курсу «Вычислительные системы»
I семестр
Задание 3
«Вещественный тип. Приближенные вычисления. Табулирование функций»

Группа	М8О-109Б-22
Студент	Степанов А.Н.
Преподаватель	Сысоев М.А.
Оценка	
Дата	

Москва, 2022

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 точка включая концы отрезка), находящихся в рекомендованной области хорошей точности формулы Тейлора. Вычисления по формуле Тейлора проводить по экономной в сложностном смысле схеме с точностью $\varepsilon * 10^k$, 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.

Вариант 6:

6	$x+\frac{x^3}{3!}+\ldots+\frac{x^{2n-1}}{(2n-1)!}$	0.0	1.0	$\operatorname{sh} x$

Теоретическая часть

Формула Тейлора — формула разложения функции в бесконечную сумму степенных функций. Формула широко используется в приближённых вычислениях, так как позволяет приводить трансцендентных функций к более простым. Сама она является следствием теоремы Лагранжа о среднем значении дифференцируемой функции. В случае a=0 формула называется рядом Маклорена.

$$\sum_{n=0}^{k} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a)+f^{(1)}(a)(x-a)+\frac{f^{(2)}(a)}{2!}(x-a)^2+\ldots+\frac{f^{(k)}(a)}{k!}(x-a)^k$$

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 это машинное эпсилон определяют как число, удовлетворяющее равенству 1 + ε = 1. Фактически, два отличных от нуля числа являются равными с точки зрения машинной арифметики, если их модуль разности меньше или не превосходит машинное эпсилон.

В языке Си машинные эпсилон определено для следующих типов: float – $1.19 * 10^{-7}$, double – $2.20 * 10^{-16}$, long double – $1.08 * 10^{-19}$.

Описание алгоритма

Рассмотрим алгоритм решения. Сперва нужно найти машинное эпсилон, на котором будет основываться точность вычисления. Это можно сделать просто деля 1 на 2, пока это число не будет удовлетворять условию машинного эпсилон: 1 + ε = 1.

Проанализировав формулу Тейлора, мы можем заметить, что для каждого нового n, последующий элемент будет отличаться в (x^2) \((2n-1)*(2n-2))(из отношения между n членом и n-1 членом)=> вычисление формулы(а именно факториалы и возведения в степень) можно свести к умножению предыдущего члена на это выражение, что уменьшит время на каждую итерацию и количество нетривиальных операций. Получив новый элемент, мы складываем его с предыдущим, пока значение формулы Тейлора по Маклорену не “сравняется” с библиотечной функцией (то есть их разность не будет меньше машинного эпсилон).

Использованные в программе переменные

Название переменной	Тип переменной	Смысл переменной
epsilon	double	Машинное эпсилон
x	double	Аргумент функции
n(new_elementus)	int	Степень, до которых мы разложили ряд Тейлора (по сов-ву кол-во итераций)
a, b	double	Границы отрезка (левая и правая)
ans	double	Sh(x), вычесленная при помощи формулы Тейлора(а также сумма эл-ов ряда)
normal_function	double	Sh(x) – библиотечная
cur	double	Nый элемент ряда
last	double	Nый-1 элемент ряда
k	int	Степень, регулирующая погрешность вычисления(не больше 16, иначе уйдете в машинный epsilon и не вернетесь)
n(main)	int	Число, показывающего на сколько частей будет поделена ~~Польша~~ ~~Украина~~ отрезок
rate_error	double	ЕЕ благородие, госпожа Погрешность.
delta	double	Изменение x – значение(велечина) 1/n части ~~Польши~~ ~~Украины~~ отрезка.
num_of_iter	int	Количество итераций, превышающее настоящее число на единицу(иначе возникло бы деление на 0(~~нам же нужен кусок земель Польши?~~))

