Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8

Кафедра №806

**Курсовой проект**

**по учебной дисциплине “Вычислительные системы”**

**I семестр**

**Задание 3.**

ФИО: Манташев А. У.

Группа: М8О-105Б-20

№ по списку: 17

Преподаватель: доц. каф. 806 Никулин Сергей Петрович

Оценка: Подпись преподавателя:

Дата сдачи: 12.12.2020

**Задание**

Составить программу на языке 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 [а,b] на n частей, находящихся в рекомендованной области хорошей точности формулы Тейлора. Вычисления по формуле Тейлора проводить по экономной в сложностном смысле схеме с точностью ε\*k, где ε — машинное эпсилон типа doub1e для данной ЭВМ, k — экспериментально подбираемый коэффициент, обеспечивающий приемлемую сходимость. Число итераций должно ограничиваться сверху числом порядка 100. Программа должна сама

определять машинное ε.





**Общий метод решения**

Рассмотрим элемент ряда Тейлора как дробь:

=

Где T – элемент ряда Тейлора, U и D – составные части дроби, i – номер элемента в ряде Тейлора.

Проанализируем X и Y в зависимости от хода i:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | … | n |
| U | 1 | x\*ln3 | \*3 | \*3 | … | \*3 |
| D | 1 | 1! | 2! | 3! | … | n! |

Очевидно, что

=\*x\*ln3, =1

=\*(i+1), =1

Из чего следует допустимость элементарной оптимизации алгоритма: используем две переменные для U и D, на каждом этапе высчитываем новый элемент ряда по указанной выше рекуррентной формуле.

Чтобы достичь максимальной точности, будем продолжать исчисления до выполнения:

V (i < 100)

Для вычисления ряда Тейлора нам потребуется машинное эпсилон, которое мы будет находить с помощью отдельной функции.

**Описание программы**

Программа работы:

-Определяем стандартные функции языка си подключая заголовки stdio.h и math.h

-Определяем функцию вычисления машинного эпсилон

-Определяем функцию для вычисления ряда Тейлора

-Вычисляем машинное эпсилон и выводим.

-Печатаем таблицу аргументов функции, значений полученных средствами языка си и ряда тейлора, количество итераций запрошенное машиной для вычисления значения функции. В процессе построчного вывода таблицы вызываем подфункции для вычисления значений.

**Описание функций программы:**

|  |  |  |
| --- | --- | --- |
| Название функции | Аргументы и их тип | Описание функции |
| double epsil() | Отсутствуют | Функция вычисляет машинный эпсилон путём цикличного деления на два и проверки на различаемость полученного числа машиной, после чего возвращает  найденное значение. |
| double fonk() | double x,  double eps,  int \* i | Функция вычисления значения члена ряда Тейлора по заданному х. i является указателем на числовую переменную куда будет записано потребовавшееся количество итераций. |
| int main() | Отсутствуют | Основа программы. Сначала вычисляется и  выводится машинное  эпсилон, затем  выводится заголовок таблицы, затем циклом по всему отрезку [a;bl с константным шагом, вычисляя значение функции с помощью ряда Тейлора для каждого значения отрезка и стандартных функций языка си, выводим в таблице: сам аргумент, значение функции, значение суммы ряда Тейлора, число  итераций, потребовавшееся для вычисления значения функции |

**Описание переменных:**

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| i | int | Количество итераций |
| x | double | Значение аргумента |
| res | double | Значение функции, полученное через ряды Тейлора |
| n | int | Количество делений отрезка [a,b] |
| eps | double | Машинный эпсилон |
| up | double | Числитель элемента ряда Тейлора |
| down | double | Знаменатель элемента ряда Тейлора |

Входные данные: число разбиений .

Выходные данные: Машинный эпсилон, таблица.

