МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Компьютерные науки и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Курсовая работа по
курсу
«Фундаментальная информатика»
I семестр
Задание 3
«Вещественный тип. Приближенные вычисления. Табулирование функций»

Группа	М8О-109Б-22
Студент	Недосекин А.А.
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 8:**

Ряд Тэйлора:

$$-\frac{1}{5}-\frac{2x}{5^2}-\frac{4x^2}{5^3}-\ldots-\frac{2^{n-1}x^{n-1}}{5^n}$$

Функция:

$$\frac{1}{2x-5}$$

Значения a и b: 0.0 и 2.0

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 $1 + \varepsilon = 1$. 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.

Для каждой N+1 строки нужно просуммировать i членов формулы Тейлора, пока $|A_1-A_2|> \varepsilon$. Для этого просто ищем каждый новый член из формулы Тэйлора и суммируем с результатом

Использованные в программе переменные

Название переменной	Тип переменной	Смысл переменной
n	int	То самое число N, на которое нужно разбить отрезок
LDBL_EPSILON	Long double	машинное эпсилон 1.0842e-19
step	Long double	Разница между текущим и предыдущем значениями переменной
x	Long double	Переменная, для которой производятся вычисления
taylor(int n, long double x)	Long double	Значение ряда Тейлора для функции
function(long double x)	Long double	Значение функции

i	int	Счетчик числа итераций

Исходный код программы:

Входные данные

Единственная строка содержит одно целое число N (0≤N≤100) – число разбиений отрезка на равные части.

Выходные данные

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

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, и Δ – разница значений A_1 и A_2 по модулю. A_1, A_2 и Δ должны быть выведены с точностью K знаков после запятой.

Протокол исполнения

```
#include <stdio.h>
#include <math.h>
#include <float.h>

long double taylor(int n, long double x){
    long double result = 0;
    for (int i = 0; i <= n; ++i){
        result -= powl(2, (n - 1)) * powl(x, (n - 1)) / powl(5, n);
    }
    return result;
}

long double function(long double x){
    return (1 / (2*x - 5));
}

int main(){
    long double a = 0.0;
    long double b = 2.0;

    int n = 100;

    printf("Machine epsilon is equal to: %Lg\n\n",
LDBL_EPSILON);
```

```
   printf("        Table of values of Taylor series and standard function\n");
printf("_____\n");
   printf("|   x   |  sum of Taylor series  |  f(x) function value  |number of iterations|\n");

printf("_____\n");

   long double x = 0;
   long double step = (b - a) / n;
   long double func = 1;
   int i = 0;
   while (fabsl(func) > LDBL_EPSILON && (i < 100) && (i < n)){
      i += 1;
      x += step;
       func = function(x);

       printf("| %.3Lf | %.20Lf | %.19Lf |         %d          |\n", x, taylor(i, x), func, i);
   }

printf("_____\n");

   return 0;
}
```

RESULT

Machine epsilon is equal to: 1.0842e-19

Table of values of Taylor series and standard function

x	sum of Taylor series	f(x) function value	number of iterations
0.020	-0.40000000000000001	-0.20161290322580645	1
0.040	-0.00960000000000000	-0.20325203252032520	2
0.060	-0.00046080000000000	-0.20491803278688525	3
0.080	-0.00003276800000000	-0.20661157024793388	4
0.100	-0.00000307200000000	-0.20833333333333333	5
0.120	-0.00000035672555520000	-0.21008403361344538	6
0.140	-0.00000004934556712960	-0.21186440677966102	7
0.160	-0.00000000791648371999	-0.21367521367521368	8
0.180	-0.00000000144440827262	-0.21551724137931035	9
0.200	-0.00000000029527900160	-0.21739130434782609	10
0.220	-0.00000000006684023424	-0.21929824561403509	11
0.240	-0.00000000001659422259	-0.22123893805309735	12
0.260	-0.00000000000448289021	-0.22321428571428571	13
0.280	-0.00000000000130904793	-0.22522522522522523	14
0.300	-0.00000000000041085391	-0.22727272727272727	15
0.320	-0.00000000000013792039	-0.22935779816513761	16
0.340	-0.00000000000004930887	-0.23148148148148148	17
0.360	-0.00000000000001870449	-0.23364485981308411	18
0.380	-0.00000000000000750317	-0.23584905660377358	19
0.400	-0.00000000000000317343	-0.23809523809523810	20