Исходный код программы:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <limits.h>
#include <assert.h>
#include <stdint.h>
double Eps(){
    double epsilon=1.0;
    while(1.0+(epsilon/2.0)!=1.0){
        epsilon/=2.0;
    }
return epsilon;
}
void TEST_EPS(){
assert(Eps()<=_DBL_EPSILON_);
}
double new_elementus(double x,int n){
return x*x/((2*n-2)*(2*n-1));
}


int main()
{   TEST_EPS();
    double epsilon=Eps();
    printf("Mashine epsilon = %.16e¥n",epsilon);
    double a=0;
    double b=1;
    double x=0;
    double ans=0;
    double normal_function=0;
    double cur=0;
    int k=0;
    int n=0;
    scanf("%d%d",&n,&k);
```

```
    assert(n!=0);
    assert(k<=16);
    assert(n<75||k!=16);
    double rate_error=epsilon*pow(10,16-k);



    double delta=(b-a)/(double)n;

printf("rate_error_max=%.16f|¥n",rate_error);
printf("_____¥n");
printf("x                |sh(x)mine          |sh(x)libs
|number_of_iteration¥n");
    for(int i=1;i<=n+1;i++){

        normal_function=sinh(x);
        ans=x;
        cur=x;
        double last=x;
        int num_of_iter=2;

        while(fabs(normal_function-ans)>rate_error){
         last=cur;
         cur=last*new_elementus(x,num_of_iter);
         ans+=cur;
         num_of_iter++;




        }
    printf("%.16f|%.16f | %.16f|%d¥n",x,ans,normal_function,num_of_iter-1);
    printf("_____¥n");