**Таблица значений.**

Машинный эпсилон: 0.00000000000000022204

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Частная сумма ряда для** | **Точное значение** | **Погрешность** | **Число итераций** |

|0.000000|1.00000000000000000000|1.00000000000000000000|0.00000000000000000000|2|

|0.050000|1.05646730854953752043|1.05646730854953796452|0.00000000000000044409|10|

|0.100000|1.11612317403390459702|1.11612317403390437498|0.00000000000000022204|11|

|0.150000|1.17914764568136631873|1.17914764568136654077|0.00000000000000022204|12|

|0.200000|1.24573093961551761844|1.24573093961551739639|0.00000000000000022204|13|

|0.250000|1.31607401295249260009|1.31607401295249237805|0.00000000000000022204|14|

|0.300000|1.39038917031590925077|1.39038917031590947282|0.00000000000000022204|14|

|0.350000|1.46890070460007371445|1.46890070460007371445|0.00000000000000000000|15|

|0.400000|1.55184557391536004367|1.55184557391535982163|0.00000000000000022204|16|

|0.450000|1.63947411675887333438|1.63947411675887289029|0.00000000000000044409|16|

|0.500000|1.73205080756887741522|1.73205080756887719318|0.00000000000000022204|16|

|0.550000|1.82985505494334543819|1.82985505494334543819|0.00000000000000000000|17|

|0.600000|1.93318204493176293290|1.93318204493176293290|0.00000000000000000000|17|

|0.650000|2.04234363194535140096|2.04234363194535140096|0.00000000000000000000|18|

|0.700000|2.15766927997459401212|2.15766927997459312394|0.00000000000000088818|18|

|0.750000|2.27950705695477706669|2.27950705695477751078|0.00000000000000044409|18|

|0.800000|2.40822468528069189020|2.40822468528069233429|0.00000000000000044409|19|

|0.850000|2.54421065164105097978|2.54421065164105097978|0.00000000000000000000|19|

|0.900000|2.68787537952228650084|2.68787537952228650084|0.00000000000000000000|20|

|0.950000|2.83965246792047754099|2.83965246792047798507|0.00000000000000044409|20|

|1.000000|3.00000000000000000000|3.00000000000000000000|0.00000000000000000000|20|

**Протокол.**

gladiator@asadec:~/Downloads$ cat kp3.c

#include <stdio.h>

#include <math.h>

double epsil(){

double eps=1.0;

while(1.0+eps/2.0 > 1.0)

eps/=2.0;

return eps;

}

double fonk(double x, double eps, int \* i){

double res=1.0,up=1.0,down=1.0;

\*i=1;

int k=13;

while (up/down>eps\*k){

if (\*i==100){

printf("lteration = 100, diff = %lf\n", fabs(pow(3.0,x) - res));

break;

}

down\*=\*i;

up\*=x\*log(3);

res+=up/down;

(\*i)++;

}

return res;

}

int main(){

double eps = epsil();

printf("EPSILON=%.20lf\n",eps);

double res,x,a=0.0,b=1.0;

int j,\*i,it=0,n;

printf("Введите число разбиения отрезка:\n");

scanf("%d\n",&n);

double t=(b-a)/n;

i=&it;

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

printf("| x | Частная сумма ряда | 3^x | Погрешность | i |\n");

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

for (j=0;j<=n;j++){

x=j\*t;

res=fonk(x,eps,i);

printf("|%lf|%.18f|%.18f|%.18f|%3d|\n",x,res,pow(3.0,x),fabs(pow(3.0,x)-res),\*i);

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

}

return 0;

}

gladiator@asadec:~/Downloads$ gcc kp3.c -lm -Wall --std=c99 -pedantic -o kp3.out && ./kp3.out

EPSILON=0.00000000000000022204

Введите число разбиения отрезка:

25

----------------------------------------------------------------------------------------------------------

| x | Частная сумма ряда | 3^x | Погрешность | i |

----------------------------------------------------------------------------------------------------------

|0.000000|1.000000000000000000|1.000000000000000000|0.000000000000000000| 2|

----------------------------------------------------------------------------------------------------------

|0.040000|1.044924351144087993|1.044924351144087549|0.000000000000000444| 9|

----------------------------------------------------------------------------------------------------------

|0.080000|1.091866899613892716|1.091866899613892494|0.000000000000000222| 10|

----------------------------------------------------------------------------------------------------------

|0.120000|1.140918311614752989|1.140918311614753211|0.000000000000000222| 11|

----------------------------------------------------------------------------------------------------------

|0.160000|1.192173326472454109|1.192173326472453887|0.000000000000000222| 12|

----------------------------------------------------------------------------------------------------------

|0.200000|1.245730939615517618|1.245730939615517396|0.000000000000000222| 12|

----------------------------------------------------------------------------------------------------------

|0.240000|1.301694593777858788|1.301694593777859010|0.000000000000000222| 13|

----------------------------------------------------------------------------------------------------------

|0.280000|1.360172378791096204|1.360172378791095982|0.000000000000000222| 13|

----------------------------------------------------------------------------------------------------------

|0.320000|1.421277240352396110|1.421277240352396110|0.000000000000000000| 14|

----------------------------------------------------------------------------------------------------------

|0.360000|1.485127198171086738|1.485127198171086738|0.000000000000000000| 14|

----------------------------------------------------------------------------------------------------------

|0.400000|1.551845573915360044|1.551845573915359822|0.000000000000000222| 15|

----------------------------------------------------------------------------------------------------------

|0.440000|1.621561229399331561|1.621561229399331339|0.000000000000000222| 15|

----------------------------------------------------------------------------------------------------------

|0.480000|1.694408815470505481|1.694408815470505258|0.000000000000000222| 15|

----------------------------------------------------------------------------------------------------------

|0.520000|1.770529032078339782|1.770529032078339782|0.000000000000000000| 16|

----------------------------------------------------------------------------------------------------------

|0.560000|1.850068900026228258|1.850068900026228702|0.000000000000000444| 16|

----------------------------------------------------------------------------------------------------------

|0.600000|1.933182044931762711|1.933182044931762711|0.000000000000000000| 16|

----------------------------------------------------------------------------------------------------------

|0.640000|2.020028993943722817|2.020028993943722373|0.000000000000000444| 17|

----------------------------------------------------------------------------------------------------------

|0.680000|2.110777485788887908|2.110777485788888352|0.000000000000000444| 17|

----------------------------------------------------------------------------------------------------------

|0.720000|2.205602794747502760|2.205602794747502760|0.000000000000000000| 17|

----------------------------------------------------------------------------------------------------------

|0.760000|2.304688069183120191|2.304688069183120191|0.000000000000000000| 17|

----------------------------------------------------------------------------------------------------------

|0.800000|2.408224685280691890|2.408224685280692334|0.000000000000000444| 18|

----------------------------------------------------------------------------------------------------------

|0.840000|2.516412616676102054|2.516412616676101610|0.000000000000000444| 18|

----------------------------------------------------------------------------------------------------------

|0.880000|2.629460820691071454|2.629460820691071010|0.000000000000000444| 18|

----------------------------------------------------------------------------------------------------------

|0.920000|2.747587641919417845|2.747587641919417401|0.000000000000000444| 19|

----------------------------------------------------------------------------------------------------------

|0.960000|2.871021233944160755|2.871021233944160755|0.000000000000000000| 19|

----------------------------------------------------------------------------------------------------------

|1.000000|3.000000000000000000|3.000000000000000000|0.000000000000000000| 19|

----------------------------------------------------------------------------------------------------------

gladiator@asadec:~/Downloads$ gcc kp3.c -lm -Wall --std=c99 -pedantic -o kp3.out && ./kp3.out

EPSILON=0.00000000000000022204

Введите число разбиения отрезка:

20

----------------------------------------------------------------------------------------------------------

| x | Частная сумма ряда | 3^x | Погрешность | i |

----------------------------------------------------------------------------------------------------------

|0.000000|1.000000000000000000|1.000000000000000000|0.000000000000000000| 2|

----------------------------------------------------------------------------------------------------------

|0.050000|1.056467308549537520|1.056467308549537965|0.000000000000000444| 9|

----------------------------------------------------------------------------------------------------------

|0.100000|1.116123174033904597|1.116123174033904375|0.000000000000000222| 11|

----------------------------------------------------------------------------------------------------------

|0.150000|1.179147645681366319|1.179147645681366541|0.000000000000000222| 12|

----------------------------------------------------------------------------------------------------------

|0.200000|1.245730939615517618|1.245730939615517396|0.000000000000000222| 12|

----------------------------------------------------------------------------------------------------------

|0.250000|1.316074012952492600|1.316074012952492378|0.000000000000000222| 13|

----------------------------------------------------------------------------------------------------------

|0.300000|1.390389170315909251|1.390389170315909473|0.000000000000000222| 14|

----------------------------------------------------------------------------------------------------------

|0.350000|1.468900704600073714|1.468900704600073714|0.000000000000000000| 14|

----------------------------------------------------------------------------------------------------------

|0.400000|1.551845573915360044|1.551845573915359822|0.000000000000000222| 15|

----------------------------------------------------------------------------------------------------------

|0.450000|1.639474116758873334|1.639474116758872890|0.000000000000000444| 15|

----------------------------------------------------------------------------------------------------------

|0.500000|1.732050807568877415|1.732050807568877193|0.000000000000000222| 15|

----------------------------------------------------------------------------------------------------------

|0.550000|1.829855054943345438|1.829855054943345438|0.000000000000000000| 16|

----------------------------------------------------------------------------------------------------------

|0.600000|1.933182044931762933|1.933182044931762933|0.000000000000000000| 16|

----------------------------------------------------------------------------------------------------------

|0.650000|2.042343631945351401|2.042343631945351401|0.000000000000000000| 17|

----------------------------------------------------------------------------------------------------------

|0.700000|2.157669279974594012|2.157669279974593124|0.000000000000000888| 17|

----------------------------------------------------------------------------------------------------------

|0.750000|2.279507056954777067|2.279507056954777511|0.000000000000000444| 17|

----------------------------------------------------------------------------------------------------------

|0.800000|2.408224685280691890|2.408224685280692334|0.000000000000000444| 18|

----------------------------------------------------------------------------------------------------------

|0.850000|2.544210651641050980|2.544210651641050980|0.000000000000000000| 18|

----------------------------------------------------------------------------------------------------------

|0.900000|2.687875379522286501|2.687875379522286501|0.000000000000000000| 18|

----------------------------------------------------------------------------------------------------------

|0.950000|2.839652467920477541|2.839652467920477985|0.000000000000000444| 19|

----------------------------------------------------------------------------------------------------------

|1.000000|3.000000000000000000|3.000000000000000000|0.000000000000000000| 19|

----------------------------------------------------------------------------------------------------------

**Вывод**

В процессе выполнения этот задания, я получил навыки вычисления и дальнейшего использования так называемого «машинного эпсилон». Я на практике применил знания, полученные на занятиях ранее. Также я научился выполнять организацию вывода результатов выполнения программы в виде таблицы, используя элементы псевдографики, и конкретно - вывод вещественных чисел с заданной точностью. Из-за того, что существует понятие ограниченности разрядной сетки, вещественные числа имеют диапазон представления в памяти компьютера. Что неизбежно приводит к тому, что в вычислениях в окрестностях границ этого диапазона возникают погрешности. Пожалуй, наиболее наглядный пример этот можно привести из самой работы, вычисление машинного эпсилон: при многократном делении 1 на 2 в конечном итоге мы получаем минимальное число, которое может хранить машина. Программа не имеет прикладного применения, так как вычисление значения функции по ряду Тейлора требует много процессорного времени что не эффективно в перспективе глобального применения.