0.420	-0.00000000000000141136	-0.2403846153846153846	21
0.440	-0.00000000000000065845	-0.2427184466019417476	22
0.460	-0.00000000000000032154	-0.2450980392156862745	23
0.480	-0.00000000000000016402	-0.2475247524752475247	24
0.500	-0.00000000000000008724	-0.2500000000000000000	25
0.520	-0.00000000000000004830	-0.2525252525252525253	26
0.540	-0.00000000000000002780	-0.2551020408163265306	27
0.560	-0.00000000000000001660	-0.2577319587628865979	28
0.580	-0.00000000000000001028	-0.2604166666666666667	29
0.600	-0.00000000000000000658	-0.2631578947368421053	30
0.620	-0.00000000000000000436	-0.2659574468085106383	31
0.640	-0.00000000000000000299	-0.2688172043010752688	32
0.660	-0.00000000000000000211	-0.2717391304347826087	33
0.680	-0.00000000000000000153	-0.2747252747252747253	34
0.700	-0.00000000000000000115	-0.2777777777777777778	35
0.720	-0.00000000000000000089	-0.2808988764044943820	36
0.740	-0.00000000000000000070	-0.2840909090909090909	37
0.760	-0.00000000000000000057	-0.2873563218390804597	38
0.780	-0.00000000000000000048	-0.2906976744186046511	39
0.800	-0.00000000000000000041	-0.2941176470588235294	40

0.820	-0.00000000000000000036	-0.2976190476190476190	41
0.840	-0.00000000000000000033	-0.3012048192771084337	42
0.860	-0.00000000000000000030	-0.3048780487804878048	43
0.880	-0.00000000000000000029	-0.3086419753086419753	44
0.900	-0.00000000000000000028	-0.3124999999999999999	45
0.920	-0.00000000000000000027	-0.3164556962025316455	46
0.940	-0.00000000000000000028	-0.3205128205128205127	47
0.960	-0.00000000000000000028	-0.3246753246753246752	48
0.980	-0.00000000000000000030	-0.3289473684210526315	49
1.000	-0.00000000000000000032	-0.3333333333333333332	50
1.020	-0.00000000000000000035	-0.3378378378378378378	51
1.040	-0.00000000000000000040	-0.3424657534246575342	52
1.060	-0.00000000000000000045	-0.3472222222222222221	53
1.080	-0.00000000000000000053	-0.3521126760563380281	54
1.100	-0.00000000000000000062	-0.3571428571428571428	55
1.120	-0.00000000000000000075	-0.3623188405797101448	56
1.140	-0.00000000000000000093	-0.3676470588235294116	57
1.160	-0.00000000000000000116	-0.3731343283582089551	58
1.180	-0.00000000000000000147	-0.3787878787878787877	59
1.200	-0.00000000000000000190	-0.3846153846153846152	60

1.220	-0.00000000000000000250	-0.3906249999999999998	61
1.240	-0.00000000000000000335	-0.3968253968253968252	62
1.260	-0.00000000000000000455	-0.4032258064516129030	63
1.280	-0.00000000000000000628	-0.4098360655737704916	64
1.300	-0.00000000000000000881	-0.4166666666666666665	65
1.320	-0.00000000000000001254	-0.4237288135593220337	66
1.340	-0.00000000000000001813	-0.4310344827586206894	67
1.360	-0.00000000000000002661	-0.4385964912280701752	68
1.380	-0.00000000000000003963	-0.4464285714285714283	69
1.400	-0.00000000000000005988	-0.4545454545454545452	70
1.420	-0.00000000000000009178	-0.4629629629629629627	71
1.440	-0.00000000000000014268	-0.4716981132075471695	72
1.460	-0.00000000000000022490	-0.4807692307692307689	73
1.480	-0.00000000000000035940	-0.4901960784313725487	74
1.500	-0.00000000000000058215	-0.4999999999999999996	75
1.520	-0.00000000000000095563	-0.5102040816326530608	76
1.540	-0.00000000000000158949	-0.5208333333333333329	77
1.560	-0.00000000000000267837	-0.5319148936170212761	78
1.580	-0.00000000000000457143	-0.5434782608695652169	79
1.600	-0.00000000000000790193	-0.5555555555555555550	80

1.620	-0.00000000000001383070	-0.5681818181818181813	81
1.640	-0.00000000000002450863	-0.5813953488372093017	82
1.660	-0.00000000000004396345	-0.5952380952380952375	83
1.680	-0.00000000000007981759	-0.6097560975609756091	84
1.700	-0.00000000000014664833	-0.6249999999999999993	85
1.720	-0.00000000000027262551	-0.6410256410256410249	86
1.740	-0.00000000000051275190	-0.6578947368421052623	87
1.760	-0.00000000000097553161	-0.6756756756756756748	88
1.780	-0.00000000000187720605	-0.6944444444444444435	89
1.800	-0.00000000000365311690	-0.7142857142857142847	90
1.820	-0.00000000000718856098	-0.7352941176470588224	91
1.840	-0.00000000001430191027	-0.7575757575757575746	92
1.860	-0.00000000002876523454	-0.7812499999999999987	93
1.880	-0.00000000005848072876	-0.8064516129032258050	94
1.900	-0.00000000012016525243	-0.8333333333333333318	95
1.920	-0.00000000024952705039	-0.8620689655172413776	96
1.940	-0.00000000052357790634	-0.8928571428571428553	97
1.960	-0.00000000111000074597	-0.9259259259259259240	98
1.980	-0.00000000237737554738	-0.9615384615384615363	99
2.000	-0.00000000514351584024	-0.9999999999999999976	100

...Program finished with exit code 0
Press ENTER to exit console.

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 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 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