    x+=delta;
    }
    return 0;
```

} Входные данные

Единственная строка содержит два целых числа N (0≤N≤100) – число разбиений отрезка на равные части, K (0≤K≤16) — коэффициент для вычисления точности формулы Тейлора.(исключая то, что при n>=75 k>=16 – уйдет за эпсилон из-за погрешности и на 1 итерации сломается – сделан assert для защиты от этого).

Выходные данные

Программа должна вывести значение машинного эпсилон,погрешность, а затем N+1строку.

В каждой строке должно быть значение x, для которого вычисляется функция, число A_1 — значение, вычисленное с помощью формулы Тейлора, A_2 – значение, вычисленное с помощью встроенных функций языка, i – количество итерация, требуемых для вычисления. Разница в значениях между A_1 и A_2 не должна превышать rate_error.

Протокол исполнения и тесты

Тест №1

Ввод:

2 8

Вывод:

Mashine epsilon = 2.2204460492503131e-016

rate_error_max=0.0000000222044605|

---

x |sh(x)mine |sh(x)libs |number_of_iteration

0.0000000000000000|0.0000000000000000 | 0.0000000000000000|1

0.5000000000000000|0.5210953000992064 | 0.5210953054937474|4

1.0000000000000000|1.1752011934824436 | 1.1752011936438014|6

Тест №2

Ввод:

12 16

Вывод:

Mashine epsilon = 2.2204460492503131e-016

rate_error_max=0.0000000000000002|

---

x |sh(x)mine |sh(x)libs |number_of_iteration

0.0000000000000000|0.0000000000000000 | 0.0000000000000000|1

0.0833333333333333|0.0834298174459528 | 0.0834298174459528|5

0.1666666666666667|0.1674393439875159 | 0.1674393439875160|5

0.2500000000000000|0.2526123168081683 | 0.2526123168081683|6

0.3333333333333333|0.3395405572561500 | 0.3395405572561501|6

0.4166666666666666|0.4288280830938848 | 0.4288280830938848|7

0.4999999999999999|0.5210953054937473 | 0.5210953054937473|7

0.5833333333333333|0.6169833399428540 | 0.6169833399428542|8

0.6666666666666666|0.7171584610110419 | 0.7171584610110419|8

0.7500000000000000|0.8223167319358299 | 0.8223167319358300|8

0.8333333333333334|0.9331888411928733 | 0.9331888411928734|8

0.9166666666666667|1.0505451796586409 | 1.0505451796586409|9

1.0000000000000000|1.1752011936438016 | 1.1752011936438014|9

Тест №3

Ввод:

50 16

Вывод:

Mashine epsilon = 2.2204460492503131e-016

rate_error_max=0.0000000000000002|

x	sh(x)mine	sh(x)libs	number_of_iteration
0.0000000000000000	0.0000000000000000	0.0000000000000000	1
0.0200000000000000	0.0200013333600003	0.0200013333600003	4
0.0400000000000000	0.0400106675200325	0.0400106675200325	4
0.0600000000000000	0.0600360064805554	0.0600360064805555	4
0.0800000000000000	0.0800853606441614	0.0800853606441614	5
0.1000000000000000	0.1001667500198440	0.1001667500198440	5
0.1200000000000000	0.1202882074311091	0.1202882074311091	5
0.1400000000000000	0.1404577817292107	0.1404577817292108	5
0.1600000000000000	0.1606835410127994	0.1606835410127994	5
0.1800000000000000	0.1809735758552689	0.1809735758552691	5
0.2000000000000000	0.2013360025410940	0.2013360025410940	6
0.2200000000000000	0.2217789663124512	0.2217789663124511	6
0.2400000000000000	0.2423106446274256	0.2423106446274256	6
0.2600000000000000	0.2629392504311027	0.2629392504311027	6
0.2800000000000000	0.2836730354408557	0.2836730354408557	6
0.3000000000000000	0.3045202934471425	0.3045202934471426	6
0.3200000000000000	0.3254893636311330	0.3254893636311331	6

0.3400000000000000|0.3465886339004920 | 0.3465886339004921|6

0.3600000000000000|0.3678265442446547 | 0.3678265442446547|7

0.3800000000000001|0.3892115901109344 | 0.3892115901109344|7

0.4000000000000001|0.4107523258028156 | 0.4107523258028156|7

0.4200000000000001|0.4324573679017887 | 0.4324573679017886|7

0.4400000000000001|0.4543353987140975 | 0.4543353987140975|7

0.4600000000000001|0.4763951697437780 | 0.4763951697437781|7

0.4800000000000002|0.4986455051933764 | 0.4986455051933764|7

0.5000000000000001|0.5210953054937475 | 0.5210953054937475|7

0.5200000000000001|0.5437535508643463 | 0.5437535508643462|7

0.5400000000000002|0.5666293049054345 | 0.5666293049054345|7

0.5600000000000002|0.5897317182236431 | 0.5897317182236432|7

0.5800000000000002|0.6130700320923357 | 0.6130700320923359|7

0.6000000000000002|0.6366535821482414 | 0.6366535821482415|8

0.6200000000000002|0.6604918021258340 | 0.6604918021258340|8

0.6400000000000002|0.6845942276309518 | 0.6845942276309517|8

0.6600000000000003|0.7089704999551666 | 0.7089704999551665|8

0.6800000000000003|0.7336303699324294 | 0.7336303699324294|8

0.7000000000000003|0.7585837018395339 | 0.7585837018395338|8

0.7200000000000003|0.7838404773419585 | 0.7838404773419584|8

0.7400000000000003|0.8094107994866656 | 0.8094107994866655|8

0.7600000000000003|0.8353048967434550 | 0.8353048967434551|8

0.7800000000000004|0.8615331270964893 | 0.8615331270964892|8

0.8000000000000004|0.8881059821876235 | 0.8881059821876235|8

0.8200000000000004|0.9150340915132036 | 0.9150340915132038|8

0.8400000000000004|0.9423282266760064 | 0.9423282266760066|8

0.8600000000000004|0.9699993056940235 | 0.9699993056940237|8

0.8800000000000005|0.9980583973678148 | 0.9980583973678149|9

0.9000000000000005|1.0265167257081758 | 1.0265167257081760|8

0.9200000000000005|1.0553856744258927 | 1.0553856744258925|9

0.9400000000000005|1.0846767914853757 | 1.0846767914853759|9

0.9600000000000005|1.1144017937240034 | 1.1144017937240036|9

0.9800000000000005|1.1445725715390096 | 1.1445725715390096|9

1.0000000000000004|1.1752011936438023 | 1.1752011936438020|9

Тест №4

Ввод:

100 15

Вывод:

Mashine epsilon = 2.2204460492503131e-016

100 15

rate_error_max=0.0000000000000022|

x	|sh(x)mine	|sh(x)libs	|number_of_iteration
0.0000000000000000	0.0000000000000000	0.0000000000000000	1
0.0100000000000000	0.0100001666675000	0.0100001666675000	3
0.0200000000000000	0.0200013333600000	0.0200013333600003	3
0.0300000000000000	0.0300045002025043	0.0300045002025043	4
0.0400000000000000	0.0400106675200325	0.0400106675200325	4
0.0500000000000000	0.0500208359376550	0.0500208359376550	4
0.0600000000000000	0.0600360064805554	0.0600360064805555	4
0.0700000000000000	0.0700571806741340	0.0700571806741341	4
0.0800000000000000	0.0800853606441610	0.0800853606441614	4
0.0900000000000000	0.0901215492169900	0.0901215492169911	4
0.1000000000000000	0.1001667500198440	0.1001667500198440	5
0.1100000000000000	0.1102219675811715	0.1102219675811715	5
0.1200000000000000	0.1202882074311091	0.1202882074311091	5
0.1300000000000000	0.1303664762020302	0.1303664762020302	5
0.1400000000000000	0.1404577817292107	0.1404577817292107	5
0.1500000000000000	0.1505631331516126	0.1505631331516127	5
0.1600000000000000	0.1606835410127994	0.1606835410127994	5
0.1700000000000000	0.1708200173619908	0.1708200173619909	5
0.1800000000000000	0.1809735758552689	0.1809735758552691	5
0.1900000000000000	0.1911452318569443	0.1911452318569446	5
0.2000000000000000	0.2013360025410935	0.2013360025410940	5

0.2100000000000001|0.2115469069932773 | 0.2115469069932781|5

0.2200000000000001|0.2217789663124498 | 0.2217789663124512|5

0.2300000000000001|0.2320332037130720 | 0.2320332037130720|6

0.2400000000000001|0.2423106446274257 | 0.2423106446274257|6

0.2500000000000001|0.2526123168081684 | 0.2526123168081684|6

0.2600000000000001|0.2629392504311028 | 0.2629392504311028|6

0.2700000000000001|0.2732924781981971 | 0.2732924781981972|6

0.2800000000000001|0.2836730354408558 | 0.2836730354408558|6

0.2900000000000001|0.2940819602234536 | 0.2940819602234536|6

0.3000000000000001|0.3045202934471427 | 0.3045202934471427|6

0.3100000000000001|0.3149890789539444 | 0.3149890789539444|6

0.3200000000000001|0.3254893636311331 | 0.3254893636311332|6

0.3300000000000001|0.3360221975159271 | 0.3360221975159272|6

0.3400000000000001|0.3465886339004921 | 0.3465886339004922|6

0.3500000000000001|0.3571897294372719 | 0.3571897294372721|6

0.3600000000000002|0.3678265442446545 | 0.3678265442446548|6

0.3700000000000002|0.3785001420129847 | 0.3785001420129851|6

0.3800000000000002|0.3892115901109340 | 0.3892115901109345|6

0.3900000000000002|0.3999619596922384 | 0.3999619596922391|6

0.4000000000000002|0.4107523258028146 | 0.4107523258028157|6

0.4100000000000002|0.4215837674882658 | 0.4215837674882673|6

0.4200000000000002|0.4324573679017867 | 0.4324573679017887|6

0.4300000000000002|0.4433742144124827 | 0.4433742144124827|7

0.4400000000000002|0.4543353987140976 | 0.4543353987140976|7

0.4500000000000002|0.4653420169341981 | 0.4653420169341980|7

0.4600000000000002|0.4763951697437781 | 0.4763951697437782|7

0.4700000000000003|0.4874959624673300 | 0.4874959624673300|7

0.4800000000000003|0.4986455051933765 | 0.4986455051933766|7

0.4900000000000003|0.5098449128854817 | 0.5098449128854817|7

0.5000000000000002|0.5210953054937476 | 0.5210953054937476|7

0.5100000000000002|0.5323978080668101 | 0.5323978080668105|7

0.5200000000000002|0.5437535508643464 | 0.5437535508643463|7

0.5300000000000003|0.5551636694700981 | 0.5551636694700980|7

0.5400000000000003|0.5666293049054346 | 0.5666293049054347|7

0.5500000000000003|0.5781516037434545 | 0.5781516037434545|7

0.5600000000000003|0.5897317182236432 | 0.5897317182236435|7

0.5700000000000003|0.6013708063670992 | 0.6013708063670994|7

0.5800000000000003|0.6130700320923358 | 0.6130700320923360|7

0.5900000000000003|0.6248305653316750 | 0.6248305653316754|7

0.6000000000000003|0.6366535821482412 | 0.6366535821482416|7

0.6100000000000003|0.6485402648535688 | 0.6485402648535692|7

0.6200000000000003|0.6604918021258336 | 0.6604918021258341|7

0.6300000000000003|0.6725093891287226 | 0.6725093891287234|7

0.6400000000000004|0.6845942276309510 | 0.6845942276309518|7

0.6500000000000004|0.6967475261264392 | 0.6967475261264404|7

0.6600000000000004|0.7089704999551652 | 0.7089704999551666|7

0.6700000000000004|0.7212643714246970 | 0.7212643714246990|7

0.6800000000000004|0.7336303699324295 | 0.7336303699324295|8

0.6900000000000004|0.7460697320885138 | 0.7460697320885139|8

0.7000000000000004|0.7585837018395342 | 0.7585837018395341|8

0.7100000000000004|0.7711735305928933 | 0.7711735305928933|8

0.7200000000000004|0.7838404773419587 | 0.7838404773419586|8

0.7300000000000004|0.7965858087919606 | 0.7965858087919606|8

0.7400000000000004|0.8094107994866657 | 0.8094107994866656|8

0.7500000000000004|0.8223167319358304 | 0.8223167319358306|8

0.7600000000000005|0.8353048967434551 | 0.8353048967434553|8

0.7700000000000005|0.8483765927368442 | 0.8483765927368441|8

0.7800000000000005|0.8615331270964894 | 0.8615331270964894|8

0.7900000000000005|0.8747758154867912 | 0.8747758154867911|8

0.8000000000000005|0.8881059821876236 | 0.8881059821876236|8

0.8100000000000005|0.9015249602267660 | 0.9015249602267658|8

0.8200000000000005|0.9150340915132037 | 0.9150340915132039|8

0.8300000000000005|0.9286347269713242 | 0.9286347269713243|8

0.8400000000000005|0.9423282266760066 | 0.9423282266760067|8

0.8500000000000005|0.9561159599886326 | 0.9561159599886329|8

0.8600000000000005|0.9699993056940236 | 0.9699993056940238|8

0.8700000000000006|0.9839796521383194 | 0.9839796521383196|8

0.8800000000000006|0.9980583973678147 | 0.9980583973678150|8

0.8900000000000006|1.0122369492687653 | 1.0122369492687653|8

0.9000000000000006|1.0265167257081758 | 1.0265167257081762|8

0.9100000000000006|1.0408991546755906 | 1.0408991546755910|8

0.9200000000000006|1.0553856744258920 | 1.0553856744258927|8

0.9300000000000006|1.0699777336231269 | 1.0699777336231278|8

0.9400000000000006|1.0846767914853750 | 1.0846767914853761|8

0.9500000000000006|1.0994843179306724 | 1.0994843179306735|8

0.9600000000000006|1.1144017937240023 | 1.1144017937240038|8

0.9700000000000006|1.1294307106253763 | 1.1294307106253778|8

0.9800000000000007|1.1445725715390078 | 1.1445725715390098|8

0.9900000000000007|1.1598288906636092 | 1.1598288906636092|9

1.0000000000000007|1.1752011936438025 | 1.1752011936438025|9

Вывод

В работе описано определение машинного эпсилон, приведены его значения для разных переменных языка Си, описана формула Тейлора и составлен алгоритм реализации вычисления значения функции(sh(x)) с заданной точностью для заданного числа точек на отрезке. На основе алгоритма составлена программа на языке Си, проведено её тестирование на различных тестах, составлен протокол исполнения программы. В целом, работа понравилась. Приятно применять знания из других областей(матана) для решения какой-либо задачи по программированию.

Список литературы

1. Машинный ноль – URL: https://ru.wikipedia.org/wiki/Машинный_ноль
2. Ряд Тейлора – URL: https://ru.wikipedia.org/wiki/Ряд_Тейлора